VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

INTEGRACIÓN DEL CONTEXTO TÉCNICO Y TECNOLÓGICO AL PROCESO DE DESARROLLO PARA LA GENERACIÓN DE SOFTWARE CON CALIDAD

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A:

GUSTAVO ADOLFO ARELLANO SANDOVAL

DIRECTORA DE LA TESIS: DRA. HANNA OKTABA

MÉXICO, D.F. 2003.

A

Dedicatorias

A la memoria y espíritu trabajador y siempre honesto de mi padre.

Agradecimientos.

Agradezco el apoyo y atención recibida durante tres años de todas las personas que me ayudaron a hacer realidad este sueño y en especial, a la Doctora Hanna Oktaba quién fue paciente y dedicada en esta ardua labor.

Agradezco a cada uno de mis sinodales el haber aceptado serlo, consumir tiempo en la revisión y contribuir a mejorar con valiosos comentarios este trabajo. Lupita, Fer, Reynaldo y Gustavo. ¡Gracias!

Agradezco la amistad incondicional de mis grandes amigas Lulú, Viole, Amalia, Juanita y claro, Don Mardonio quienes siempre me hicieron sentir como en mi casa.

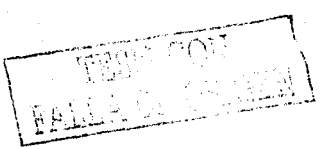
El soporte técnico fue invaluable. Gracias Dante y Pedro.

Y en primer lugar, agradezco a Dios todo, pero en especial la oportunidad de haberme permitido conocer a grandes seres humanos, que fueron mis compañeros de salón y de desvelos: Paco Noguez, Mauricio Espinoza, Ana Pérez. Un gran equipo de trabajo, pero sobre todo, un gran equipo de amigos. ¡Nos vemos en el camino!

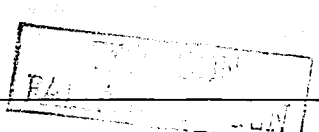
A mi madre, quién con su ejemplo siempre me guió por el buen camino.

A mi hermano, por ser amigo y soportarme tal como soy.

A Ale, por creer en mi, por sufrir carencias y desvelos junto a mi, por amarme y por tratar de ser mejor cada día..



Índice



1

Introducción

El presente trabajo se intitula "*Integración del contexto técnico y tecnológico al proceso de desarrollo para la generación de software con calidad.*" A continuación, se presentan los antecedentes, objetivos y distribución de capítulos así como una breve descripción del contenido de cada uno de ellos.

1.1 Objetivos

Este documento pretende hacer evidente la importancia de las propuestas o sugerencias técnicas que se presentan a lo largo del ciclo de vida de un desarrollo de software y mostrar la relevancia de tales propuestas en la obtención de productos de software con calidad.

Son cuatro los objetivos principales de este documento:

- Definir el concepto de 'Hojas de Recomendaciones Técnicas' y su finalidad.

- Proponer la integración de las mismas al proceso de desarrollo de software y justificar el porqué de ésta propuesta.

- Hacer evidente el impacto en la calidad de software que se genera haciendo uso de buenas técnicas de desarrollo en conjunción con el conocimiento de tecnologías de punta, vía las 'Hojas de Recomendaciones Técnicas' y mostrar que las HRT's influyen en el proceso de toma de decisiones durante el desarrollo de un sistema.

- Ilustrar, vía un caso de estudio, la influencia de esta propuesta en la calidad del producto final.

1.2 Antecedentes

El proceso de desarrollo de software, desde sus inicios, ha evolucionado rápidamente y esto ha conllevado al surgimiento de numerosas metodologías, modelos y estándares de organización de las diversas actividades a realizar en dicho proceso.

Estas metodologías de desarrollo suponen que se posee algún tipo de tecnología y que se domina la técnica necesaria para la implementación de la misma. Sin embargo, cada nuevo desarrollo de software requiere del uso de técnicas específicas que no pueden ser previstas o extraídas íntegramente de otras experiencias de desarrollo.

Dado que cada desarrollo requiere del uso de técnicas específicas y de que la técnica empleada depende de la tecnología en la que se basa, no es posible sólo con el análisis de aspectos funcionales (i.e. casos de uso) definir qué tipo de técnica (y en ocasiones, qué tipo de tecnología) debe ser empleada. Más aún, muchos aspectos no funcionales (como el que un desarrollo sea usable, mantenible, escalable, modularizable, y demás aspectos clave en la definición de un sistema con calidad) dependen en mucho del empleo de una buena técnica de implementación de tecnologías.

Por lo anterior y dado que, el conocimiento de tecnologías no garantiza que el uso de las mismas sea adecuado (aún bajo un esquema metodológico) es por lo que el proceso de desarrollo de software debe considerar en cada una de sus fases todos aquellos aspectos de implementación de tecnología (i.e. técnicas) con el fin de contribuir a la generación de software con calidad.

El presente trabajo presenta un nuevo artefacto: Las hojas de recomendaciones técnicas, la vía que se propone para integrar el contexto técnico al proceso de desarrollo. Las hojas de recomendaciones técnicas conforman una herramienta que será de utilidad en la toma de decisiones estratégicas y en el proceso de mitigación de riesgos identificados. Adicionalmente, lo anterior está ejemplificado con un caso de estudio real que muestra cómo estas recomendaciones impactaron positivamente a la calidad del producto final del desarrollo y a la del proceso de desarrollo del mismo.

1.3 Propuestas

Con base en criterios técnicos y tomando a TSPi[1] como línea base, se propone incrementar (o modificar) artefactos, fases y actividades que serán de utilidad en el desarrollo del caso de estudio y posiblemente en otros desarrollos independientes al mismo vía una propuesta propia: "Las hojas de recomendaciones técnicas."

1.4 Organización del documento

El presente trabajo se encuentra dividido en ocho capítulos y tres apéndices. Los tres primeros capítulos son introductorios. En ellos se exponen conceptos generales y definiciones de elementos que serán de utilidad a lo largo del trabajo.

Los capítulos del cuatro al siete, son propiamente el primer ciclo del proceso de desarrollo en el que se basó el caso de estudio.

El capítulo ocho resume el segundo ciclo y posteriormente se presentan una serie de conclusiones generales. Finalmente, se adjuntan cuatro apéndices que

contienen el producto final de las actividades realizadas a lo largo de este documento, manuales de instalación del caso de estudio, convenciones de codificación e instrucciones de instalación del software requerido para el correcto funcionamiento de las aplicaciones desarrolladas en este trabajo.

Capítulo I. Introducción. Se ofrece una idea general de lo que es el presente trabajo, cuales son sus antecedentes, objetivos y propuestas para lograrlos.

Capítulo II. El proceso de desarrollo de software. Se presentan tres procesos de desarrollo de software: TSPi[1], RUP[2], y eXtreme Programming[3]. El objetivo es extender el primero con fases, artefactos, roles y actividades de los dos últimos. Adicionalmente, se propone agrupar las fases de TSPi en cuatro grandes flujos: Concepción, Elaboración, Construcción y Transición.

Capítulo III. Hojas de Recomendaciones Técnicas. Se define el concepto de 'hojas de recomendaciones técnicas' como un nuevo artefacto que servirá como herramienta en el proceso de mitigación de riesgos y a la toma de decisiones durante el desarrollo de un sistema. Adicionalmente se define un proceso administrativo para el control y seguimiento de las recomendaciones que es complementado con un aplicativo WEB que ayuda a automatizar este proceso.

Capítulo IV. Flujo de concepción. Este flujo comprende las fases de factibilidad y lanzamiento. Con base en el caso de estudio **SIPU©** [8] se presentan las fases de factibilidad y lanzamiento que fueron llevadas a cabo en el desarrollo original.

Capítulo V. Flujo de elaboración. Comprende las fases de estrategia, planeación y requerimientos. Se presentan estrategias y planes de ejecución de éstas incluyendo calendarios de análisis, diseño, implementación y pruebas del sistema. En la parte final, (que en su mayoría contiene casos de uso) se incluye la fase requerimientos. En ésta última fase se plantean una serie de estrategias para realizar entrevistas efectivas, de las que se derivarán una serie de requerimientos explícitos, otros implícitos y en su caso, argumentos

para justificar la no inclusión de ciertos requisitos hechos por el cliente, todo lo anterior modelado con UML (Ver [4]).

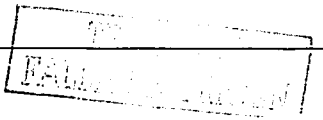
Capítulo VI. Flujo de construcción. Incluye las fases de análisis, diseño e implementación. En este capítulo, la fase de Diseño, concebida originalmente como una sola fase por TSPi, se dividirá en dos: La fase de Análisis y la Fase de Diseño Se propone mantener ambas vistas (con su respectiva administración de configuración) a lo largo del desarrollo, con el fin de tener en todo momento una clara visión del negocio (diagramas de análisis) y de la implementación técnica (diagramas de diseño).

La última fase (que es la de implementación y pruebas) sugiere desarrollar e inmediatamente probar cada subproducto con un proceso incremental de prueba. Un producto puede ser la unión de dos o más subproductos y debe ser sujeto del proceso de prueba incremental diseñado para este nuevo producto.

Capítulo VII. Post mortem o proceso de Mejora continua. Se realiza un análisis en retrospectiva de aquellos aspectos de sistema susceptibles de ser mejorados. Se realiza un registro de esto y se prepara el ambiente para una nueva iteración en donde serán aplicados los criterios de mejora recién definidos. Adicionalmente, se hace un recuento de las hojas de recomendaciones técnicas generadas a lo largo del primer ciclo con el fin de llevar un registro de las recomendaciones nuevas, las reutilizadas de otros desarrollos y las inconclusas para así detectar tanto las debilidades como las fortalezas adquiridas.

Capítulo VIII. Segundo ciclo. En este capítulo, se presentan los aspectos que deberán ser considerados en el segundo ciclo. Las diferencias que resaltan en ésta segunda etapa y las recomendaciones pertinentes al conjunto de actividades a realizar.

Conclusiones. Se presenta una reflexión final acerca del presente trabajo mostrando los beneficios que se obtuvieron al llevar de ésta manera un proceso de desarrollo.



Apéndices. Los apéndices incluirán información complementaria y necesaria para el correcto funcionamiento e instalación del caso de estudio, del administrador de hojas de recomendaciones técnicas y documentación referente a las convenciones de código, y demás software requerido de terceros.

A continuación se presenta un esquema de la organización conceptual del presente documento:

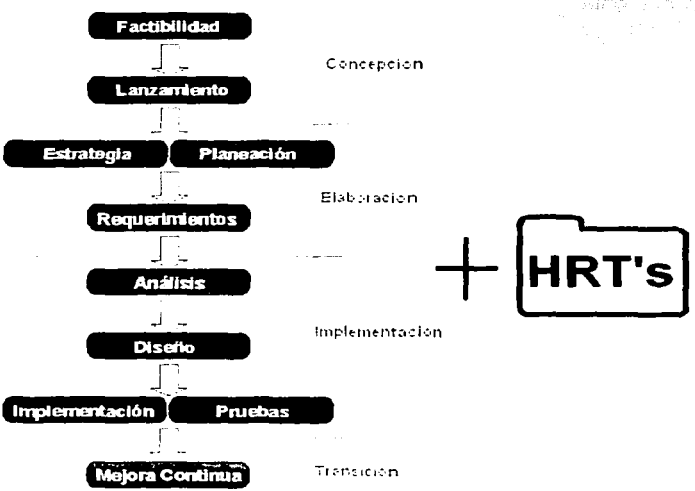


Figura 1.1

1.5 Resumen capítulo

La tesis presenta una propuesta de integración de un nuevo artefacto al proceso de desarrollo de software con el fin de incrementar la calidad final del producto y apoyar al proceso de toma de decisiones durante el desarrollo del

mismo. Se presentan argumentos que justifican ésta propuesta y se aplica lo propuesto a un caso de estudio real.

Este trabajo también incluye una aplicación de soporte a la administración de las hojas de recomendaciones técnicas. Esta aplicación desarrollada en un entorno WEB podrá ser utilizada por cualquier individuo u organización que esté interesada en hacerlo.

Adicionalmente, este trabajo pretende servir como un manual práctico, para el sector académico en términos técnicos y tecnológicos y a la vez, una referencia metodológica para el sector industrial en términos de organización y administración de equipos de desarrollo incluyendo el detalle de la secuencia de desarrollo de un sistema de información real (nuestro caso de estudio) que ejemplifique los conceptos presentados.

2

El proceso de desarrollo de software

2.1 El proceso de desarrollo de software

Un proceso es una guía que define *quién* está haciendo *qué*, *cuándo* y *cómo* para alcanzar una cierta meta. La principal meta en la ingeniería de software, es construir productos *satisfactorios* (de software) o modificar los ya existentes para mejorarlos. Un proceso efectivo guía en el desarrollo eficiente de software con *calidad*. Éste captura y presenta las mejores prácticas que el actual *estado de arte* permite. En consecuencia, éste reduce riesgos e incrementa predictibilidad.

Adicionalmente, un proceso debe servir como una guía para todos los participantes: clientes, usuarios, desarrolladores, etc., y debe estar a la disposición de todos los involucrados de manera que éstos puedan entender su rol en el desarrollo a efectuar.

Un proceso de desarrollo debe ser capaz de evolucionar a través de los años. Durante esta evolución, éste debe limitar su alcance en cualquier momento dado, a las realidades que las tecnologías, técnicas, herramientas, recursos humanos y patrones organizacionales permitan en ese momento.

Actualmente existen diversos procesos de desarrollo. En este documento se presentan tres de ellos debido a que diversos elementos de cada uno son utilizados y propuestos para que formen parte de la metodología TSPi.

Los procesos de desarrollo que en este documento se presentan son: TSPi (Team, Software Process [1]), RUP (Rational Unified Process [2]) y XP (eXtreme Programming [3]) y serán discutidos en la siguiente sección. A continuación, en la figura 2.1, se presenta un diagrama general del proceso de desarrollo, basado en diagrama del artículo [14]:

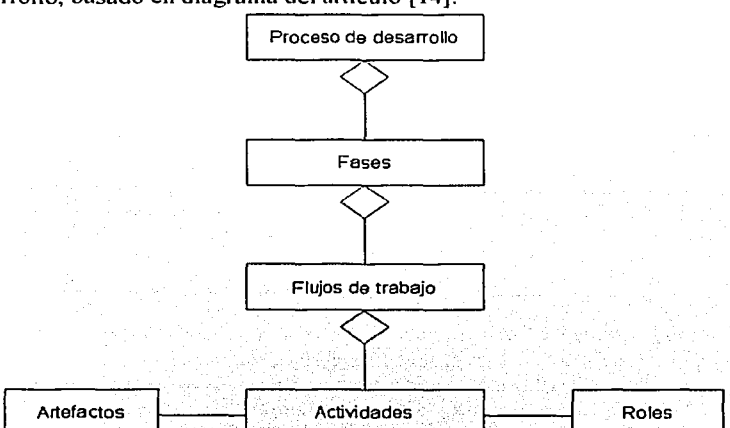


Figura 2.1

Como se mencionó en el capítulo anterior, la técnica de diagramación es en UML[4] y el anterior diagrama de clases puede ser entendido de la siguiente manera:

- Cada proceso de desarrollo se compone de varias fases
- Cada fase se compone de varios flujos de trabajo
- Cada flujo de trabajo se compone de varias actividades
- Cada actividad es realizara por diversos roles
- Cada actividad genera diversos artefactos

A continuación, en la tabla 2.1, se presenta una breve explicación para cada término:

Elemento	Descripción
Fase	Una fase es una etapa en el proceso de desarrollo en la que se realiza un tipo específico de actividades con un objetivo también específico.
Flujo de Trabajo	Un flujo de trabajo es un conjunto de actividades relacionadas entre si, y que en conjunto integran una fase. Dicho de otra manera, un flujo de trabajo es un agrupador de actividades relacionadas entre si y una fase contiene varios flujos de trabajo.
Actividad	Una actividad es una tarea que debe ser llevada a cabo para satisfacer cierto fin.
Rol	Un rol es la definición de actividades, responsabilidades y alcances específicos que pueden ser asignados a un cierto individuo durante el desarrollo. Un individuo puede tener más de un rol y un rol puede ser asignado a más de un solo individuo.
Artefacto	Un artefacto es el resultado del trabajo de un individuo con cierto rol al realizar una actividad específica.

Tabla 2.1

Adicionalmente, un proceso de desarrollo puede ser iterativo e incremental:

Un proceso de desarrollo se dice que es iterativo si éste es repetido varias veces a lo largo de un desarrollo de un producto de software.

Un proceso de desarrollo se dice que es incremental si su intención es generar versiones inicialmente simples, pero funcionales del producto y en posteriores iteraciones incrementar tal funcionalidad o refinar el producto obteniendo así cada vez versiones mejoradas (o extendidas) del mismo.

TSPi, RUP y eXtreme Programming son procesos de desarrollo iterativos e incrementales. Éstos serán vistos con mayor detalle en las siguientes secciones.

En el presente trabajo, el proceso base será TSPi y éste será extendido con elementos de las metodologías de desarrollo RUP y eXtreme Programming.

En cada proceso de desarrollo se espera, entre otras cosas más, que su uso conlleve a la generación de software con calidad. Sin embargo, cada uno de los procesos de desarrollo mencionados establece en sus términos (como se verá mas adelante) qué es calidad y esto provoca que no exista unificación de este concepto entre procesos de desarrollo. Lo mismo ocurre al reflexionar acerca de lo *satisfactorio* que un producto puede ser; o la forma de saber si algo ha sido realmente *mejorado*.

Actualmente, existen estándares (de calidad, de proceso, de control de producción, etc.) aceptados internacionalmente y también es posible encontrar diversos procesos que guían en el desarrollo de sistemas de manera organizada y metódica en la ejecución de procesos y subprocesos, asignación de roles y tareas distribuidas en fases de desarrollo.

El estándar de calidad en el producto que se considerará a lo largo de todo este documento será el ISO/IEC 9126-1 [5], comentado al final de este capítulo.

2.2 Tres procesos de desarrollo concretos

A continuación se mencionan tres procesos de desarrollo, de los cuales se subrayarán elementos que serán de especial importancia en el resto del presente documento.

2.2.1 Team Software Process TSPi

TSPi o Team Software Process[1], por sus siglas en inglés, es una metodología de desarrollo elaborada por Watts Humphrey para la Carnegie Mellon University. El principal objetivo de TSPi es proveer un marco para el uso de útiles métodos de ingeniería para el desarrollo de software. TSPi es un proceso iterativo que define roles, actividades, artefactos, fases y ciclos durante su desarrollo.

TSPi define cinco roles:

- Líder de proyecto (LP)
- Administrador de desarrollo (AD)
- Administrador de planeación (AP)
- Administrador de calidad (AQ)
- Administrador de configuración y soporte (AS)

Sus actividades están descritas en la tabla 2.2:

Actividad / Rol	LP	AD	AP	AC	AS
Construir y mantener un equipo efectivo	✔				
Resolver diferencias entre los miembros del equipo	✔				
Llevar un registro y realizar reportes del progreso del desarrollo	✔				
Actuar como un facilitador de las reuniones de trabajo	✔				
Mantener una libreta de notas del proyecto	✔				
Auxiliar en la asignación de tareas	✔				
Guiar el trabajo del desarrollo		✔			
Guiar la planeación para el equipo y su seguimiento			✔		
Proveer soporte para el equipo				✔	
Actuar como moderador en las inspecciones				✔	
Mantener los estándares del equipo y el glosario				✔	
Generación y administración de minutar				✔	

Alertar al equipo en problemas de calidad				✔	
Obtención de herramientas necesarias y dar soporte					✔
Administración del control de versiones					✔
Guiar el pizarrón de control de cambios					✔
Impulsar el reuso de elementos en el equipo					✔
Administración de riesgos					✔
Desarrollar el producto					✔
Hacer planes personales	✔	✔	✔	✔	✔
Llevar registro del trabajo personal	✔	✔	✔	✔	✔
Producir productos de calidad	✔	✔	✔	✔	✔
Seguir disciplinadas prácticas personales	✔	✔	✔	✔	✔

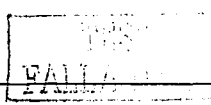
Tabla 2.2

Así mismo, TSPi define ocho fases. Sus actividades están descritas en la siguiente tabla:

Fase	Actividad
Lanzamiento	• Se construye el equipo. • Se determinan los roles de los miembros. • Se establecen metas de equipo. • Se guía a los miembros del equipo en la selección de sus prácticas de trabajo.
Estrategia	• Se define un mínimo de funcionalidad que hace al producto un subconjunto funcional del desarrollo final. • Se construye una base fácilmente mejorable. • Los productos de cada ciclo son de alta calidad y fáciles de probar. • El diseño del producto tiene una estructura modular que permite a los miembros del equipo trabajar independientemente.
Planeación	• Se generan documentos que definen la distribución y calendarización de las actividades a realizar. • Se mantiene un control acerca del avance por individuo y

	por proyecto.
Requerimientos	• Se describe (en nuestras propias palabras) la funcionalidad que se desea tenga el producto. • Se produce un documento con criterios claros y no ambiguos para la evaluación del producto terminado así como la guía para verificar que el producto realiza lo que se supone que debe hacer.
Diseño	• Se produce la estructura de un diseño global • Se divide este diseño global en sus principales componentes • Los miembros del equipo detallan el diseño de cada componente • Se reúnen éstos últimos detalles de diseño para generar la especificación de diseño del sistema. • Se produce e inspecciona el plan de integración de componentes en dónde se define cómo las partes del producto interactúan y cómo ésta van a ser ensambladas.
Implementación	• Se planea la implementación • Se codifica • Se inspecciona el código • Se prueban las unidades • Se revisa la calidad del producto • Se libera el producto
Pruebas y Documentación	• Se evalúa al producto, no se corrige. • Se generan planes de prueba y se define el orden de las mismas. • Se genera la documentación, manuales técnicos y de operación.
Post mortem	• Se revisa el trabajo del equipo para asegurar el haber realizado todo el trabajo necesario. • Se examina lo hecho para documentar lo bien y mal hecho, para realizarlo mejor en el siguiente ciclo.

Tabla 2.3



2.2.1.1 Calidad en TSPi

TSPi establece un plan de calidad con el fin de promover la generación de productos de software con alta calidad. El plan, que será presentado con detalle en la fase de planeación y en dónde se generarán los artefactos requeridos, está basado en la siguiente tabla, conocida como criterios de calidad de TSPi:

Medida	Meta	Comentarios
Porcentaje libre de defectos		
Compilación	> 10%	
Pruebas unitarias	> 50%	
Pruebas de integración	> 70%	
Pruebas de sistema	> 90%	
Defectos por LOC		
Total de defectos inyectados	75 –100	Usar de 100-200 sin PSP
Compilación	< 10	Sólo defectos vía compilador
Pruebas unitarias	< 5	Sólo defectos mayores
Construcción e integración	< 0.5	Sólo defectos mayores
Pruebas de sistema	< 0.2	Sólo defectos mayores
Radios de defecto		
Revisión del diseño detallado (o DLD por sus siglas en inglés: Detail Level Design) (defectos en pruebas unitarias	> 2.0	Sólo defectos mayores
Defectos en revisión de código (Defectos de compilación)	>2.0	Sólo defectos mayores
Razón de tiempo de desarrollo		
Inspección de requerimientos entre tiempo de requerimientos	> 0.25	Incluye tiempo de licitación
Inspecciones de diseño de alto nivel (o HLD por sus siglas en inglés: *High Level Design*) / tiempo de HLD	> 0.5	Sólo trabajo de diseño, no estudios
DLD / tiempo de codificación	> 1.00	
Revisión de DLD / tiempo de DLD	> 0.5	
Revisión de código / tiempo de codificación	> 0.5	

Tasas de revisión e inspección		
Páginas de requerimientos por hora	< 2	Texto a un solo espacio
Páginas de HDL por hora	< 5	
Líneas de texto DLD por hora	< 100	3 LOC = 1 pseudo LOC
Líneas de código (o LOC por sus siglas en inglés: *Lines of Code*) por hora	< 200	
Tasas de inyección de defectos		
Páginas de requerimientos por hora	0.25	Sólo defectos mayores
Páginas de HDL por hora	0.25	Sólo defectos mayores
Defectos DLD por hora	2.0	Sólo defectos de diseño
Defectos de código por hora	4.0	Sólo defectos mayores
Defectos de compilación por hora	0.3	Sólo defectos vía compilador
Defectos en pruebas unitarias por hora	0.2	Sólo defectos mayores
Tasas de remoción de defectos		
Defectos en la inspección de requerimientos por hora	0.5	Sólo defectos mayores
Defectos en la inspección de HLD/ hora	0.5	Sólo defectos mayores
Defectos en la revisión de DLD/hora	2.0	Sólo defectos de diseño
Defectos en la inspección de DLD/hora	0.5	Sólo defectos de diseño
Defectos en la revisión de código por hora	6.0	Sólo defectos mayores
Defectos en la inspección de código por hora	1.0	Sólo defectos mayores
Avance de fase		
Inspección de requerimientos	≈ 70%	No cuentan comentarios editoriales
Revisiones de diseño e inspecciones	≈ 70%	
Revisiones de código e inspecciones	≈ 70%	Usar listas de verificación personales
Compilación	≈ 50%	
Menos de 5 defectos por KLOC en pruebas unitarias	≈ 90%	
Menor a 1.0 defectos por KLOC en la construcción, integración y pruebas de sistema	≈ 80%	
Avance de proceso		
Antes de compilar	> 75%	
Antes de pruebas unitarias	> 85%	
Antes de construcción e integración	> 97.5%	Máximo 1 defecto para productos pequeños
Antes de pruebas de sistema	> 99%	Máximo 1 defecto para productos pequeños

Tabla 2.4

De lo anterior, podemos observar que la base para la obtención de productos con calidad, según TSPi es el apego a las medidas que establece su lista de criterios de calidad. Como se mencionó anteriormente, TSPi (como proceso de desarrollo) es el eje principal del presente trabajo. Éste será extendido con la inclusión (remoción o modificación) de fases, artefactos y actividades extraídas de otros procesos. En el caso de calidad, los lineamientos estarán basados en el ISO/IEC 9126-1 comentado mas adelante.

La siguiente figura (figura 2.2) muestra la forma iterativa del proceso TSPi. En este esquema se puede apreciar que el número de ciclos puede variar según se requiera de un desarrollo a otro. Es también posible apreciar que las ocho fases del proceso están consideradas en cada ciclo y éstas se presentan en su orden de ejecución.

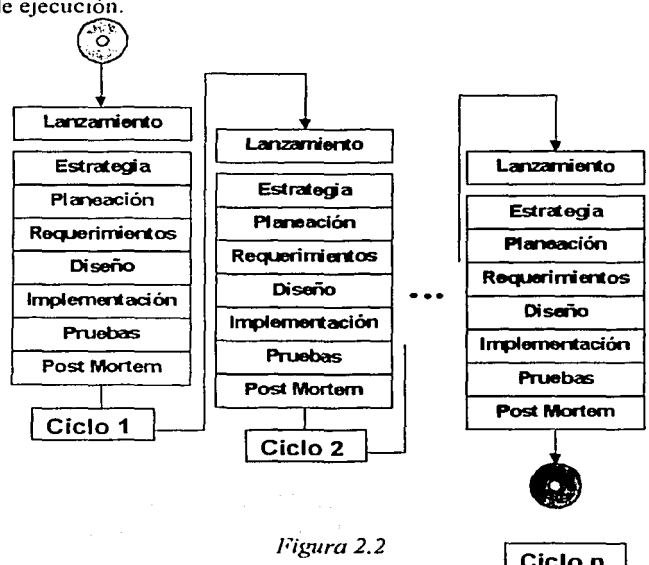


Figura 2.2

2.2.2 Rational Unified Process RUP

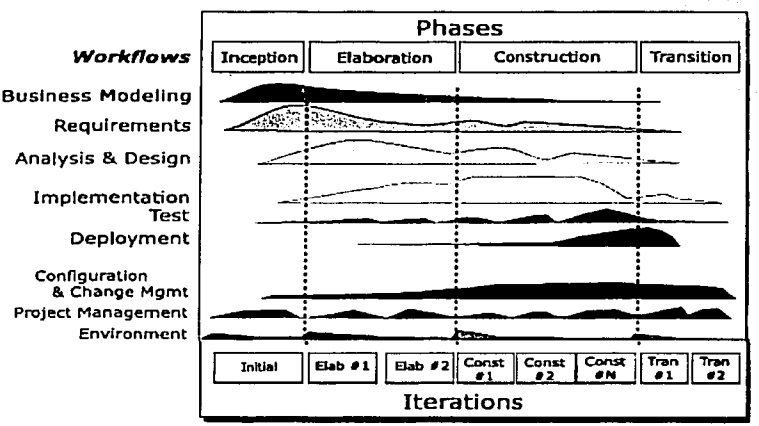


Figura 2.3 (extraído del libro Unified Process Development [13])

El segundo proceso de desarrollo que se presenta es el "Rational Unified Process" (Proceso Unificado Racional). Éste provee una aproximación disciplinada a la asignación de tareas y responsabilidades en una organización de desarrollo de software. Su meta es asegurar la producción de software de alta calidad que satisfaga las necesidades de sus usuarios finales, en un tiempo y costo predictibles.

El "Rational Unified Process" (o RUP, como se le mencionará en adelante) es además un proceso de desarrollo iterativo e incremental basado en la tecnología de orientación a objetos. La figura anterior (figura 2.3) muestra la arquitectura global del RUP:

El RUP tiene dos dimensiones:

- El eje horizontal representa tiempo y muestra los aspectos de ciclo de vida del proceso tal y cómo éste se desarrolla.
- El eje vertical representa los flujos de trabajo principales, los cuales agrupan actividades relacionadas entre si.

La primera dimensión representa el aspecto dinámico del proceso, como éste es llevado y es expresado en términos de fases e iteraciones.

La segunda dimensión representa el aspecto estático del proceso: cómo éste es descrito en términos de componentes, actividades, flujos de trabajo, artefactos y roles.

La gráfica muestra cómo el énfasis varía con el tiempo. Por ejemplo, en las primeras iteraciones, se usa más tiempo en requerimientos y en las últimas, más tiempo en implementación.

A continuación se incluye una breve descripción del objetivo de cada fase:

Fase	Objetivo
Concepción	Evaluar diversos aspectos técnicos y de negocio con base en los que se decidirá si el proyecto continuará o será cancelado.
Elaboración	Establecer una línea base de la arquitectura del sistema para proveer una base estable para la mayor parte del esfuerzo de diseño e implementación en la fase de construcción.
Construcción	Clarificar los requerimientos restantes y completar el desarrollo del sistema basado en la línea base de la arquitectura.
Transición	Asegurar que el software estará disponible para sus usuarios finales.

Tabla 2.5

La fase de Concepción será de especial importancia para el presente trabajo, ya que ésta será integrada en el marco de TSPi.

Cada fase se compone de nueve flujos de trabajo. A continuación, se presenta la tabla 2.6 que incluye una descripción para cada flujo:

Flujo	Propósito
Modelado del negocio	• Entender la estructura y la dinámica de la organización en la cual un sistema va a ser instalado (la organización adquiriente). • Entender los problemas actuales en la organización adquiriente e identificar mejoras potenciales. • Asegurar que clientes, usuarios finales y desarrolladores tienen un entendimiento común acerca de la organización adquiriente. • Derivar los requerimientos del sistema necesarios para soportar a la organización adquiriente.
Requerimientos	• Establecer y mantener acuerdos entre los clientes y otros involucrados acerca de lo que el sistema debe hacer. • Proveer a los desarrolladores del sistema con un mejor entendimiento del los requerimientos del sistema. • Definir los límites (el alcance) del sistema. • Proveer una base para la planeación de los contenidos técnicos de las iteraciones. • Proveer una base para la estimación del costo y tiempo de desarrollo del sistema. • Definir una interfaz de usuario para el sistema, enfocándose en las necesidades y metas de los usuarios.
Análisis y Diseño	• Transformar los requerimientos en un diseño tipo "El cómo debe ser". • Proponer una arquitectura robusta para el sistema. • Adaptar el diseño para que encaje en el ambiente de

	implementación y hacerlo para que se obtenga el mejor desempeño.
Implementación	• Definir la organización del código, en términos de la implementación de subsistemas organizado en capas. • Implementar clases y objetos en términos de componentes (archivos fuente, binarios, ejecutables y otros). • Probar los componentes desarrollados como unidades. • Integrar los resultados producidos por implementadores individuales (o equipos) en un programa ejecutable.
Pruebas	• Verificar la interacción entre objetos. • Verificar la propia integración de todos los componentes del producto. • Verificar que todos los requerimientos han sido correctamente implementados. • Identificar y asegurar que los defectos han sido removidos antes de la instalación del sistema.
Instalación	• Asegurar que el producto de software está disponible para los usuarios finales.
Administración del control de cambios	• Asegurar que la última versión del sistema es la correcta y mantener sincronía entre las distintas versiones de cada documento.
Administración del proyecto	• Proveer un marco para la administración de proyectos de desarrollo intensivo de software. • Proveer una guía práctica para planear, ejecutar y monitorear proyectos. • Proveer un marco para la administración de riesgos.
Ambiente	• Proveer a la organización de desarrollo el ambiente de desarrollo de software (tanto herramientas como procesos) que soportarán al equipo de desarrollo.

Tabla 2.6

Finalmente, RUP define cinco roles principales; cada uno de éstos tiene a su vez una gran variedad de subroles. A continuación se presenta la lista de éstos:

Rol	Subrol
Analista	• Analista del proceso de negocio • Diseñador de negocio • Revisor del modelo de negocio • Revisor de requerimientos • Analista de sistemas • Especificador de requerimientos • Diseñador de interfaz humana
Desarrollador	• Arquitecto de software • Revisor de arquitectura • Diseñador de cápsulas • Revisor de código • Diseñador de bases de datos • Revisor de diseño • Diseñador • Implementador • Integrador
Probador	• Diseñador de pruebas • Probador
Administrador	• Admin. de control de cambios • Admin. de configuración • Admin. de instalación • Admin. de proyecto • Ingeniero de proceso • Revisor de proyecto
Otro	• Desarrollador de cursos • Artista gráfico • Administrador de sistema • Escritor técnico • Especialista de herramientas

Tabla 2.7

2.2.2.1 La calidad en RUP

Según RUP, la calidad del producto es la producida por el proceso. En el desarrollo de software, el producto es la agregación de varios artefactos, incluyendo:

- Código ejecutable instalable, que es quizá la parte más visible de los artefactos, y por la cual el proyecto existe. Este es el producto primario que provee valor al cliente.
- Artefactos no ejecutables instalables, como manuales de usuario y materiales de cursos.
- Ejecutables no instalables, tales como scripts de prueba y herramientas de desarrollo creadas para soportar la aplicación.
- Artefactos no ejecutables no instalables, como los planes de implementación, planes de prueba y modelos varios.

Debido a que varios artefactos están basados en otros, la calidad de todos los artefactos está relacionada en cierto grado. Por esta razón, la calidad de cada artefacto debe ser medida y evaluada.

2.2.2.2 Calidad en productos ejecutables (instalables y no instalables)

Los artefactos ejecutables están descritos por sus requerimientos, como casos de uso o requerimientos suplementarios, como desempeño o restricciones de mercado. Para medir y asegurar calidad, esos requerimientos deben ser conocidos y puestos de una manera clara, concisa y verificable. Para el software no todos los requerimientos serán el objetivo de las pruebas hechas por un rol encargado de pruebas. Para aquellos para los que si lo serán, un diseñador de pruebas debe ser capaz de especificar un método para verificar que el requerimiento ha sido satisfecho, que no está desviado de su intención y que no ha fallado.

La calidad del producto es determinada midiendo y evaluando las siguientes dimensiones de calidad:

- Funcionamiento. El código instalado ejecuta los casos de uso requeridos.
- Desempeño. El código instalado responde en manera y tiempo aceptable y continúa de esa manera cuando es instalado en el ambiente de producción.
- Robustez. Los códigos instalados tienen soporte a fallos durante su ejecución.

Para cada una de esas dimensiones, una o mas pruebas individuales deben ser implementadas y ejecutadas durante uno o mas etapas del desarrollo.

Adicionalmente, la calidad del producto es evaluada midiendo el número y tipo de cambios que han sido hechos al artefacto ejecutable para cada nueva versión del artefacto.

2.2.2.3 Calidad en productos no ejecutables (instalables y no instalables)

La calidad de productos no ejecutables queda establecida por el propósito de los artefactos, meta y estructura de los mismos y se evalúa asegurando que los artefactos cumplen con lo siguiente:

- Consistencia interna dentro y entre los artefactos (uso del lenguaje, terminología, semántica, etc.).
- Acuerdo en las guías, estándares y requerimientos contractuales.

Adicionalmente, la calidad de productos no ejecutables puede ser evaluada por el número y tipo de cambios que han sido hechos al artefacto ejecutable para cada nueva versión del artefacto.

2.2.3 eXtreme Programming

Extreme Programming es una metodología ligera, dedicada a atender las necesidades específicas del desarrollo conducido por un equipo pequeño de desarrolladores enfrentando requerimientos posiblemente vagos y cambiantes.

Los fundamentos de eXtreme Programming (o XP como se le mencionará en adelante) incluyen:

- Distinguir entre las decisiones hechas por los intereses del negocio y aquellas hechas por los involucrados.
- Escribir unidades de prueba antes de programar y mantener todas las pruebas corriendo en todo momento.
- Integrar y probar el sistema completo varias veces al día.
- Producir todo el software organizado en parejas de compañeros de trabajo.
- Iniciar proyectos con un diseño simple que constantemente añade flexibilidad y remueve complejidad innecesaria.
- Poner en producción un sistema mínimo rápidamente y hacerlo crecer en cualquier dirección que incremente su valor.

XP menciona que existen cuatro valores esenciales en una disciplina de desarrollo: comunicación, simplicidad, realimentación y disposición. Así mismo, define cuatro actividades básicas descritas en la tabla 2.8:

Actividad	**Descripción**
Codificar	Inicio del ciclo. Base fundamental del desarrollo.
Probar	Implementar unidades de prueba para cada producto generado.
Escuchar	Escuchar al experto del negocio, para poder obtener datos para comparación con las unidades de prueba.
Diseñar	Con base en lo obtenido en las dos fases anteriores, se diseña un marco mejorado y se prepara el desarrollo para una nueva iteración.

Tabla 2.8

Adicionalmente, XP define doce prácticas a seguir durante todo desarrollo:

- Planear.
- Hacer pequeñas liberaciones de código (diario, si es posible).
- Mantener simpleza global conceptual.
- Generar diseños simples.

- Probar cada producto todo el tiempo (vía las unidades de prueba).
- Refabricación.
- Programar en pares (parejas de programadores).
- Permitir la propiedad colectiva de código.
- Promover continua integración del sistema varias veces al día.
- No trabajar mas de 40 horas por semana.
- Solicitar al menos un cliente en sitio.
- Implantar estándares de codificación.
- Generar soluciones experimentales (SPIKE)

XP incorpora dos artefactos:

Artefacto	Descripción
Historia del cliente	Formato para el seguimiento de una requisición explícita del cliente.
Tarjeta de tareas	Formato para guiar la realización o cumplimiento de una requisición hecha vía una historia del cliente. Generalmente varias tarjetas son necesarias para lograr este fin. En ocasiones, una tarjeta no estará directamente relacionada con una historia particular.

Tabla 2.9

Los aspectos a retomar de XP para ser incorporados a TSPi son básicamente cuatro:

- A cada generación de un subproducto le debe corresponder la generación de una unidad de prueba.
- Los diseños deben ser tan simples como sea posible y la complejidad adicional debe ser eliminada tan ponto como sea identificada.
- Realizar pequeñas liberaciones varias veces al día.
- Soluciones experimentales (SPIKE)

2.2.3.1 Calidad en eXtreme Programming

La calidad en el producto, como la define eXtreme Programming, es una consecuencia del cuidadoso seguimiento de la calidad en el proceso de

desarrollo. Durante el desarrollo de un sistema, cada producto debe ser probado vía *Unit Tests* o unidades de prueba[1] generando información estadística que ayudará a dar administración y seguimiento a la remoción de código defectuoso. eXtreme Programming, sugiere la elaboración de las siguientes gráficas estadísticas.

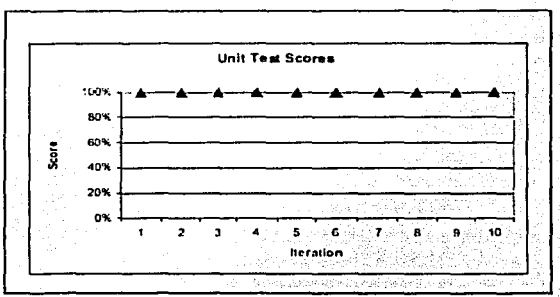


Figura 2.4

La figura 2.4 muestra la estadística de la puntuación obtenida en la realización de las unidades de prueba por iteración. El autor de la metodología sugiere que la gráfica resultante debe reflejar que la puntuación para cada unidad de prueba debe ser del 100%.

La siguiente figura muestra en tonos obscuros el número aceptable de pruebas a realizar por mes y en tonos claros el número real de pruebas realizadas antes de que el producto esté libre de defectos.

[1] Una prueba es básicamente un experimento; al probar la funcionalidad de algo, se propone un resultado, se realiza la prueba y se compara el resultado de ésta con el valor propuesto. Si dichos resultados coinciden, entonces decimos que se efectuó una prueba exitosa.

Existen otras propuestas para definir el concepto de prueba, sin embargo a lo largo de éste documento la definición de 'prueba' a usar será la mencionada en el párrafo anterior.

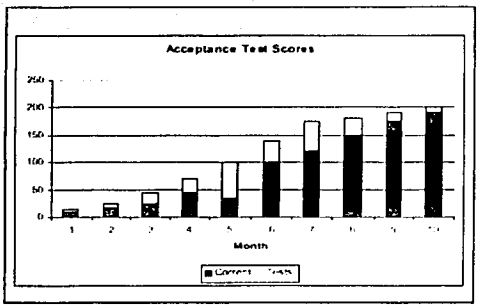


Figura 2.5

Se sugiere la elaboración de un acumulado mensual que muestre la diferencia entre el número de pruebas efectuadas y el sugerido, que es definido por el desarrollador al inicio del proyecto.

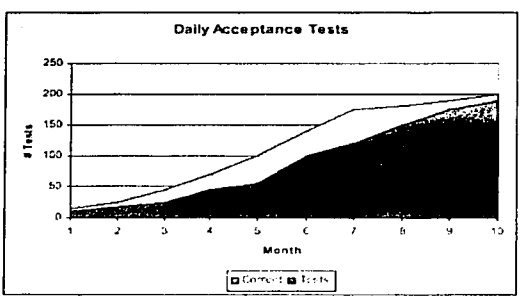


Figura 2.6

Finalmente, en la figura 2.7, se presenta una gráfica de fácil interpretación para el ser humano, en donde se indican, con un punto negro, las pruebas exitosas efectuadas por día y con un punto en blanco las pruebas efectuadas no exitosas en el mismo día. Esto es un instrumento útil en el seguimiento de ejecución de pruebas a lo largo de un desarrollo de software:

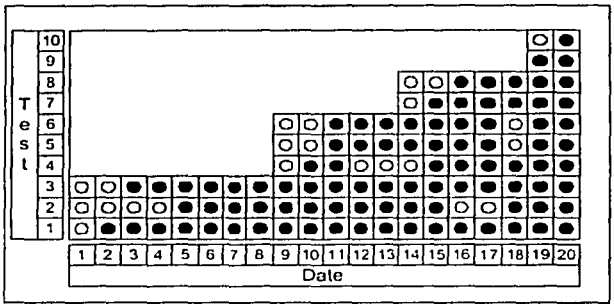


Figura 2.7

2.2.4 TSPi extendido

El siguiente diagrama describe la propuesta de la extensión de TSPi. La motivación para sugerir esta extensión es la experiencia adquirida en el desarrollo de varios proyectos. En esta extensión se incorporan flujos de trabajo, fases, artefactos y prácticas aceptadas y reconocidas por otros procesos de desarrollo, como los mencionados anteriormente y una propuesta propia, que será presentada y discutida en detalle en el siguiente capítulo.

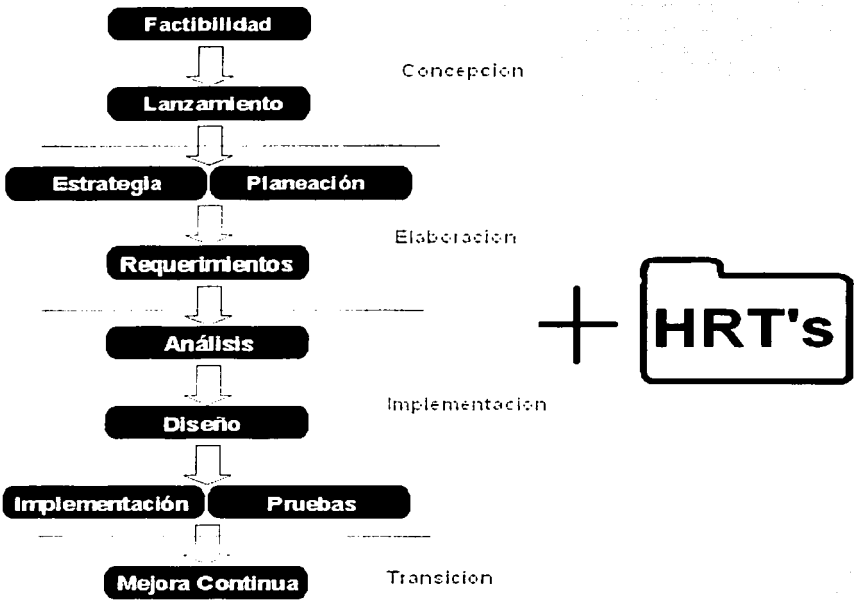


Figura 2.8

Las adiciones y propuestas sugeridas son las siguientes:

Adición / Propuesta	Justificación
Se propone una agrupación de fases en cuatro flujos principales.	Esta división agrupa actividades que están muy relacionadas entre si. Da sentido al propósito de las mismas y delimita el inicio y fin de un conjunto de esfuerzos encaminados a la realización de un objetivo común.
Se añade la fase de Factibilidad de RUP al inicio de cada ciclo.	A diferencia de lo que ocurre en cursos escolares, la fase de factibilidad siempre es

	considerada en la industria debido a que en ella se determina la viabilidad del proyecto.
Se trabajan conjuntamente las fases de Estrategia y Planeación.	La dependencia de ambas fases según lo propuesto por TSPi es alta y de aquí la propuesta de que se trabajen conjuntamente.
Se inserta la fase de Análisis entre la de requerimientos y la de Diseño.	En la fase de análisis se generarán artefactos que permitirán tener una visión clara del proyecto en términos del negocio, antes de que la arquitectura sea aplicada.
Se trabajan conjuntamente las fases de Implementación y Pruebas.	Se sugiere desarrollar e inmediatamente probar cada subproducto con un proceso incremental de prueba. Un producto puede ser la unión de dos o mas subproductos y debe ser sujeto del proceso de prueba incremental propuesto.
Se extiende la fase de Post mortem con registros estadísticos de la identificación de defectos por inspección y detección en la fase de Pruebas.	Se realiza un análisis en retrospectiva de aquellos aspectos del sistema susceptibles de ser mejorados. Se realiza un registro *estadístico* y se prepara el ambiente para una nueva iteración en donde serán aplicados los criterios de mejora continua recién definidos.
Se añade el artefacto "Hoja de recomendaciones técnicas", descrito con mayor detalle en el siguiente capítulo.	Ver capítulo 3.

Tabla 2.10

Los demás aspectos serán trabajados tal y como lo indica el modelo TSPi. Para su seguimiento, podemos basarnos en el sitio guía de TSPi.[6]

2.3 El Proceso de desarrollo y la técnica

Según el Proceso Unificado de Desarrollo: "... El fin último es la obtención de un confiable, robusto y escalable producto de software ..." y añade "... Se requiere tanto del proceso como del lenguaje (la tecnología) para obtener esto, sin embargo el proceso unificado sólo trata el proceso ...".

TSPi y el Proceso Unificado mencionan que es importante la contraparte tecnológica, pero se asume el conocimiento de ésta y también se asume que será bien empleada; es decir, que se posee una buena técnica para su uso. De hecho, se sugiere, hasta cierto punto, una independencia entre proceso y técnica aunque se admite el hecho de que se requiere de ambas para la obtención de productos de software con calidad.

Las metodologías mencionadas anteriormente no analizan la influencia del contexto técnico y tecnológico en el proceso de desarrollo de software con calidad. Sin embargo, para la obtención de un producto con calidad, es necesario el buen uso de la tecnología en un contexto metodológico.

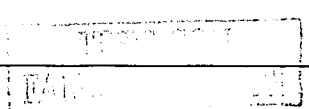
2.4 Calidad en el producto

La calidad siempre está presente como característica de un producto de un proceso de desarrollo bien llevado a cabo. En general, cada proceso de desarrollo establece su propia definición de *un producto con calidad*, por lo que actualmente no existe una única definición adoptada por todos los procesos, tal y como se pudo observar previamente, en las secciones dedicadas a TSPi, RUP y eXtreme Programming.

Sin embargo, sabemos que un modelo de calidad debe ser un:

- Apoyo para las revisiones, verificaciones y validaciones.
- Apoyo para el establecimiento de los objetivos de calidad a nivel de la administración organizacional.
- Marco para los requerimientos de calidad de producto para los procesos.

En este documento, las características de calidad en un producto de software estarán definidas por el ISO/IEC 9126-1. A continuación se presentan las características que según éste debe poseer un producto de software con calidad.



2.4.1 Calidad interna y externa de un producto

Definida por 6 características, cada una de las cuales contiene subcaracterísticas que se manifiestan cuando el software está en uso, y son consecuencia de ciertos atributos internos de software.

1. Funcionalidad	La capacidad del producto de software de proporcionar funcionalidades que satisfacen las necesidades establecidas e implícitas cuando es usado en condiciones especificadas.	
	Idóneo.	Capacidad de proporcionar un conjunto de funciones apropiado para las tareas y los objetivos de los usuarios.
	Exacto.	Capacidad de proporcionar resultados o efectos esperados o acordados con niveles de precisión que se necesiten.
	Interoperable.	Capacidad de interactuar con uno o más sistemas especificados.
	Seguro.	Capacidad de proteger información o datos de tal manera que las personas no autorizadas u otros sistemas no puedan leerlos ni modificar, y que a las personas autorizadas o sistemas no se les niegue el acceso.

Tabla 2.11

2. Confiabilidad	La capacidad del producto de software de mantener un nivel de desempeño especificado cuando se está usando en condiciones específicas.	
	Maduro.	Capacidad de evitar fallas que son

		resultado de defectos en software.
	Tolerante a fallos.	Capacidad de mantener un nivel especificado de desempeño en el caso de fallas de software o en el caso de violación de interfaces especificadas.
	Recobrable	Capacidad de reestablecer cierto grado especificado de desempeño y de recuperación de datos afectados directamente en el caso de una falla.

Tabla 2.12

3. Usabilidad	Capacidad de los productos de software de ser entendidos, aprendidos, usados y atractivos para el usuario, cuando son usados en un contexto especificado.	
	Entendible.	Capacidad de permitir al usuario entender si el software es apropiado, y como puede ser usado para tareas particulares y condiciones de uso.
	Asequible.	Capacidad de facilitar el aprender como usarlo.
	Operable.	Capacidad de facilitar la operación y el control por parte de usuario.
	Atractivo.	Capacidad de ser atractivo para el usuario.

Tabla 2.13

4. Eficiencia	Capacidad de un producto de software de proporcionar el desempeño apropiado en función de la totalidad de los

recursos utilizados, bajo condiciones establecidas.

Desempeño con respecto al tiempo.	Capacidad de proporcionar un tiempo de respuesta o de procesamiento apropiado dentro de ciertos rangos.
Utilización de recursos.	Capacidad de utilizar cantidades apropiadas de cada tipo de recurso durante la ejecución bajo las condiciones establecidas.

Tabla 2.14

5. Mantenibilidad — Capacidad de producto de software de ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones de software a cambios en el ambiente, requerimientos y especificaciones funcionales.

Analizable.	Capacidad para hacer diagnóstico con respecto a deficiencias o causas de fallas y de identificar partes que tienen que ser modificadas.
Cambiable.	Capacidad de implementar modificaciones especificadas.
Estable.	Capacidad de evitar efectos inesperados a causa de modificaciones.
Probable.	Capacidad de validar software modificado.

Tabla 2.15

6. Portabilidad — Capacidad de un producto de software de ser transferido de un ambiente a otro.

Adaptable.	Capacidad de poder adaptarse a diferentes ambientes especificados sin tener que realizar acciones otras de las que fueron consideradas para este propósito.
Instalable.	Capacidad de ser instalable en ambientes específicos.
Coexistente.	Capacidad de co-existir con otro software independiente en un ambiente común, compartiendo recursos.
Reemplazable.	Capacidad de ser utilizado en lugar de otro software para propósitos específicos en el mismo ambiente.

Tabla 2.16

Y adicionalmente, se incluye una característica más, que el autor de este documento considera que debe ser incluida como la séptima característica de un producto con calidad:

7. Reutilizabilidad	Capacidad de un producto de software de ser usado por futuros sistemas.	
	Escalable.	Capacidad de soportar mayores cargas de trabajo.
	Modular.	Capacidad de poder ser dividido en unidades independientes.
	Independiente.	Capacidad de poseer no dependencia con otras entidades.

Tabla 2.17

2.4.2 Calidad de uso

Definida por 4 características, que el usuario percibe como efecto combinado del cumplimiento de las 6 características de calidad interna y externa.

Característica	Definición
Efectividad	Capacidad de permitir que los usuarios logren objetivos específicos con exactitud y completitud en contexto específico de uso.
Productividad	Capacidad de permitir que los usuarios empleen cantidades apropiadas de recursos con respecto a la efectividad lograda en un contexto de uso específico.
Seguridad	Capacidad de lograr niveles aceptables de riesgo con respecto a los daños a personas, negocio, software, propiedad o ambiente en el contexto específico del uso.
Satisfacción	Capacidad de satisfacer a los usuarios en el contexto específico de uso.

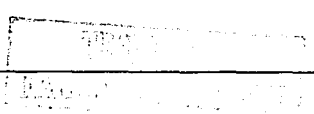
Tabla 2.18

2.5 Conclusiones del capítulo II

Existen diversas propuestas o metodologías para el proceso de desarrollo. En este capítulo se exponen tres de ellas: TSPi, RUP y XP. Cada una de éstas posee características que la hace aceptable o no para una organización. En este capítulo se han tomado los mejores aspectos de cada una de ellas y se ha propuesto la integración en una metodología resultante que se presentará a detalle en los siguientes capítulos. Cabe mencionar que la base para la integración fue TSPi y que adicionalmente, algunos artefactos también serán incorporados.

Dado que cada una de las metodologías expuestas describe la calidad de un producto de manera diferente, se propuso que, para describir la calidad en un producto, la base sería el ISO/IEC 9126-1.

La efectividad del proceso TSPi extendido fue probada exitosamente en un desarrollo real. Este desarrollo se reproduce y ejemplifica en los siguientes capítulos. El resultado fue la generación de un producto apegado principalmente a TSPi con la documentación, calidad y adecuada utilización de recursos materiales, técnicos y tecnológicos de los que se disponía en ese momento.



3

Hojas de recomendaciones técnicas

3.1 Introducción

Las hojas de recomendaciones técnicas (en lo sucesivo HRT's) son una propuesta propia y son una adición al conjunto de artefactos a generar en un proyecto. Éstas indican qué tipo de actividades técnicas deben ser consideradas o llevadas a cabo durante cierta fase del proceso de desarrollo por un cierto rol, roles o el equipo completo. Una HRT puede ser presentada en cualquier momento del desarrollo por cualquier miembro del equipo, para su evaluación y posterior aprobación para su ejecución.

3.2 Objetivo

Los objetivos de las HRT son los siguientes:

- Apoyar al plan de mitigación de riesgos generado en alguna fase, o mitigar riesgos que se identificaron posterior a dicho plan.

- Servir como base o referencia para la toma de decisiones en el desarrollo del sistema.
- Servir como referencia técnica en posteriores desarrollos, minimizando así tiempos y costos, reutilizando el conocimiento generado previamente.
- Incrementar la capacidad de desarrollo de la organización reduciendo la dependencia de recursos humanos especializados.

3.3 Descripción

Las hojas de recomendaciones técnicas son documentos que definen un marco formal de administración y seguimiento de propuestas que sugieren la realización de actividades que facilitarán, en el desarrollo, la toma de decisiones y/o la mitigación de riesgos identificados.

Cada HRT deberá incluir datos de carácter administrativo, como:

- Número de referencia (secuencial que define de manera única cada hoja de recomendaciones técnicas).
- Nombre del proyecto al que pertenece.
- Fase y ciclo en la que la recomendación es generada.
- Fecha de creación.
- Fecha límite para ejecución.
- Autor de la recomendación.
- Acrónimo del responsable de ejecución. (Será posible, de manera opcional, indicar el rol del responsable indicándolo después del acrónimo y separándolo con un símbolo ::)
- Aprobación (Conjunto de firmas de los integrantes del equipo que coinciden en consenso con la puesta en ejecución de la recomendación).

De manera similar, cada hoja presentará datos particulares como:

- Origen (razón que motivó la propuesta).
- Propuesta (la recomendación en sí misma). Ésta debe cumplir con los siguientes lineamientos:
  - ✓ Recomendar actividades no contempladas en los guiones de TSPi.

- ✓ Se debe evaluar el impacto positivo de la propuesta en el proyecto o en el proceso de desarrollo y concluir que éste es representativo.
- Justificación (conjunto de razones que apoyan la propuesta).
- Estrategia (forma en la que se propone llevar a cabo).
- Beneficios.
- Desventajas.
- Impacto en calidad (Característica de calidad –según el ISO/IEC 9126- que la propuesta promueve en el desarrollo)
- Tecnologías involucradas.

El siguiente es el formato que se propone para la elaboración de una HRT:

Referencia: *nnnn*	**Hoja de recomendaciones técnicas**
Proyecto:	*Nombre del proyecto*
Fase / Ciclo:	*Fase*, Ciclo *n*
Fecha de creación:	*dddd dd* de *mmmm* de *yyyy*
Fecha límite para ejecución:	*dddd dd* de *mmmm* de *yyyy* *Puede añadirse texto libre*
Autor:	*XYZ (Acrónimo del autor de la recomendación)*
Origen:	*Texto libre indicando porqué surge la recomendación*
Propuesta:	*Texto libre proponiendo una solución al problema origen*
Justificación:	*Conjunto de argumentos que sostienen la validez de la propuesta*
Estrategia:	*Plan de ejecución de la propuesta*
Beneficios:	*Lista de beneficios derivados de la ejecución de la propuesta*
Desventajas:	*Lista de desventajas derivadas de la ejecución de la propuesta*
Impacto en calidad:	*Lista de características de calidad (del ISO IEC 9126-1) que la propuesta promueve y una breve justificación.*
Tecnologías Involucradas:	*Lista de tecnologías requeridas (o involucradas) para llevar a cabo la propuesta*
Responsable de ejecución:	*XYZ (Acrónimo del responsable de la ejecución de la recomendación)*

Aprobación:	*Nombres y firmas de los involucrados que apoyan la propuesta*

Tabla 3.1

A continuación, se presenta un ejemplo del uso de este formato:

Referencia: 0071	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Factibilidad, Ciclo I
Fecha de creación:	Viernes 31 de Agosto de 2001
Fecha límite para ejecución:	No posterior al inicio de la fase de Lanzamiento
Autor:	GAA
Origen:	Existe un requerimiento explícito del cliente solicitando la implementación de cinco bases de datos para SIPU.
Propuesta:	Asumir que sólo habrá una sola base de datos que contendrá toda la información de los cinco programas universitarios y no cinco como originalmente el cliente solicitó.
Justificación:	- No existen reglas de negocio que sugieran que deba haber más de una base de datos. - Actualmente, no se posee la infraestructura para el ambiente de desarrollo con cinco bases de datos. - Las complejidades técnicas (en términos de desarrollo) se incrementan con varias bases de datos.
Estrategia:	Presentar al cliente un documento con argumentos que justifiquen el uso de una sola base de datos y que al mismo

	tiempo muestre que sus necesidades de esta manera quedan cubiertas.
Beneficios:	- Simplicidad en el desarrollo y la implementación. - Centralización de información. - Facilidad de administración, actualización y mantenimiento.
Desventajas:	- Se pierde la independencia del almacenamiento de la información - Se complica la definición del esquema de la base de datos
Impacto en calidad:	- Mantenible. A medida que la distribución de la información se concentra en menos almacenes persistentes, el mantenimiento de los mismos se minimiza. - Robusto. El control centralizado de información evita que existan diversos nodos para los cuales se verifica que el fallo en uno de ellos implica un fallo general en el sistema, ya que no son almacenes redundantes si no independientes.
Tecnologías Involucradas:	Ninguna.
Responsable de ejecución:	FNL (Administrador de Soporte)
Aprobación:	_____ _____ _____ _____ _____ _____

Tabla 3.2

3.4 Seguimiento

Con el fin de proveer un marco formal de administración y seguimiento de las distintas recomendaciones, se propone un formato de control que registrará lo necesario para tal efecto. Este formato será llamado en lo sucesivo forma RCSH (Registro para Control y Seguimiento de HRT's).

A continuación, se listan y explican los elementos que serán incluidos en dicho formato:

- ID. Identificador único de la HRT.
- IDR. Identificador del documento o documentos producto de la recomendación. En la siguiente sección se propone y describe un formato para el registro de éstos documentos resultantes.
- Fecha de creación. Fecha en la que se propuso la recomendación.
- Responsable de ejecución. Acrónimo y rol del recurso asignado a la ejecución de la tarea.
- Estatus. Uno de los siguientes indicadores:
  - ✓ Propuesta
  - ✓ Aprobada
  - ✓ Rechazada
  - ✓ En proceso
  - ✓ Terminada
- Observaciones. Datos adicionales de consideración en la propuesta o en la ejecución (o razón de rechazo) de la misma.

En la tabla 3.3 se muestra un ejemplo de dicho formato:

ID	IDR	Fecha de creación	Responsable de ejecución	Estatus	Observaciones
0001	007	12/Jun/2000	GAA::AD	Terminado	Ninguna
0002	009	21/Jun/2000	FNL::AC	En Proceso	N/A
0003	021	19/Nov/2000	APA	Propuesta	Analizándose su viabilidad
0004	N/A	17/Ago/2001	FNL::AQ	Rechazada	Ya incluida en TSPi

Tabla 3.3

3.5 Documentos resultantes

A cada propuesta técnica le corresponde uno o varios documentos resultantes que deben ser propiamente organizados y clasificados para su posterior recuperación y así lograr incrementar la madurez del proceso de producción de software. Adicionalmente, en éstos documentos queda registrado el cómo realizar ciertas actividades que podría ser necesario repetir en ocasiones futuras.

El formato de registro posee cinco atributos:

- ID. Identificador único que junto con el atributo 'parte' forma una llave.
- Parte. Consecutivo asociado a cada ID, útil para la identificación de sub documentos con un mismo ID.
- Fecha de generación. Fecha en la que se generó el documento.
- Ubicación de archivos. Ubicación física de los archivos resultantes de la ejecución y/o puesta en marcha de la recomendación.
- Reporte de resultados. Una descripción general del resultado final de la recomendación.

A continuación, en la tabla 3.4, se presenta un ejemplo de este formato:

ID	Parte	Fecha de generación	Ubicación de Archivos	Reporte de resultados
07	1	12/Sep/2001	C:\docs\x.doc	Se generaron seis documentos JSP obteniendo una eficaz interfaz de usuario
07	2	21/Sep/2001	/etc/rc/z.tar	Surgieron 11 clases mas de las previstas, en soporte a las propuestas
07	3	21/Sep/2001	%WORK%\w.txt	Se concluyó que sólo era necesaria la implementación de un único controlador

08	1	11/Oct/2001	../home/y.zip	Investigación de tecnologías WEB exitosa y fructífera
08	2	12/Oct/2001	./x.pdf	Implementación de tecnologías WEB candidatas concluidas con base en lo propuesto
09	1	11/Ene/2002	http://www.a.com	Propuesta de cambio alternativa al requerimiento explícito de implementación de cinco bases de datos aceptada.

Tabla 3.4

3.6 Herramienta para la administración de HRTs

Con el fin de llevar un control automatizado para la administración de las HRT's y adicionalmente generar una base de conocimiento con información acerca de cómo surgen y se resuelven ciertos aspectos técnicos, a continuación se presenta una herramienta - AHRTe por sus siglas Administrador de Hojas de Recomendaciones Técnicas - cuyo objetivo es apoyar y facilitar este tipo de actividades.

El administrador de HRT's o 'AHRTe', es una herramienta WEB de acceso autentificado que permitirá:

- definir proyectos y participantes
- asignar roles
- generar, asignar y reasignar HRT's
- dar seguimiento al desarrollo de las HRT's
- generar reportes y
- definir y ejecutar consultas sobre HRT's.

El AHRTe guarda y recupera información desde un repositorio central de datos y puede ser accedido desde cualquier máquina que tenga acceso a la red Internet.

Los clientes de ésta aplicación deben contar con un navegador (de preferencia el iExplorer 5.0 o una versión superior) y tener habilitadas las opciones *jScript* y *cookies*, para su correcto funcionamiento. La aplicación es independiente del sistema operativo y los requerimientos de memoria y disco duro son los mismos que se tienen para el navegador.

Existen tres tipos de usuarios del sistema los cuales se modelan en la figura 3.1. Éstos son especializaciones del "usuario genérico", que es abstracto y que por lo tanto nunca se instanciará.

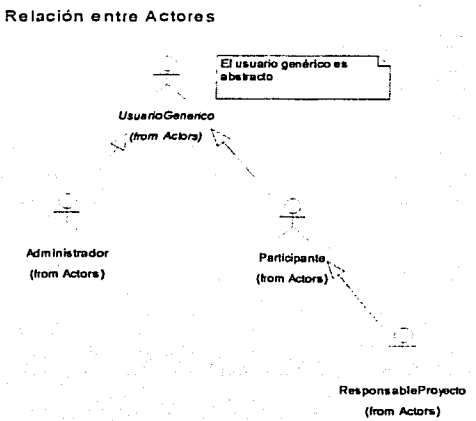


Figura 3.1

A continuación se presentan los casos de uso realizados por cada actor:

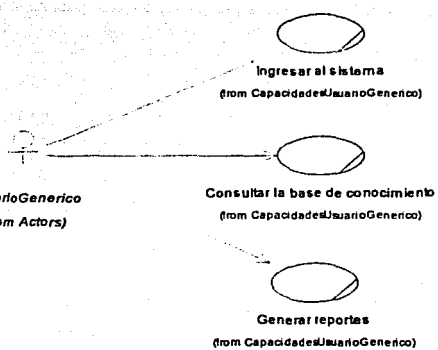


Figura 3.2

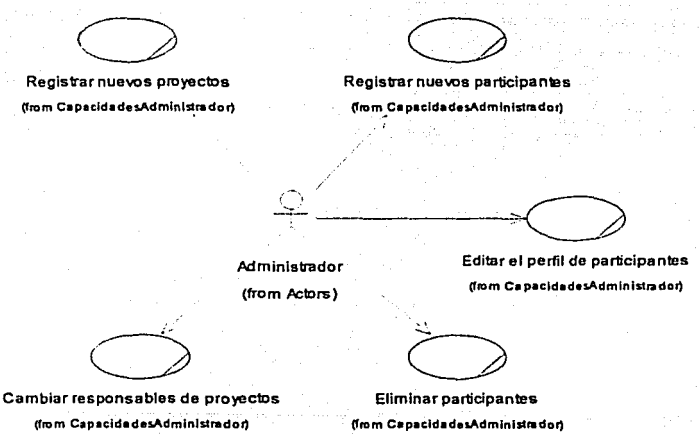


Figura 3.3

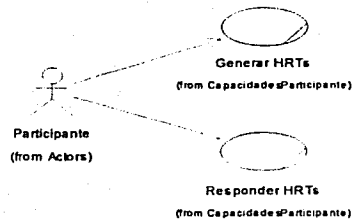


Figura 3.4

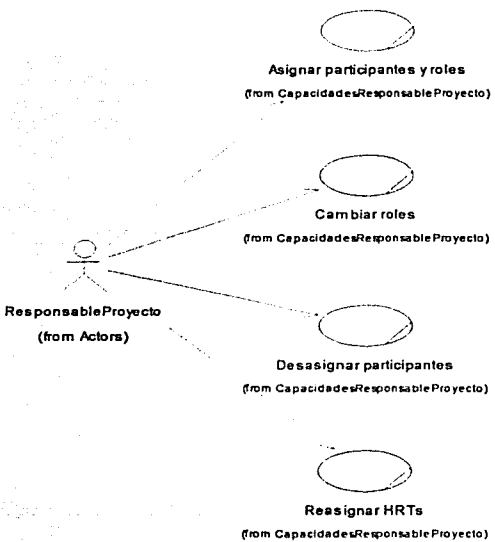


Figura 3.5

La secuencia para el uso del sistema queda descrita por los siguientes diagramas de actividades:

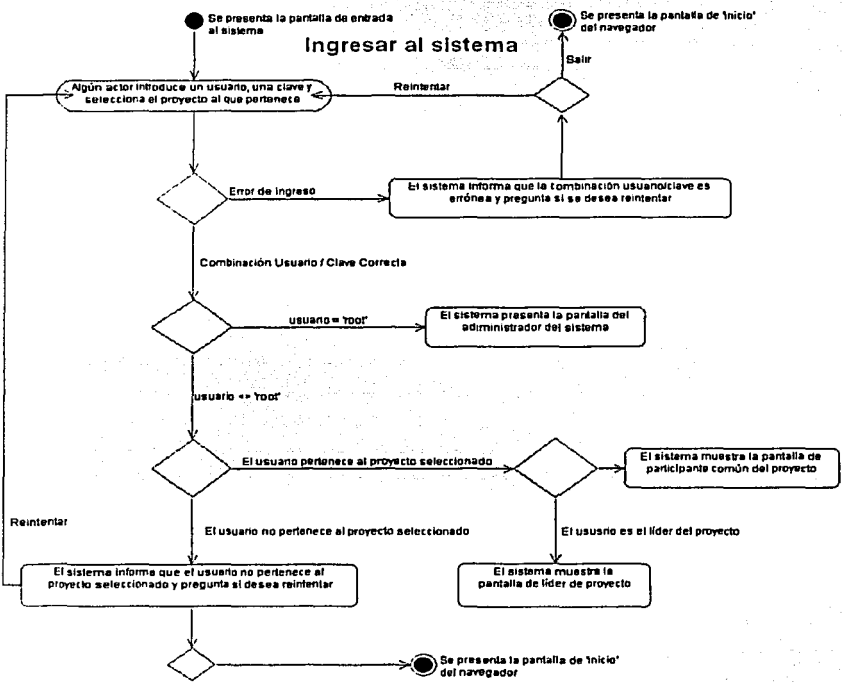


Figura 3.6

Los dos siguientes diagramas muestran secuencias de actividades típicas del sistema.

El primer diagrama representa la secuencia de registro de usuarios, creación de un proyecto, asignación del responsable del mismo y cómo éste integra personal a su equipo; finalmente, el diagrama muestra que el participante puede generar y responder HRT's.

El segundo representa la secuencia de desasignación de participantes de un proyecto.

Todos los actores pueden realizar funciones comunes descritas en los casos de usos del actor abstracto 'usuario genérico'.

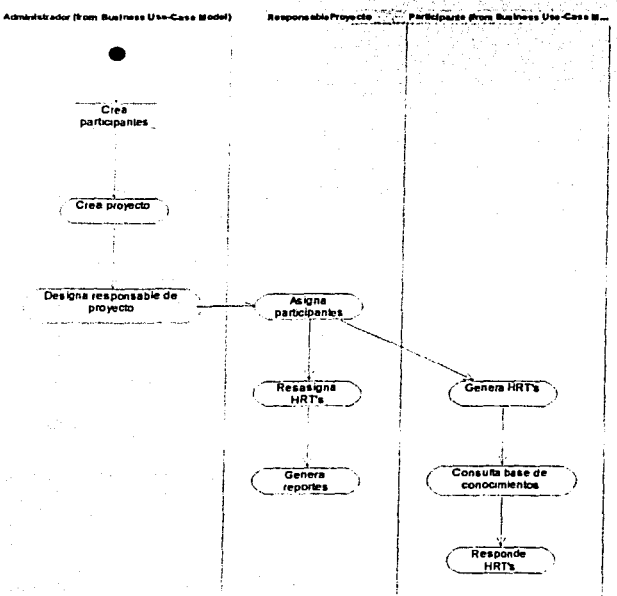


Figura 3.7

Un participante puede ser desasignado de un proyecto por el responsable del mismo, siempre y cuando dicho participante no tenga HRTs asignadas para responder.

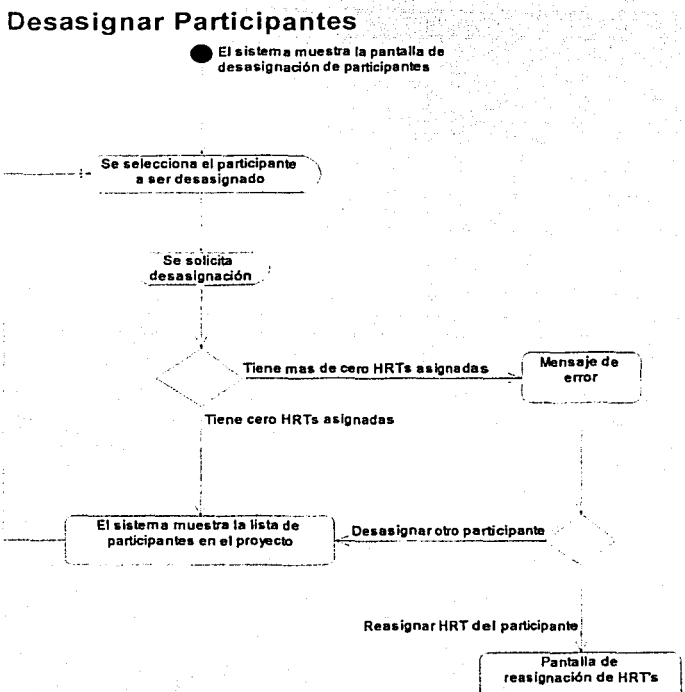


Figura 3.8

A continuación se presentan algunas de las pantallas asociadas a los diagramas anteriores:

Pantalla de entrada al sistema:

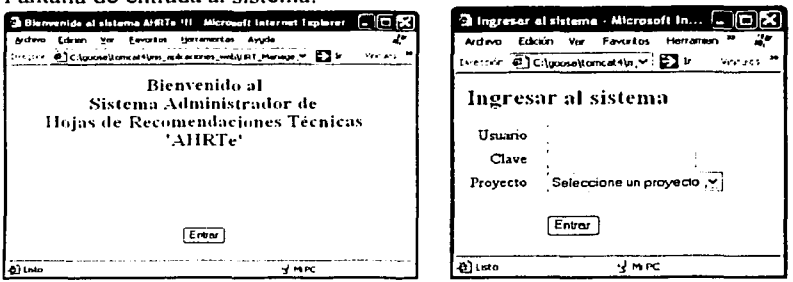


Figura 3.9

Pantalla principal del administrador del sistema

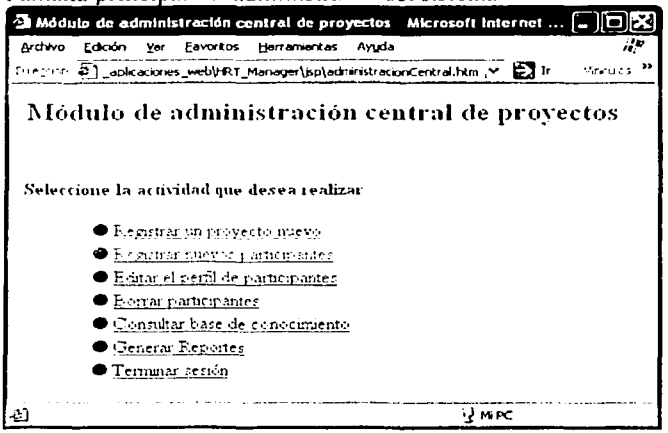


Figura 3.10

Pantalla de registro de participantes

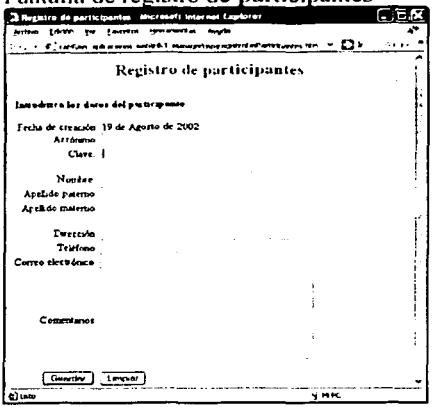


Figura 3.11

Pantalla de creación de proyectos

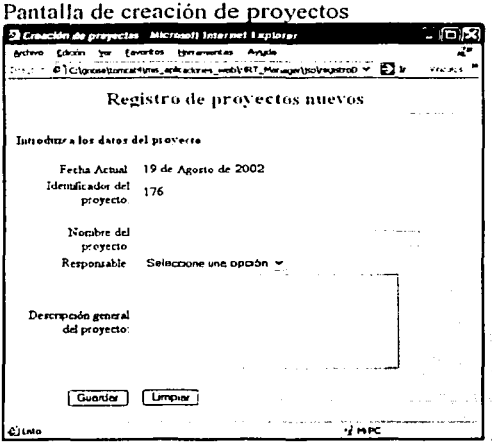


Fig 3.12

Pantalla de edición del perfil de participantes

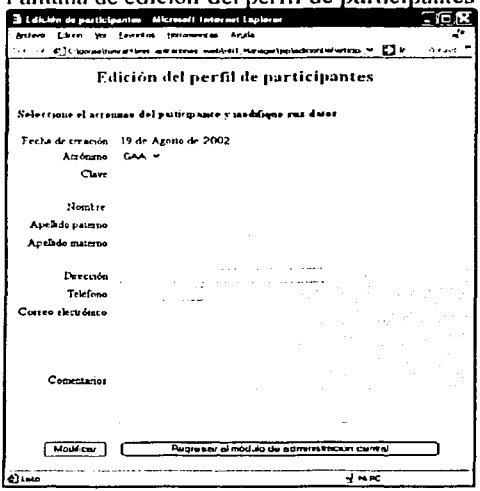


Figura 3.13

Pantalla de eliminación de participantes

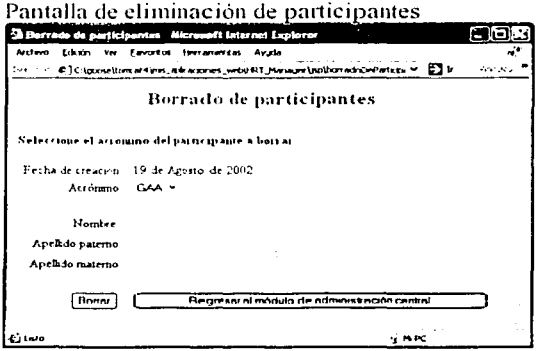


Figura 3.14

Pantalla de administración personal. En ésta pantalla si se es líder de proyecto, aparece un apartado especial para la administración del proyecto, tal y como se muestra en la siguiente figura:

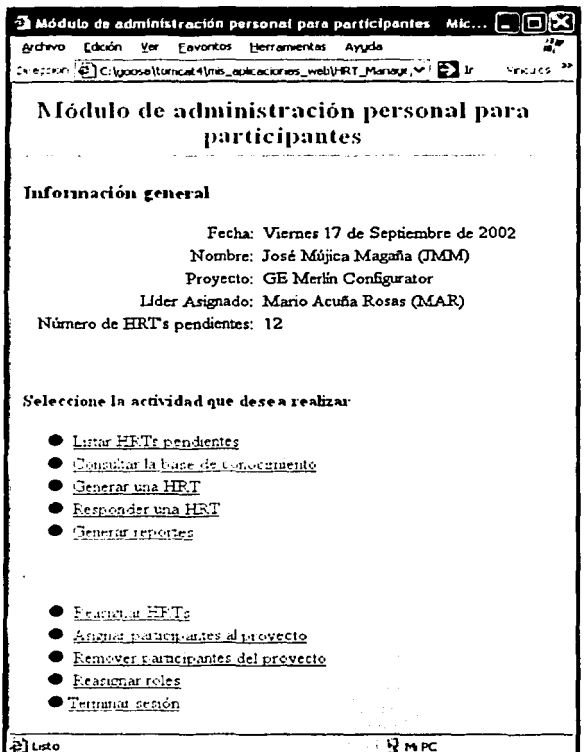


Figura 3.15

Pantalla del listado de HRTs pendientes de responder

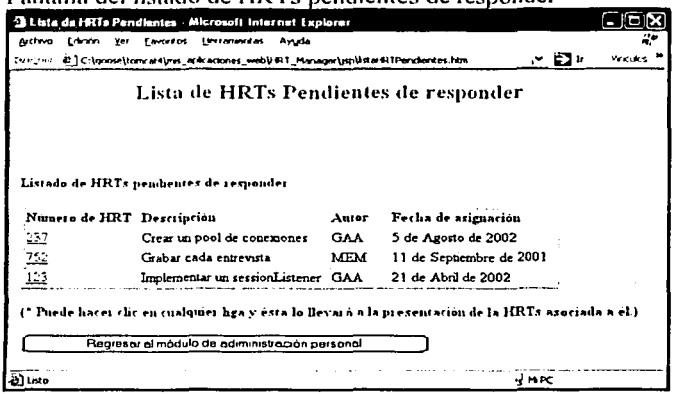


Figura 3.16

Pantalla de asignación de usuarios a proyectos

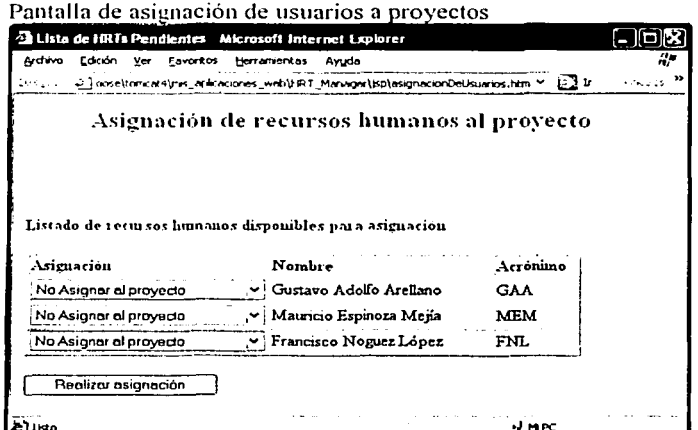


Figura 3.17

Pantalla para responder HRT's:

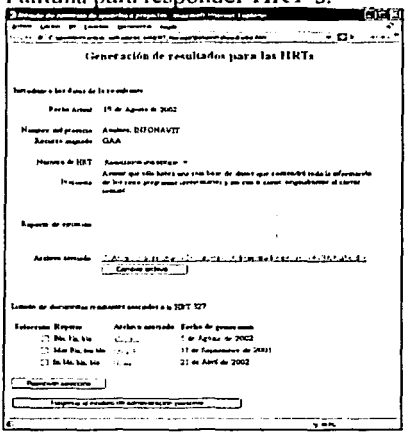


Figura 3.18

Pantalla para reasignación de HRT's

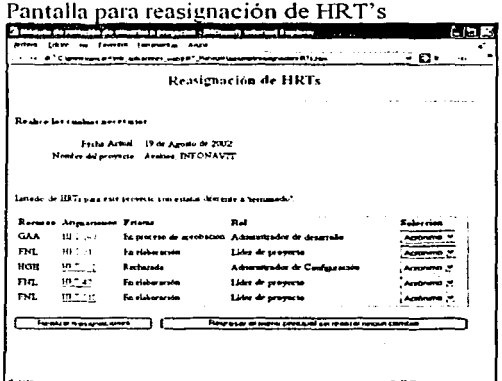


Figura 3.19

Pantalla de reasignación de roles

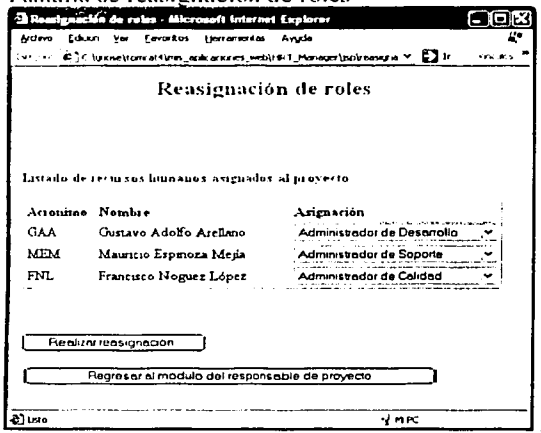


Figura 3.20

Remoción de usuarios del proyecto

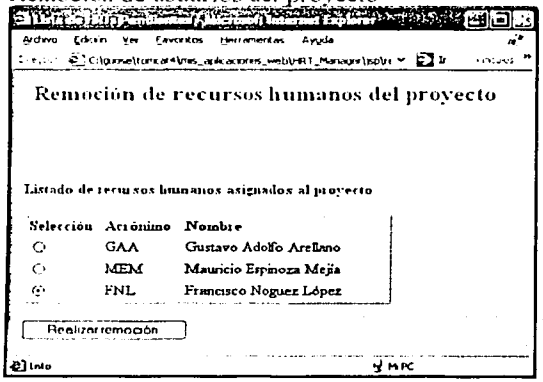


Figura 3.21

Pantalla de generación de recomendaciones:

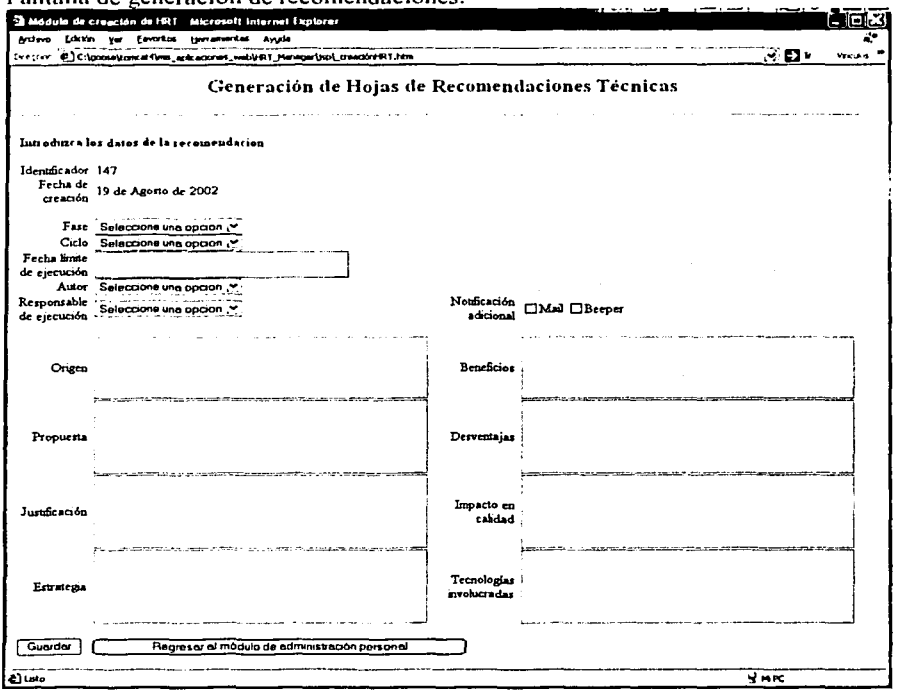


Figura 3.22

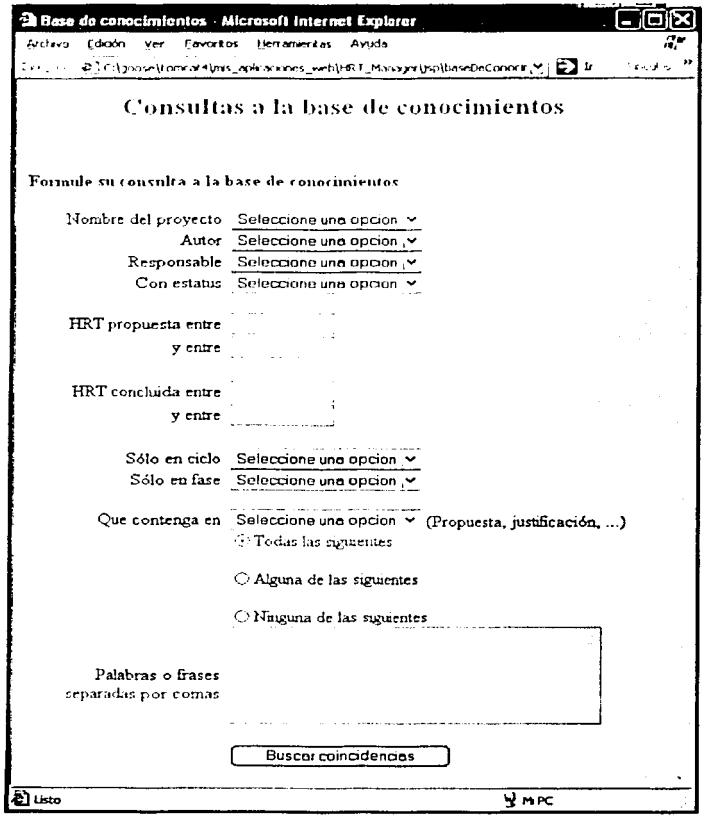

Base de conocimientos - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Consultas a la base de conocimientos
Formule su consulta a la base de conocimientos
Nombre del proyecto Seleccione una opcion
Autor Seleccione una opcion
Responsable Seleccione una opcion
Con estatus Seleccione una opcion
HRT propuesta entre
y entre
HRT concluida entre
y entre
Sólo en ciclo Seleccione una opcion
Sólo en fase Seleccione una opcion
Que contenga en Seleccione una opcion (Propuesta, justificación, ...)
Todas las siguientes
Alguna de las siguientes
Ninguna de las siguientes
Palabras o frases separadas por comas
Buscar coincidencias
Listo
Mi PC

Figura 3.23

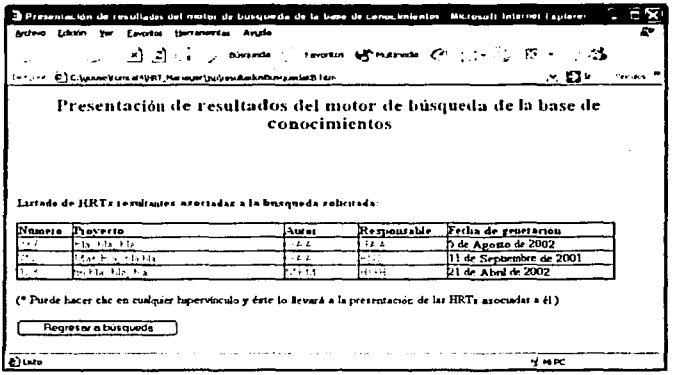


Figura 3.24

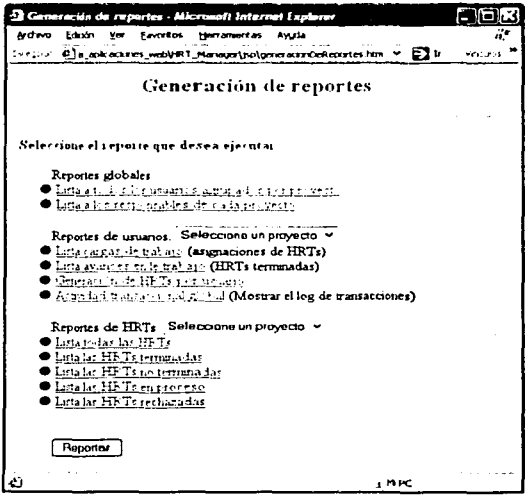


Figura 3.25

3.7 Conclusiones del capítulo III

Las hojas de recomendaciones técnicas ofrecen una forma administrada de control y seguimiento de propuestas técnicas que facilitarán la labor de los recursos involucrados en un desarrollo en el momento de mitigar riesgos o de tomar decisiones que afectarán al mismo.

Otro aspecto importante a considerar a favor de este proceso de adquisición de conocimientos es que reduce de manera importante la curva de aprendizaje e investigación por parte de los nuevos recursos humanos, que se vayan integrando a la organización, ya que le información técnica requerida, debidamente filtrada estará disponible en un formato estándar y común entre los recursos humanos de la organización.

Lo anterior repercute directamente en la calidad del producto, ya que este mecanismo permite analizar y en su caso, actualizar o mejorar una propuesta con el fin de generar, posteriormente, productos superiores.

Adicionalmente, este mecanismo es independiente de cualquier proceso de desarrollo y por ello es fácilmente integrable a la labor de desarrollo de cualquier organización.

Finalmente, conviene mencionar que existe una propuesta similar de eXtreme Programming llamada solución SPIKE que básicamente sugiere la implementación de diversas soluciones técnicas para un mismo problema difícil de atacar de lleno. Éstas se analizan y se toma la mas viable para la resolución del problema. eXtreme Programming no propone llevar un control administrado de las mismas y tampoco sugiere formatos específicos de propuesta, recabado, seguimiento o control. Adicionalmente, en eXtreme Programming (en algunas fuentes, XP) no hay manera de aprovechar el conocimiento generado a través de éstas soluciones, ya que una vez implementada, ésta no tiene forma de ser recuperada para su posterior reutilización.

4

Concepción

En este capítulo se presenta el flujo de concepción del caso de estudio descrito en este documento. Este flujo incluye las fases de factibilidad y de lanzamiento.

4.1 Fase de Factibilidad

La primer fase considerada en todo proceso de desarrollo de software es la de factibilidad, en donde se evalúan diversos aspectos técnicos y de negocio con base en los que se decidirá si el proyecto continuará o será cancelado, que es el objetivo principal de esta fase. En caso de que el desarrollo sea viable, se procede al lanzamiento formal de éste, en el que básicamente se define el ambiente general de desarrollo

Formalmente, factibilidad pertenece a RUP pero dada su importancia, se integró al marco principal de este trabajo, ya que TSPi no la considera. Es por

esto que la fase de factibilidad se verá con un poco mas de detalle que el resto de las fases que TSPi sí integra. Lo anterior es también debido a que dentro de los objetivos, actividades y artefactos definidos en esta fase, existen aspectos que involucran la toma de decisiones técnicas que impactarán de forma importante al resto del desarrollo del proyecto.

Los objetivos secundarios de esta fase son seis:

Número	Objetivo
1	Establecer el alcance del proyecto y los límites de éste, incluyendo una visión operacional, criterios de aceptación y qué es lo que se desea tener como producto final y qué no
2	Discriminar los casos de uso críticos para el sistema, los escenarios de operación primarios que van a definir los aspectos mayores del diseño
3	Exhibir al menos una arquitectura candidata para alguno de los escenarios primarios
4	Realizar un estimado general del tiempo y costo para el proyecto entero
5	Estimar riesgos potenciales
6	Preparar el ambiente general de desarrollo y soporte para el proyecto. Esto incluye, entre otras cosas, la formación del equipo de trabajo, asignación de roles y responsabilidades, adquisición e instalación del hardware y el software requerido, generación de formatos específicos para el proyecto y demás elementos que serán expuestos en su momento

Tabla 4.1

Los objetivos 1 y 4 no involucran la toma de decisiones técnicas o tecnológicas, sin embargo, la realización de las actividades asociadas a estos objetivos es primordial para la toma de decisiones acerca de la viabilidad del proyecto. En este punto, conviene mencionar que el equipo de desarrollo aun no está formado, y que por lo tanto la responsabilidad del cumplimiento de estos objetivos recae en el equipo de 'Factibilidad' formado por uno o dos recursos humanos. Si al concluir la fase se decide que el proyecto es viable, sería deseable que éstos formen parte de dicho equipo de desarrollo.

4.1.1 Actividades a realizar

Con base en los objetivos secundarios 2, 3, 5 y 6 y para que éstos se lleven a cabo, se definen cuatro actividades a realizar por el administrador de desarrollo. Esto es debido a que todas ellas involucran la toma de decisiones y consideraciones técnicas, propias de su rol.

Obj.	Actividad	Descripción
2	Identificar los casos de uso críticos del sistema y los escenarios de operación primarios	Se realiza un análisis del contexto, restricciones y requerimientos mas importantes del sistema. Con ello se obtienen los casos de uso mas importantes y se elaboran escenarios primarios con base en los que se definirá una arquitectura.
3	Sintetizar una arquitectura candidata.	Evaluar aspectos de diseño y componentes a hacer, comprar o reutilizar de manera que se pueda realizar un estimado de costo, calendario y recursos. Se deberá demostrar confiabilidad a través de algún tipo de prueba de concepto. Esto puede tomar la forma de un modelo que simula lo que es requerido o un prototipo inicial que explora lo que es considerado como área de alto riesgo. El prototipo en esta fase debe de generar la confianza de que una solución es posible, solución que será realizada en las fases de elaboración y construcción.
5	Elaborar lista de riesgos	Identificar fuentes de impredictibilidad que propicien incertidumbre o sean fuentes de posibles fallos del sistema o del desarrollo del mismo a futuro.
6	Preparar ambiente para el proyecto	Seleccionar recursos, herramientas y determinar procesos o habilidades dentro de la organización que se deben mejorar o adquirir.

Tabla 4.2

Respecto a los objetivos 1 y 4, las actividades propuestas para la realización de éstos son:

- Elaborar un documento que describa el alcance del proyecto
- Realizar un estimado general del tiempo y costo para el proyecto entero.

Estas actividades no son técnicas, sin embargo, vale la pena mencionarlas ya que son parte importante de los criterios de evaluación mencionados en la siguiente sección.

4.1.2 Criterios de evaluación de viabilidad de proyecto

Con base en los siguientes criterios, se decidirá si el proyecto debe o no continuar. Basta con que uno solo de estos puntos no se cumpla para que se justifique la cancelación del proyecto.

1. Acuerdo entre los involucrados en términos de costo, tiempo y alcance del proyecto.
2. Se ha hecho una primer captura de requerimientos y existe un entendimiento mutuo de que estos son los adecuados.
3. Aceptación de que las prioridades, estimados, riesgos y el proceso de desarrollo son adecuados.
4. La mayoría de los riesgos han sido identificados y se ha establecido una estrategia de mitigación para cada uno de ellos.

La evaluación de los criterios antes mencionados consiste básicamente en presentar ante los involucrados uno o varios documentos resultado de las actividades definidas en 4.1.1 y discutir acerca de si se llega a un acuerdo mutuo (o aceptación) de tales documentos.

4.1.3 Artefactos generados en la fase

La tabla 4.3 muestra una lista de los artefactos que son generados durante la fase de factibilidad y lanzamiento del proyecto. La segunda columna describe el estado del documento al final de la fase.

Artefacto	Estado
Visión	Los requerimientos centrales del proyecto así como las principales características y restricciones clave están documentadas.
Caso de negocio	Definido y aprobado.
Lista de riesgos	Riesgos generales iniciales identificados.
Plan de iteraciones	Número de iteraciones propuestas.
Plan de aceptación de producto	Revisado y en línea base.
Plantillas específicas para el proyecto	Terminadas.
Herramientas	Identificadas e instaladas.
Glosario	Revisado y en línea base.
Equipo de desarrollo	Formado. Roles asignados.

Tabla 4.3

Cada uno de los artefactos mencionados posee una descripción detallada (Ver documentación de RUP). Por el momento, nos concentraremos en los artefactos "Lista de riesgos" y "Herramientas".

4.1.4 Lista de riesgos

Como elemento fundamental en la toma de decisiones para definir la factibilidad de un proyecto, está el generar un documento que indique cuales son los riesgos identificados hasta el momento y elaborar un plan para su tratamiento. Los dos atributos que un riesgo puede tener son:

- Probabilidad de ocurrencia
- Impacto en el proyecto (severidad)

Existen tres posibles maneras de atacar riesgos:

Estrategia	Descripción
Evasión	Reorganizar el proyecto de manera que éste no pueda ser afectado por el riesgo.

Transferencia	Reorganizar el proyecto de manera que algo o alguien ajeno o externo al proyecto asuma el riesgo.
Aceptación	Decidir continuar con el riesgo como una contingencia; monitorear los síntomas del riesgo, mitigar su impacto y las posibilidades de ocurrencia y decidir con un plan de contingencia que hacer si el riesgo emerge.

Tabla 4.4

Los riesgos pueden ser clasificados básicamente en cuatro tipos, y cada tipo describe, a su vez, varios subtipos comentados a continuación:

1. Riesgos de recursos. Riesgos que involucran recursos humanos, materiales y adecuada organización administrativa y estratégica.

- Organización. Adecuado tamaño de la organización respecto al tamaño del desarrollo. Posesión de un proceso de desarrollo.
- Fondos. Suficientes fondos o adecuada administración de los mismos.
- Personal. Disponibilidad de recursos humanos con las habilidades y experiencia adecuadas. Capacidad de control de trabajo en equipo.
- Tiempo. Calendario realista, suficiencia de tiempo para el desarrollo, criticidad de fechas de entrega.

2. Riesgos técnicos. Riesgos que involucran el conocimiento, uso, alcance y dependencia de factores tecnológicos.

- Alcance. Inadecuada definición de la extensión y/o funcionalidad del desarrollo.
- Tecnológicos. Ausencia, falta o poco conocimiento de las tecnologías requeridas para el

		desarrollo.
	Dependencia externa.	Sujeción a las limitaciones o restricciones impuestas por entidades independientes al desarrollo.
3. Riesgos de negocio	Riesgos relacionados con la actual competencia en el mercado, honesta administración de la organización y sincronía de la variable valor – costo del proyecto.	
4. Riesgos de calendario	Riesgos relacionados con la posible extensión de la fecha de entrega del sistema o de alguna de sus partes.	

En la clasificación anterior podemos ver que los riesgos de organización y de negocio están fuera del alcance de un proceso de desarrollo. Sin embargo, los riesgos tecnológicos y los riesgos de calendario pueden ser tratados de una forma sistemática para su mitigación.

4.1.5 Generación de documentos para el caso de estudio **SIPU©**

El proyecto **SIPU©**, "*Sistema de Información para Programas Universitarios*", surge como un requerimiento de la *Coordinación de la Investigación Científica de la UNAM* ante la coordinación del área de ingeniería de software del posgrado en ciencia e ingeniería de la UNAM.

El proyecto SIPU contempla dos objetivos principales:

- Generar un producto de software que satisfaga las necesidades de quién lo solicita.
- Aplicar lo aprendido durante la maestría en el área de ingeniería de software.

Para lograr estos dos objetivos, se contaba con un grupo de doce estudiantes de maestría y del laboratorio para el área de ingeniería de software del posgrado en ciencia e ingeniería de la UNAM. Las herramientas de desarrollo fueron provistas por la compañía SIGA Desarrollos SA de CV.

Este trabajo incluirá actividades y artefactos que fueron generados en el proceso de desarrollo del proyecto **SIPU©**. La intención es mostrar cómo la integración, vía las hojas de recomendaciones técnicas, del contexto técnico y tecnológico al proceso de desarrollo promueve la generación de software con calidad, y cómo estas recomendaciones (y sus documentos resultantes) podrían ser reutilizados en otros desarrollos.

A continuación, se presenta el enunciado general del problema, que es la principal parte del documento de la *visión* del proyecto.

"*... La Coordinación de la Investigación Científica de la UNAM solicita la elaboración de un sistema de consulta para programas universitarios a través de Internet, debiendo ser éste, un sistema poderoso y al mismo tiempo, un versátil motor de búsqueda de información. También se requiere la elaboración de un sistema de captura de información concerniente a los proyectos, investigadores, dependencias, equipos, publicaciones y líneas de investigación para ser explotada por el mencionado motor de búsqueda en Internet. Finalmente, se requiere de un módulo de administración central que otorgue y revoque permisos de captura de información al sistema. ...* ".

Y añade:

"*... Actualmente existen cinco programas universitarios y la información debe estar organizada por programa universitario, por lo que se solicita se generen cinco bases de datos; una para cada programa universitario. Nosotros tenemos Oracle 8i...*"

Dado lo anterior, se generó la siguiente hoja de identificación de riesgos:

Hoja de Riesgos Técnicos Identificados.
Proyecto: SIPU
Fecha: Martes 21 de Agosto de 2001
Autor: GAA

Riesgos de alcance	1.1	Actualmente el alcance es demasiado grande.
	1.2	El alcance que se supone puede variar significativamente del que el cliente espera.
	1.3	Los requerimientos aún no están completamente estables y hay algunos que no están claros.
Riesgos Tecnológicos	2.1	No se conoce la suficiente tecnología WEB.
	2.2	No se tienen componentes para reuso.
	2.3	No conocemos el volumen transaccional exacto del aplicativo.
	2.4	Disponibilidad de un servidor de bases de datos ORACLE (y aprendizaje del mismo sobre SOLARIS 2.7)
	2.5	Dado que el desarrollo debe ser llevado a cabo con tecnología Java, concluimos que existe alta complejidad en el aprendizaje de la tecnología JSP.
Riesgos de dependencia externa	3.1	Éxito codependiente del desempeño de los otros grupos de desarrollo.
	3.2	Disponibilidad de equipo y herramientas de desarrollo.

A cada uno de los riesgos identificados le corresponde una estrategia de mitigación. En términos generales, tal estrategia toma la forma de una breve propuesta y una técnica de implementación de la misma. Adicionalmente, una estrategia de mitigación, puede incluir documentos anexos (como una hoja de recomendaciones técnicas) que complemente la técnica de implementación asociada a una propuesta. A continuación, se presenta la estrategia de mitigación de riesgos y posteriormente, una tabla que justifica los aspectos de calidad asociados a cada propuesta.

Estrategia de mitigación de riesgos		
Riesgo	**Propuesta**	**Estrategia**

1.1	Se propone dividir el alcance.	Formar tres equipos de desarrollo con cuatro personas cada uno y repartir el trabajo en tres porciones.
1.2	Riesgo asumido.	N/A
1.3	Se propone realizar mas entrevistas con el cliente y asumir ciertos requerimientos preliminares.	Con base en la información proveída por el cliente, se decidió que el sistema utilizará sólo una base de datos.

2.1	Obtener fuentes de información, recursos, documentación asociada y practicar la tecnología.	Realizar micro programas de prueba.
2.2	Crear componentes básicos generales para su posterior utilización.	Definir y generar componentes y construir micro programas que los usen y los prueben.
2.3	Suponer un volumen alto de transacciones.	Construir componentes de soporte al alto desempeño y volumen transaccional.
2.4	Desarrollar para un manejador ya conocido, disponible e instalable localmente y posteriormente, solicitar el acceso un servidor SUN con ORACLE.	Codificar para Access y posteriormente desarrollar una clase (patrón 'pipe') para traducir las cadenas SQL de Access a las de ORACLE.
2.5	Obtener capacitación en cada tópico.	Obtener y realizar los tutoriales de SUN.

3.1	Procurar mínima interacción entre equipos.	Dividir los casos de uso en particiones del proyecto.
3.2	Obtención y aprendizaje de las diversas herramientas.	Realizar pequeñas prácticas de adiestramiento.

A continuación se presenta una serie de HRT's que contribuirán al soporte de

la estrategia para la mitigación de riesgos:

Referencia: 0001	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	*Riesgo 1.1*
Propuesta:	*Dividir el alcance del proyecto.*
Justificación:	*No es posible desarrollar el sistema sin una clara separación y asignación de actividades, dada su gran extensión y complejidad.*
Estrategia:	*Se propone la división: Módulo de consulta, Módulo de captura y Módulo de administración central. Un módulo para cada equipo, asignado al azar.*
Beneficios:	*Distribución equitativa del trabajo a realizar, mínima interacción entre equipos.*
Desventajas:	*Cada equipo sólo conoce lo que él mismo desarrolla y no lo que otros están haciendo.*
Impacto en calidad:	*Modularidad, Mantenibilidad. El desarrollo por separado promueve que los equipos desarrollen modularmente y en consecuencia, el mantenimiento es mayor. Adicionalmente, el desarrollo de módulos independientes promueve atomicidad de componentes, lo que conlleva a obtener una mayor mantenibilidad, ya que las posteriores modificaciones a éstos módulos independientes no propagarán cambios a los demás subsistemas.*
Tecnologías Involucradas:	*NA*
Responsable de ejecución:	*Grupo de desarrollo completo*
Aprobación:	(Votado y aprobado por todo el grupo)

Dada la recomendación anterior, el grupo de desarrollo fue dividido en tres equipos, cada uno con cuatro personas y las asignaciones de trabajo fueron las siguientes:

ID	Nombre del equipo	Asignación
1	METASOFT	Desarrollo del sistema de consulta en WEB
2	GRAD	Desarrollo del sistema de captura en RMI
3	DISACOMM	Desarrollo del sistema de administración y diseño de bases de datos.

Tabla 4.5

En lo sucesivo, y debido a la extensión del desarrollo completo, el presente documento se enfocará únicamente en el desarrollo del subsistema de consulta en WEB, elaborado por METASOFT.

Con base en lo anterior, ahora se presentan una serie de hojas de recomendaciones técnicas que influyeron en el subsistema de consulta.

Referencia: *0002*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	*Un requerimiento explícito del cliente solicitando la implementación de cinco bases de datos para SIPU.*
Propuesta:	*Proponer un cambio al requerimiento.*
Justificación:	*No existen reglas de negocio que sugieran que deba haber más de una base de datos.*
Estrategia:	*Implementar para una sola base de datos presentando al cliente un documento con argumentos que justifiquen el uso de una sola base de datos y que al mismo tiempo*

	muestre que sus necesidades de esta manera también quedan cubiertas.
Beneficios:	- *Simplicidad en el desarrollo y la implementación.* - *Centralización de información.* - *Facilidad de administración, actualización y mantenimiento.*
Desventajas:	*Incremento en la complejidad del desarrollo de consultas a la base de datos.*
Impacto en calidad:	*Mantenibilidad. El mantenimiento que requiere una base de datos es menor al que requieren varias de ellas.*
Tecnologías Involucradas:	*N/A*
Responsable de ejecución:	*Grupo de desarrollo completo*
Aprobación:	(Votado y aprobado por todo el grupo)

Aunque la recomendación anterior no está relacionada con el proceso de desarrollo en WEB de una aplicación, ésta es fundamental para el resto del sistema y del mismo desarrollo. Es por esto que fue importante incluirla en este conjunto de propuestas. Las siguientes recomendaciones están más dirigidas a la parte de desarrollo en WEB.

Referencia: *0003*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad,* Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	*El Servidor de WEB JSP que se utilizará será Tomcat4.*
Propuesta:	*Documentar secuencia de instalación de Tomcat4 (Tanto para Win32 como para LINUX).*
Justificación:	- *Tomcat4 integra en nuevo instalador para Win32, pero no para LINUX.*

	- *Tomcat4 integra nuevos servicios necesarios para SIPU.* - *Debido a que el sistema será desarrollado en sistemas Win32, pero probado e instalado en plataformas UNIX-like, ya para los primeros deployments será necesario dominar la técnica y uso del mismo.*
Estrategia:	*Obtener documentación y realizar pruebas de instalación en máquinas a las que no les afecte el que dicha instalación haya sido errónea.*
Beneficios:	- *Obtención de documentación para usuario y próximas instalaciones* - *Pronta disponibilidad del servidor de WEB/JSP.*
Desventajas	*Tomcat requiere de Apache para operar en producción*
Impacto en calidad	*Robustez, confiabilidad, seguridad. Se ha seleccionado el servidor de JSP's más reconocido en el mercado. Adicionalmente, éste incorpora varios servicios de seguridad y autenticación de usuarios.*
T. Involucradas:	*Servidores de Servlets y JSP's.*
Responsable de ejecución:	*GAA*
Aprobación:	

Referencia: *0004*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	Se trabajará en una ambiente de desarrollo y periódicamente se realizarán *deployments* en el ambiente de pruebas.

Propuesta:	Investigar secuencia de creación de archivos de instalación .war y proceso de *deployment* en el ambiente de pruebas.
Justificación:	- Será un proceso repetitivo y común. - Se estima que los primeros *deployments* (producto del desarrollo) serán realizados en la siguiente fase.
Estrategia:	Obtener documentación y realizar pruebas de creación e instalación de archivos de *deployment* en el ambiente de pruebas.
Beneficios:	- Obtención de documentación para usuario y próximas actualizaciones.
Desventajas:	Se invierte en investigación tiempo asignado al proyecto.
Impacto en calidad:	Funcionalidad, confiablilidad. La práctica en los distintos recursos tecnológicos promueve el desarrollo de eficientes implementaciones posteriores de las funcionalidades del sistema requeridas por el cliente.
T. Involucradas:	Servidores de Servlets y JSP's.
Responsable de ejecución:	GAA::AD
Aprobación:	

Referencia: *0005*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	Se estima que más del 70% de archivos generados serán del tipo JSP.
Propuesta:	Iniciar con conceptos básicos de JSP's.
Justificación:	- Se plantea realizar esto para tener un primer acercamiento al desarrollo de JSP's.
Estrategia:	Creación de un primer JSP lo mas simple posible.

Beneficios:	- Anticipación a la solución de posibles problemas técnicos.
Desventajas:	Se invierte en investigación tiempo asignado al proyecto.
Impacto en calidad:	Funcionalidad, confiablilidad. La práctica en los distintos recursos tecnológicos promueve el desarrollo de eficientes implementaciones posteriores de las funcionalidades del sistema requeridas por el cliente.
T. Involucradas:	JSP's.
Responsable de ejecución:	GAA::AD
Aprobación:	

Referencia: *0006*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	En SIPU, la comunicación entre JSP's será común.
Propuesta:	Ejemplificar cómo este proceso se puede llevar a cabo.
Justificación:	- Una de las principales funcionalidades de SIPU es solicitar información, realizar consultas con tal información y presentar resultados asociados. Para tal efecto será básico poder interaccionar entre objetos de tipo JSP, sin embargo esto no se realiza como de costumbre, vía instancias y mensajes. Por lo que la técnica (o técnicas) para realizar esto debe ser adquirida y dominada.
Estrategia:	Hacer un micro desarrollo con envío y recepción de información entre dos JSP's.
Beneficios:	- Obtención de código a implementar en el futuro.
Desventajas:	Se invierte en investigación tiempo asignado al proyecto.

Impacto en calidad:	Funcionalidad, confiablilidad. La práctica en los distintos recursos tecnológicos promueve el desarrollo de eficientes implementaciones posteriores de las funcionalidades del sistema requeridas por el cliente.
T. Involucradas:	JSP's.
Responsable de ejecución:	GAA::AD
Aprobación:	

Referencia: *0007*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	Los JSP's son los únicos objetos que desplegarán información obtenida de la base de datos.
Propuesta:	Ofrecer un ejemplo de desplegado de datos vía un JSP.
Justificación:	- Existe la necesidad de generar de manera dinámica páginas de resultados que provienen de una base de datos. Prácticamente el 75% del sistema genera contenido dinámico y el otro 25% es la parte estática en la que se generan las consultas. Habrá casos en que el proceso generador de consultas sea, a su vez, generado dinámicamente.
Estrategia:	Creación de un JSP que presente datos extraídos de una base de datos.
Beneficios:	- Obtención del dominio de la técnica adecuada. - Anticipación a una actividad que se prevé ocurrirá regularmente.
Desventajas:	Se invierte en investigación tiempo asignado al proyecto.
Impacto en	Funcionalidad, Confiablilidad. La práctica en los

calidad:	distintos recursos tecnológicos promueve el desarrollo de eficientes implementaciones posteriores de las funcionalidades del sistema requeridas por el cliente.
T. Involucradas:	JDBC, JSP.
Responsable de ejecución:	GAA:AD
Aprobación:	

Referencia: *0008*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Factibilidad*, Ciclo *1*
Fecha de creación:	*Viernes 31* de *Agosto* de *2001*
Fecha límite para ejecución:	*Viernes 31* de *Agosto* de *2001*
Autor:	*GAA*
Origen:	El sistema será accedido por un gran número de usuarios que a su vez, realizarán varias consultas a la base de datos.
Propuesta:	Implementar un *pool* de conexiones.
Justificación:	- En un sistema de información vía WEB no es posible asignar conexiones del servidor de bases de datos a cada consultante y menos aun mantener asignada al consultante tal conexión por tiempo indefinido. Es por ello que debe haber una clase que administre adecuadamente un número fijo de conexiones reciclando su uso.
Estrategia:	Creación de una clase de administración de conexiones al servidor de bases de datos y también un micro sistema que pruebe su uso.
Beneficios:	- Un volumen alto de peticiones de consultas puede ser atendido sin tener que llegar a pasar los límites de capacidad del DBMS. - Consumo bajo en recursos y tiempos de acceso a la información.

	- Mayor control en términos de administración de conexiones. - Facilidad en la integración de mecanismos de seguridad, en conexiones al servidor de bases de datos.
Desventajas:	La complejidad de desarrollo del componente de conexión a bases de datos es alta.
Impacto en calidad:	Robustez. Portabilidad. Con un *pool* de conexiones se podrá dar atención a un mayor número de usuarios simultáneamente, sin experimentar fallos en el sistema o detrimento en su desempeño. Adicionalmente, la clase poolConnection podrá soportar ser conectada a diversos manejadores, debido a su independencia con el resto del programa.
T. Involucradas:	JDBC
Responsable de ejecución:	GAA::AD
Aprobación:	

A continuación, se presenta el registro para el control de HRT's, que se definió en el capítulo anterior llamado "Registro para Control y Seguimiento de HRT's" y que en lo sucesivo se le denominará RCSH:

ID HRT	IDR	Fecha de creación	Responsable de ejecución	Estatus	Observaciones
0001	N/A	12/Sep/2001	GAA::AD	Terminado	Fue un acuerdo grupal
0002	001	20/Sep/2001	GAA::AD	Terminado	Consenso grupal
0003	002	7/Oct/2001	GAA::AD	Terminado	N/A
0004	003	7/Oct/2001	GAA::AD	Terminado	N/A
0005	004	19/Oct/2001	GAA::AD	Terminado	N/A
0006	005	21/Oct/2001	GAA::AD	Terminado	N/A
0007	006	30/Oct/2001	GAA::AD	Terminado	N/A
0008	007	1/Dic/2001	GAA::AD	Terminado	N/A

Tabla 4.6

Recordemos que a cada hoja de recomendaciones le corresponde un documento que detalle (en caso de que aplique) a nivel de código fuente lo

solicitado, y en muchos casos será un micro desarrollo completamente funcional y auto contenido que pueda ser probado de manera fácil y que además cumpla toda la funcionalidad que se solicitó en dicha hoja.

El registro para el control de documentos resultantes fue definido en el capítulo anterior, y a continuación es usado para documentar esto último:

ID	Parte	Fecha de generación	Reporte de resultados
0002	1	20/Sep/2001	Documento explicatorio de la propuesta de 1 versus 5 bases de datos
0003	1	7/Oct/2001	Texto libre indicando secuencia de instalación de Tomcat 4
0004	1	7/Oct/2001	Instrucciones de instalación de ambiente de desarrollo
0005	1	19/Oct/2001	Micro aplicación ejemplificando el uso de un JSP's
0006	1	21/Oct/2001	Micro aplicación ejemplificando el intercambio de información entre JSP's
0007	1	30/Oct/2001	Micro aplicación con JSP's presentando información extraída de una base de datos.
0007	1	1/Dic/2001	Creación de una clase de pool de conexiones 'poolConnection'

Tabla 4.7

Al final de cada fase, los registros de control de HRT's y de documentos resultantes serán presentados con el fin de resumir y consolidar la información expuesta en cada HRT.

4.1.6 Herramientas

En un proceso de desarrollo de software, es común requerir de herramientas para soportar las actividades en el ciclo de vida del mismo. Adicionalmente, el un proceso de desarrollo iterativo, debe procurar mantener una alta integración entre las herramientas que en él se empleen, para lograr así obtener interoperabilidad entre modelo y código. Existe la necesidad de automatizar el proceso de pruebas y documentación; de facilitar el modelado y de ser posible, automatizar también la administración del proyecto.

Dentro de las recomendaciones técnicas para la adquisición de herramientas de apoyo las siguientes fueron sugeridas:

Actividad	Herramienta(s) recomendada(s)
Administración de requerimientos	Documentar con un editor de texto
Modelado visual	Rational Rose 2000
Codificación	Jbuilder 5, JSDK 1.3
Automatización de pruebas	Rational Robot, Rational TestFactory, Rational Purify, Rational Pure Coverage, Rational Quantify, Rational TestManager
Administración de configuración	Source Safe 5.0 Winzip 8.0
Administración de cambios	Documentar con un editor de texto
Administración de proyecto	Project 98 Outlook 2000
Documentación	Word 2000 + Access 2000
Web Authoring	Front Page 2000, Web Server
Modelado de Datos	ERWIN 2.0 + Access 2000

Tabla 4.8

La adecuada selección de herramientas garantiza la minimización del esfuerzo que representa realizar las actividades mencionadas sin su soporte. Cabe

mencionar que para el proyecto **SIPU©** no fue posible obtener las herramientas recomendadas para las actividades de automatización de pruebas.

4.2 Fase de Lanzamiento

La fase de lanzamiento – la primera de TSPi, como se describió en el capítulo II – incluye varios de los aspectos mencionados en la fase de factibilidad, sin embargo, para este momento ya se decidió continuar con el proyecto y todos los criterios de factibilidad a evaluar ya fueron revisados y aprobados.

Las actividades principales que quedan a realizar en esta fase son:

- Formar el equipo y asignar roles
- Definir objetivos del equipo
- Definir objetivos del producto (única actividad del administrador de desarrollo)
- Definir objetivos de cada miembro / rol
- Definir agenda para la primera reunión
- Asistir a la primera reunión semanal
- Definir estándares de documentación
- Diseñar lista de verificación de los documentos definidos

En esta fase, la definición de los objetivos del producto – según TSPi - es la única actividad del administrador de desarrollo, sin embargo, en esta fase se deberá obtener, estudiar y ensayar todas aquellas técnicas aun no adquiridas o dominadas y que serán básicas o inclusive indispensables para poder desarrollar el producto solicitado. Éstas actividades ya fueron, en parte, definidas a través de las HRT's precedentes.

4.2.1 Formación del equipo y asignación de roles

El equipo METASOFT se formó de la siguiente manera:

Rol	Recurso Humano
LP	Ing. Mauricio Mejía

AD	Mat. Gustavo Arellano
AQ	M. en I. Francisco Noguez
AP	Lic. Ana Pérez Arteaga
AS y AC	METASOFT (El equipo completo)

4.2.2 Objetivos del equipo

Los objetivos del equipo METASOFT son los siguientes:

- Generar un producto de calidad y en tiempo.
- Distribuir equitativamente el trabajo a elaborar.
- Generar componentes y una base de conocimientos reutilizables.

4.2.3 Objetivos del Producto para el caso de estudio

Con base en el enunciado del problema (presentado en la sección 4.1) y en la decisión que surge de la HRT 001, se pueden escribir formalmente los antecedentes y el objetivo del desarrollo para el equipo de trabajo en el subsistema WEB:

Antecedentes.

Dado que no se cuenta con un sistema de administración y consulta de información perteneciente a proyectos de investigación y recursos humanos y materiales relacionados con cada uno de ellos, así como los temas que cada uno de éstos aborda, se propone la creación del sistema de Información para Programas Universitarios (**SIPU**©).

Objetivo.

- Desarrollar el sistema **SIPU**© dividido en tres módulos independientes.
- Para cada módulo, habrá un equipo encargado de su desarrollo.

- El equipo METASOFT será responsable de desarrollar el módulo de consultas, el cual tendrá las siguientes características:

  - ✓ Acceso por Internet vía un navegador (como mínimo, IE 5.5 y Netscape 6.0)
  - ✓ Identificación de tipo de usuarios con el fin de obtener estadísticas de uso e información consultada.
  - ✓ Consulta de información sobre investigadores, líneas de investigación, proyectos de investigación, equipos, publicaciones y temas varios.

4.2.4 Actividades restantes de lanzamiento

Las actividades que restan por documentar son:

i. Definir objetivos de cada miembro / rol.
ii. Definir agenda para la primera reunión.
iii. Asistir a la primera reunión semanal.
iv. Definir estándares de documentación.
v. Diseñar lista de verificación de los documentos definidos.

Los documentos resultantes de estas actividades no serán presentados en este trabajo, ya que no involucran la toma de decisiones técnicas o tecnológicas con impacto a futuro en el desarrollo.

4.2.5 Actividades derivadas de las HRT's

Con base en las hojas de recomendaciones técnicas presentadas en la fase anterior se realizaron las actividades correspondientes y en el disco compacto que se incluye, se podrán consultar los documentos producto de tales actividades.

4.3 Conclusiones del capítulo IV

Este capítulo abarca las dos primeras fases del proceso de desarrollo. En la primera de ellas, la de factibilidad, se toma la decisión de continuar o cancelar el proyecto y la intervención de personal técnico es básica para la toma de tales decisiones. En ésta fase se proponen actividades a realizar con el fin de mitigar los riesgos detectados y algunas de ellas deben ser llevadas a cabo en la siguiente fase, que es la de lanzamiento con fin de anticipar actividades que serán comunes o críticas en próximas fases.

En la fase de lanzamiento, el trabajo técnico – según TSPi – es mínimo y es por ello que se propone en esta fase realizar actividades cuyos productos servirán de apoyo en futuras fases del desarrollo. Estas actividades fueron propuestas en la fase anterior vía hojas de recomendaciones técnicas las cuales son susceptibles de aparecer en cualquier fase del desarrollo del proyecto.

Cada propuesta lleva la intención de impactar positivamente en uno o varios aspectos de calidad (según están definidos en el ISO/IEC 9126-1) del producto y en cada propuesta existe una justificación del porque ése aspecto de calidad es mencionado en ella.

Finalmente, se lleva un registro de los documentos resultantes de cada propuesta, con el fin de reutilizar ese conocimiento en desarrollos posteriores y adquirir mayor madurez en el desarrollo de un producto de software.

5

Elaboración

El flujo de trabajo de elaboración, que es el segundo de los cuatro que el presente proceso de desarrollo contempla, incluye las fases de estrategia, planeación y requerimientos.

5.1 Introducción

En el flujo anterior, se definió la viabilidad del proyecto y en términos generales, el tipo de actividades a realizar para minimizar los riesgos identificados. Así mismo, se propusieron un cierto número de actividades que llevaban como fin el de avanzar con aspectos técnicos que serán de utilidad en el posterior desarrollo del sistema.

En este capítulo, se definirán planes que describirán cómo realizar el trabajo restante, se creará un diseño conceptual del producto y se hará un estimado preliminar de tamaños y tiempo. Si el estimado de tamaño y/o tiempo es mayor al que se dispone, se deberá revisar la estrategia hasta que ésta se ajuste a los tiempos o tamaños requeridos. Finalmente, se definirán formalmente (vía diagramas UML de casos de uso) los actores y requerimientos del sistema.

Consideremos como base la siguiente simplificación del diagrama de actividades para la fase de Estrategia de Team Software Process. Éste fue obtenido del sitio de TSPi [4].

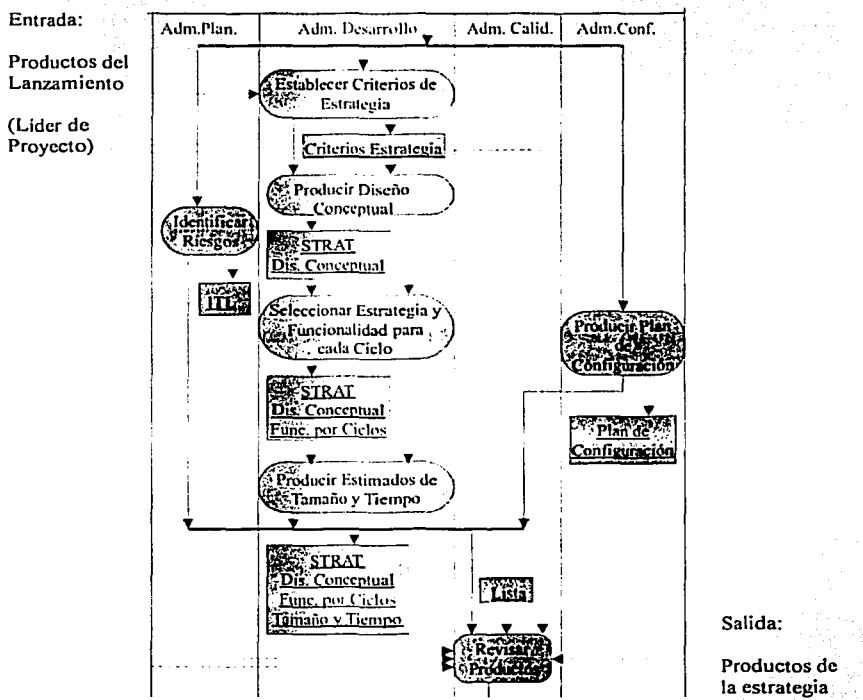


Figura 5.1

Como se puede apreciar, el rol del administrador de desarrollo es fundamental en ésta fase. El diagrama muestran en forma rectangular los productos generados en cada actividad esquematizadas a su vez, en forma de óvalos.

A continuación, nos concentraremos en detallar las actividades de la fase de estrategia y posteriormente, se presentará el diagrama de actividades para la fase de planeación con su correspondiente detalle.

5.2 Criterios de estrategia

TSPi define cuatro *criterios de estrategia* para ésta fase; el segundo propone construir una base fácilmente mejorable, pero con base en el artículo "*Una propuesta de análisis de necesidades mediante las gráficas de dependencia en la fase de estrategia de TSPi*"[7], el presente documento sugiere una modificación a ese punto.

1. El primer ciclo define un mínimo de funcionalidad que hace al producto un subconjunto funcional del desarrollo final.
2. En el primer ciclo se construye una base fácilmente *extendible*.
3. Los productos de cada ciclo son de alta calidad y fácilmente probables.
4. El diseño del producto tiene una estructura modular que permite a los miembros del equipo trabajar independientemente.

En el capítulo 4 se definió (Ver HRT 001) el alcance general del proyecto **SIPU©** donde se identificaron y distribuyeron cuatro grandes módulos:

- ✓ Administración central
- ✓ Captura de datos
- ✓ Consulta de información
- ✓ Base de datos

El presente trabajo incluirá, como caso de estudio, el subsistema de consulta de información. Se asume que la base de datos se encuentra en un estado final y estable en términos de cambio a su estructura.

Con el fin de establecer el alcance del sistema de captura y así cumplir con el primer criterio de la estrategia, a continuación se presentan los subsistemas de éste, y la secuencia de desarrollo:

La siguiente es la estrategia del subsistema de captura, en donde también se especifica, el patrón arquitectónico que se propone para éste:

Nombre:	**Subsistema de Consulta**
Tipo de desarrollo:	Iterativo e Incremental
Número de ciclos:	2
Patrón de Arquitectura:	Modelo-Vista-Controlador
Subsistemas:	- Registro de usuarios - Definición de consulta - Presentación de datos en forma tabular - Presentación de datos en forma lineal
Secuencia de desarrollo:	Procesos de definición de consultas
	Presentación de datos en forma tabular
	Presentación de datos en forma lineal
	Registro de usuarios

Tabla 5.1

En el siguiente apartado, se definirá en qué ciclo se implementará qué actividad y posteriormente, cuánto tiempo tomará realizarla y cual será el estimado en términos de tamaño (líneas de código, páginas, etc.).

5.3 Diseño conceptual

Dado que el objetivo principal del sistema es, en general, obtener datos desde un almacén persistente y presentar ésta información adecuadamente filtrada, siendo posible consultarla desde cualquier equipo con acceso a Internet, se presenta el siguiente diseño conceptual:

1. Se propone utilizar los recursos que ofrece la plataforma JAVA 2 para desarrollo de aplicaciones en Internet. Construyendo simultáneamente componentes de apoyo al desarrollo (como microsistemas de prueba y clases auxiliares) y componentes para el desarrollo mismo.

2. El producto será un conjunto de componentes separados en tres partes:

a. Componentes de visualización de información (tanto de entrada como de salida) utilizando *Java Server Pages* (JSP).
b. Componentes que modelan el problema utilizando *Java Beans*.
c. Componentes que controlan el flujo y lógica del programa utilizando *Servlets*.

3. La función de los JSP's (vistas) y de los *Servlets* (controladores) es clara, sin embargo, la función de los distintos *Java Beans* será variada:
   a. Mapeo de clases del dominio de problema con relaciones en la base de datos.
   b. Determinar el término, inicio y validez de una sesión.
   c. Gestión, validación y proceso de cadenas SQL.
   d. Control de excepciones y registro de transacciones.

Las ventajas en términos de calidad de este diseño conceptual son las siguientes:

Característica	**Justificación**
Modular	El modelo MVC promueve independencia entre los módulos de presentación de información, reglas de negocio y de control en la lógica navegacional.
Actualizable	El objetivo de definir en el primer ciclo funcionalidades básicas es extender a futuro la funcionalidad del sistema, lo que lo hace actualizable.
Probable	La baja complejidad en el primer ciclo sugiere que los módulos o clases serán fácilmente probables.
Maduro	Abordar aspectos tecnológicos tempranamente promueve desarrollos con un mayor grado de madurez
Estable	Los componentes JSP son una propuesta probada y estable de SUN.
Reutilizable	Los componentes básicos son altamente reutilizables debido a su generalidad de uso.

Tabla 5.2

5.4 Estrategia y funcionalidad para cada ciclo

Un factor que influyó en la elaboración de la siguiente tabla, fue el conjunto de herramientas y tecnologías de las que se disponía en el momento de su creación. La decisión de implementación en el ciclo I o II se debió principalmente a que, técnicamente hablando, los módulos o componentes del ciclo I (debidamente probados y funcionando) serán utilizados y posiblemente extendidos en el ciclo II.

ID	Descripción de la actividad	Ciclo I	Ciclo II
001	Crear una base de datos temporal para pruebas de conectividad y obtención de información persistente.	*	
002	Establecer conectividad con la base de datos a través de componentes JSP.	*	
003	Crear páginas JSP de despliegue de información dinámica.	*	
004	Definir lógica de navegación entre las distintas vistas.	*	
005	Implementar lógica de navegación.	*	
006	Implementar un *pool* de conexiones.	*	
007	Definir tipos de consulta general.	*	
008	Definir tipos de consulta especializada.	*	
009	Implementar consultas generales y especializadas.	*	
010	Implementar un detector de sesiones.	*	
011	Implementar un registro de transacciones.		*
012	Establecer formas CSS para uniformar el formato de presentación.	*	*
013	Establecer políticas de accesos al sistema.		*
014	Instalar en ambiente de pruebas.		*
015	Codificar clases de apoyo para correr pruebas de estrés.		*
016	Instalar en ambiente de producción.		*
017	Generar presentación *Flash*.		*

Tabla 5.3

5.5 Estimados de tamaño y tiempo por funcionalidad

Las siguientes cifras son aproximaciones que pueden diferir considerablemente de las que realmente resultarán al final del desarrollo; sin embargo, con base en la tabla anterior y en cálculos heurísticos, se propone la siguiente estimación de tamaños y tiempos (donde la unidad de medida son minutos):

ID	Descripción de la actividad	C1 LOC	C1 Tiempo	C2 LOC	C2 Tiempo
001	Crear una base de datos temporal para pruebas de conectividad y obtención de información persistente.	100	120	N/A	N/A
002	Establecer conectividad con la base de datos a través de componentes JSP.	50	180	N/A	N/A
003	Crear páginas JSP de despliegue de información dinámica.	50	180	N/A	N/A
004	Definir lógica de navegación entre las distintas vistas.	0	240	N/A	N/A
005	Implementar lógica de navegación.	400	600	N/A	N/A
006	Implementar un *pool* de conexiones.	50	120	N/A	N/A
007	Definir tipos de consulta general.	0	120	N/A	N/A
008	Definir tipos de consulta especializada.	0	120	N/A	N/A
009	Implementar consultas generales y especializadas.	400	600	N/A	N/A
010	Implementar un detector de sesiones.	50	120	N/A	N/A
011	Implementar un registro de transacciones.	N/A	N/A	50	120

012	Establecer formas CSS para uniformar el formato de presentación.	50	180	50	240
013	Establecer políticas de accesos al sistema.	N/A	N/A	200	360
014	Instalar en ambiente de pruebas.	N/A	N/A	20	60
015	Codificar clases de apoyo para correr pruebas de stress.	N/A	N/A	100	240
016	Instalar en ambiente de producción.	N/A	N/A	20	60
017	Generar presentación *Flash*.	N/A	N/A	150	120

Tabla 5.4

5.6 Plan de configuración

El propósito de esta actividad es el de realizar un plan que detalle el manejo de versiones y los cambios realizados en los productos generados. El encargado de llevar a cabo esta actividad es el administrador de configuración.

Los pasos a seguir son los siguientes:

- Definir a los integrantes de la mesa de control de cambios (MCC) encargados de aceptar o rechazar los mismos.
- Definir los documentos que serán resguardados.
- Describir el procedimiento para la solicitud de cambios.

5.6.1 Mesa de control de cambios (MCC)

La Mesa de Control de Cambios estará formada por las siguientes personas:

- Dra. Hanna Oktaba (Instructor)
- M. en C.. Guadalupe Ibargüengoitia (Instructor)
- Ing. Mauricio Espinoza (Líder de proyecto)
- Lic. Ana Pérez Arteaga (Administrador de la planeación)

5.6.2 Productos a ser controlados

Con base en la definición que TSPi provee se definen los siguientes productos a ser controlados:

- Los requerimientos.
- Los productos del diseño.
- El código fuente de los programas.
- Los materiales de prueba y los resultados de las pruebas aplicadas.
- Productos de los estándares de diseño.
- Estándares para las interfaces y pantallas.
- Bibliotecas de código reutilizables.

Todos los elementos no contemplados en estos puntos serán evaluados y en su caso aceptado por la Mesa de control de cambios.

5.6.3 Procedimiento para la solicitud de cambios

Para solicitar un cambio, el solicitante deberá llenar una forma CCR (o *Control Change Request*, por sus siglas en inglés) y ser enviada al Administrador de la configuración para que solicite la reunión con los demás integrantes de la MCC.

En los dos días hábiles siguientes, a partir de la fecha de entrega de la forma CCR, el Administrador de la configuración enviará al solicitante copia de la resolución emitida por la MCC.

5.6.4 Sobre las reuniones de la mesa de control de cambios

El administrador de la configuración estará de forma permanente atendiendo los asuntos inmediatos del equipo. La reunión se solicitará primeramente con el líder de proyecto y entre ambos tratarán de dar una resolución a cada petición. En caso en que los dos lo convengan se convocará a reunión a los

demás miembros de la mesa de control de cambios para emitir su resolución final.

5.7 Control de riesgos

El principal riesgo a combatir en la fase de estrategia es el de quedar corto con el tiempo establecido para la entrega del producto.

Para generar un estimado de tamaños y tiempos es necesario conocer que tipo de riesgos debemos enfrentar y así evaluar cuanto tiempo e inversión de código nos tomará resolver cierto punto; en cierta forma, se está planeando una administración de los tiempos y recursos a invertir en cada actividad para el desarrollo.

TSPi propone la forma ITL (Issue Tracking Log por sus siglas en inglés), que es la forma en la que se registrarán los riesgos identificados en ésta fase. Ésta es una labor del administrador de planeación, pero debe ser llevada a cabo con la colaboración estrecha del administrador de desarrollo, con el fin de que éste último discrimine si una situación (posiblemente técnica) es un verdadero riesgo a considerar o no.

En el caso de estudio, la Hoja ITL sólo presentó dos riesgos. Por cuestiones de espacio, sólo se presenta un resumen de la misma:

# Riesgo	Descripción
001	Es posible que existan cambios en los requerimientos del cliente al finalizar el primer ciclo que involucren cambios en el diseño conceptual ya expuesto y posiblemente en la planeación del proyecto.
002	Debido a las técnicas de programación empleadas, es posible que el programa no funcione en todos los navegadores de Internet

Tabla 5.5

5.8 Propuestas técnicas para la mitigación de riesgos

De la misma manera en la que se realizó en el capítulo anterior, ahora se presenta el resumen de la(s) hoja(s) de recomendaciones técnicas que servirán de apoyo a la mitigación de riesgos identificados. Con base en estas hojas, será posible realizar la planeación de tiempos y líneas de código (LOC) en siguiente apartado (que es el de planeación).

Referencia: *0009*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Planeación*, Ciclo *1*
Fecha de creación:	*Viernes 14* de *Diciembre* de *2001*
Fecha límite para ejecución:	*Viernes 14* de *Diciembre* de *2001*
Autor:	*GAA*
Origen:	*El código javaScript varía considerablemente de navegador a navegador haciendo difícil generar código para uso general.*
Propuesta:	*Considerar sólo un navegador (Microsoft iExplorer 5.0)*
Justificación:	*Se plantea esto para poder generar una primera versión entregable del producto*
Estrategia:	*Generar código javaScript sólo para iExplorer 5.0 y en otro ciclo revisar las equivalencias con otros navegadores y posiblemente codificar para ellos.*
Beneficios:	*Minimización del esfuerzo de aprendizaje e implementación de códigos con la misma funcionalidad pero con distinta implementación*
Desventajas:	*No habrá por el momento portabilidad a otros navegadores.*
Impacto en calidad:	*Robustez para la plataforma seleccionada, ya que con ésta recomendación se concentran los esfuerzos de implementación en producir un código mas sólido, pero a consta de no soportar otros navegadores. De lo anterior se impacta negativamente a la portabilidad.*

Tecnologías Involucradas:	*Java Script.*
Responsable de ejecución:	*GAA*
Aprobación:	*MEM, FNL, APA*

Tabla 5.6

Referencia: *0010*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Planeación*, Ciclo *1*
Fecha de creación:	*Viernes 14* de *Diciembre* de *2001*
Fecha límite para ejecución:	*Viernes 14* de *Diciembre* de *2001*
Autor:	*GAA*
Origen:	*El aplicativo será accedido por un número grande de usuarios, los cuales pueden ser concurrentes.*
Propuesta:	*Dotar al aplicativo de elementos que permitan soportar un número alto de accesos concurrentes.*
Justificación:	*Se plantea ésta propuesta como respuesta a un requerimiento del sistema.*
Estrategia:	*Implementar un pool de conexiones que se inicializará al arrancar el servidor de web y que permanecerá encendido a lo largo de la vida de la aplicación.*
Beneficios:	*El aplicativo podrá atender un número alto de peticiones de consultas a la base de datos sin sufrir detrimento en su capacidad de respuesta*
Desventajas:	*Mayor inversión de tiempo y recursos humanos para la implementación del pool.*
Impacto en calidad:	*Desempeño y escalabilidad en términos del número de accesos concurrentes al sistema.*
Tecnologías Involucradas:	*Java*
Responsable de	*GAA*

ejecución:	
Aprobación:	*MEM, FNL.*

Tabla 5.7

Referencia: *0011*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Planeación*, Ciclo *1*
Fecha de creación:	*Viernes 14* de *Diciembre* de *2001*
Fecha límite para ejecución:	*Viernes 14* de *Diciembre* de *2001*
Autor:	*GAA*
Origen:	*Dado que el número y complejidad de consultas a la base de datos es grande, la ejecución de consultas similares debe ser mantenida en un caché y éste debe ser destruido al terminar una sesión. Adicionalmente, se desea saber la dirección IP de la máquina que accede al aplicativo.*
Propuesta:	*Dotar al aplicativo de un detector de inicio de sesiones remotas.*
Justificación:	*Con esta propuesta se obtiene una adecuada administración del espacio en disco usado por el caché y es posible, al mismo tiempo, detectar accesos remotos al servidor.*
Estrategia:	*Implementar un detector de sesiones que se inicializará al arrancar el servidor de web y que permanecerá encendido a lo largo de la vida de la aplicación.*
Beneficios:	*Reducción del espacio de información temporal en disco, seguimiento del número y procedencia de los accesos remotos al server.*
Desventajas:	*Mayor inversión de tiempo y recursos humanos para la implementación del detector.*
Impacto en calidad:	*Desempeño (mejor utilización de los recursos del sistema. En este caso, el espacio en disco duro.) Seguridad. Capacidad para poder realizar auditorías de*

	acceso al servidor e identificar posibles atacantes.
Tecnologías Involucradas:	*Java*
Responsable de ejecución:	*GAA*
Aprobación:	*MEM, FNL.*

Tabla 5.8

Referencia: *0012*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Planeación*, Ciclo *1*
Fecha de creación:	*Viernes 14* de *Diciembre* de *2001*
Fecha límite para ejecución:	*Viernes 14* de *Diciembre* de *2001*
Autor:	*GAA*
Origen:	*Se requiere de un mecanismo que permita la revisión de actividad del aplicativo fuera de línea.*
Propuesta:	*Dotar al aplicativo de un registro de transacciones.*
Justificación:	*Un registro de transacciones guardará información de las actividades realizadas por el servidor y por los usuarios durante su operación y podrá ser revisado posteriormente fuera de línea, ya que sólo es un archivo plano.*
Estrategia:	*Implementar un registro de transacciones que se inicializará al arrancar el servidor y que permanecerá encendido a lo largo de la vida de la aplicación.*
Beneficios:	*Se obtendrá un registro por transacción de la actividad del servidor, registrando también la fecha, hora e IP generador.*
Desventajas:	*Mayor inversión de tiempo y recursos humanos para la implementación del registro.*
Impacto en calidad:	*Seguridad. Capacidad para poder realizar auditorías de acceso al servidor e identificar posibles atacantes.*
Tecnologías Involucradas:	*Java*

Responsable de ejecución:	*GAA*
Aprobación:	*MEM, FNL.*

Tabla 5.9

Con ésta última recomendación concluye la parte correspondiente a la estrategia del primer ciclo del sistema. A continuación, se presenta la planeación del mismo.

5.9 Planeación

La segunda fase del flujo de elaboración es la fase de planeación. En ésta fase el objetivo a conseguir entre el administrador de planeación y el de calidad es, dado que estamos en el primer ciclo, la generación de cinco documentos que estiman tamaños, tiempos, tareas y calendarios planeando así la distribución de recursos humanos y materiales. Los productos concretos a desarrollar son:

- Forma SUMS (Resumen de tamaños)
- Forma TASK (Tareas)
- Forma SCHEDULE (Calendario)
- Forma SUMQ (Resumen de calidad)
- Planes Individuales

A continuación se presenta el diagrama de actividades (extraído del sitio de TSPi[5]) para la fase de planeación:

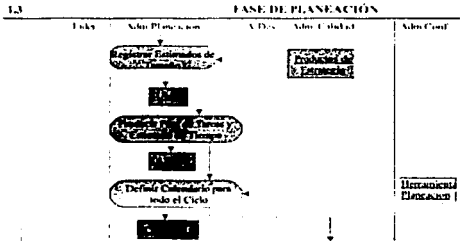


Figura 5.2 (Continúa en la siguiente página)

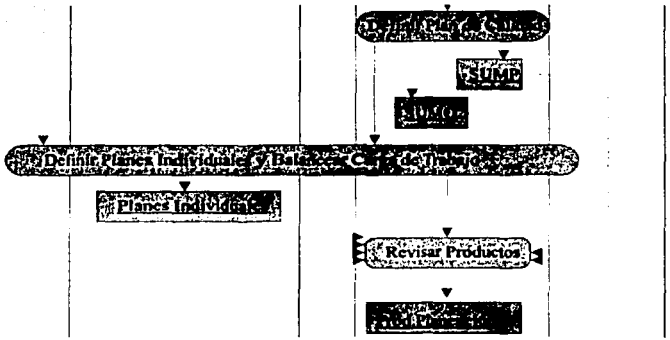


Figura 5.2 (Continuación)

5.9.1 Forma SUMS

El primer documento es la forma SUMS, cuyo objetivo es la estimación del tamaño de las distintas partes del sistema y otros productos a desarrollar.

TSPi Size Summary - Form SUMS (Planned Size)

Name Gustavo Arellano
Team Metasoft
Date 15/09/2002
Instructor Hanna Oktaba
Cycle 1

Part ID	Part Name	Assembly/Part	Part of	Owner	Size Measure	Base	Deleted	Modified	Added	Reused	New and Changed	Total
1	SRS	P	SYSTEM	MEM	Text Pages				1		1	1
2	HLD	P	SYSTEM	MEM	Text Pages				15		15	15
3	DLD	P	SYSTEM	MEM	Text Pages				18		18	18
4	Crear una base de datos temporal para pruebas de conectividad y obtención de información persistente	P	SYSTEM	MEM	LOC				175		175	175
5	Establecer conectividad con la base de datos a través de	A	SYSTEM	MEM	LOC	0	0	0	320	0	320	320

componentes JSP.													
6	Crear páginas JSP de despliegue de información dinamica.	A	Establecer conectividad con la base de datos a través de componentes JSP.	GAA	LOC				120		120	120
7	Definir lógica de navegación entre las distintas vistas	A	SYSTEM	MEM	Text Pages	0	0	0	4	0	4	4
8	Implementar lógica de navegación	A	Definir lógica de navegación entre las distintas vistas	GAA	Text Pages				4		4	4
9	Implementar un pool de conexiones	A	Establecer conectividad con la base de datos a través de componentes JSP.	GAA	LOC				200		200	200
10	Definir tipos de consulta general	A	SYSTEM	FNL	Text Pages				3		3	3
11	Definir tipos de consulta especializada	A	SYSTEM	FNL	Text Pages				10		10	10
12	Implementar consultas generales y Especializadas	A	SYSTEM	GAA	LOC				1500		1500	1500
13	Implementar un detector de sesiones	A	SYSTEM	GAA	LOC				70		70	70

Tabla 5.10

5.9.2 Forma TASK

El segundo documento es la forma TASK, cuyo objetivo es la estimación del tiempo para completar cada tarea planeada.

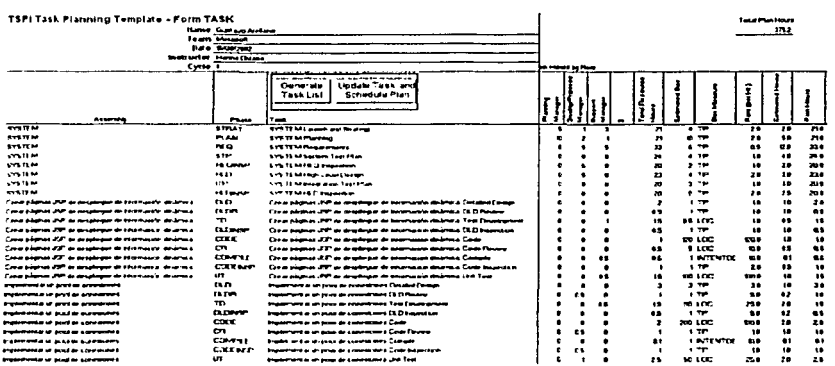


Tabla 5.11 (Continúa en la siguiente página)

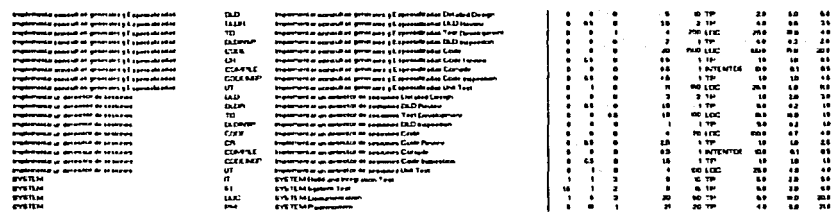


Tabla 5.11 (Continuación)

Nota: Debido al tamaño de algunas de las formas TSPi, éstas fueron reducidas para poder ser incluidas en el presente trabajo. Si se desea, es posible ver el detalle de cada una de estas formas en el disco compacto incluido.

5.9.3 Forma SCHEDULE

El tercer documento es la forma SCHEDULE, cuyo objetivo es generar una estimación de las fechas en las que se planea realizar liberaciones de subproductos.

TSPi Schedule Planning Template - Form SCHEDULE

Name Gustavo Arellano
Team Metasoft
Date 15-Sep-02
Instructor Hanna Oktaba
Cycle 1

Total Task Plan Hours 375.2
Total Schedule Plan Hours 375.5
Difference -0.3

Date	Week	Planned Hours	Cumulative Planned Hours	Actual Hours	Cumulative Actual Hours	Planned Value	Cumulative Planned Value	Earned Value	Cumulative EV
16/Sep/02	1	40.0							
22/Sep/02	2	40.0	80.0						
29/Sep/02	3	40.0	120.0						
7/Oct/02	4	40.0	160.0						
14/Oct/02	5	40.0	200.0						
21/Oct/02	6	40.0	240.0						
28/Oct/02	7	40.0	280.0						
4/Nov/02	8	40.0	320.0						

11/Nov/02	9	40.0	360.0
18/Nov/02	10	15.5	375.5

Tabla 5.12

5.9.4 Forma SUMQ

El cuarto documento es la forma SUMQ, cuyo objetivo es establecer parámetros de calidad y métricas para poder comparar los resultados planeados con los que realmente serán obtenidos al final del ciclo.

TSPi Quality Plan - Form SUMQ

Name Gustavo Arellano — Date 15/09/2002

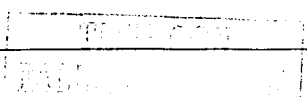
Team Metasoft — Instructor Hanna Oktaba

Cycle 1

SYSTEM Percent Defect Free	Actual
In Compile	100.0%
In Unit Test	100.0%
In Build and Integration Test	100.0%
In System Test	100.0%
In Acceptance Test	90%
In Product Life	95%

Defects/Page	Plan	Actual
REQ Inspection	0.00	0.00
HLD Inspection	0.00	0.00

Defects/KLOC	Plan	Actual
DLD Review	0.00	0.00
DLD Inspection	0.00	0.00
Code Review	0.00	0.00
Compile	0.00	0.00
Code Inspection	0.00	0.00
Unit Test	0.00	0.00
Build and Integration Test	0.00	0.00
System Test	0.00	0.00
Total Development	0.00	0.00
Acceptance Test	0.00	0.00
Product Life	0.00	0.00
Total	0.00	0.00



Defect Ratios	Plan	Actual
DLD Review/Unit Test	0.00	0.00
Code Review/Compile	0.00	0.00

Development Time Ratios	Plan	Actual
REQ Inspection/Requirements	0.61	0.00
HLD Inspection/High-Level Design	0.87	0.00
Detailed Design/Code	0.48	0.00
DLD Review/Detailed Design	0.50	0.00
Code Review/Code	0.46	0.00

Inspection/Review Rates	Plan	Actual
REQ Inspection	2.55	0.00
HLD Inspection	0.00	0.00
DLD Review	317.69	0.00
DLD Inspection	0.00	0.00
Code Review	165.20	0.00
Code Inspection	258.13	0.00

A/FR	1.50	0.00

Phase Yields	Plan	Actual
Planning		0%
Requirements		0%
System Test Plan		0%
REQ Inspection		0%
High-Level Design		0%
Integration Test Plan		0%
HLD Inspection		0%
Detailed Design		0%
DLD Review		0%
Test Development		0%
DLD Inspection		0%
Code		0%
Code Review		0%
Compile		0%
Code Inspection		0%
Unit Test		0%
Build and Integration Test		0%
System Test		0%

Process Yields	Plan	Actual
% Before Compile	0%	0%

% Before Unit Test	0%	0%
% Before Build and Integration Test	0%	0%
% Before System Test	0%	0%
% Before Acceptance Test	0%	0%

Defect Injection Rates (Defects Injected Per Hour)	Plan	Actual
Planning		0.00
Requirements		0.00
System Test Plan		0.00
REQ Inspection		0.00
High-Level Design		0.00
Integration Test Plan		0.00
HLD Inspection		0.00
Detailed Design		0.00
DLD Review		0.00
Test Development		0.00
DLD Inspection		0.00
Code		0.00
Code Review		0.00
Compile		0.00
Code Inspection		0.00
Unit Test		0.00
Build and Integration Test		0.00
System Test		0.00

Defect Removal Rates	Plan	Actual
Planning	0.00	0.00
Requirements	0.00	0.00
System Test Plan	0.00	0.00
REQ Inspection	0.00	0.00
High-Level Design	0.00	0.00
Integration Test Plan	0.00	0.00
HLD Inspection	0.00	0.00
Detailed Design	0.00	0.00
DLD Review	0.00	0.00
Test Development	0.00	0.00
DLD Inspection	0.00	0.00
Code	0.00	0.00
Code Review	0.00	0.00
Compile	0.00	0.00
Code Inspection	0.00	0.00
Unit Test	0.00	0.00
Build and Integration Test	0.00	0.00
System Test	0.00	0.00

Tabla 5.13

5.9.5 Planes individuales

El quinto documento comprende la generación e integración de los planes individuales de trabajo (Formas WEEK). Cada administrador generará su plan individual y éste será integrado al de los demás. Cada plan contendrá los tamaños, fechas y tiempos a los que el administrador se compromete a cumplir para este ciclo. A continuación –y por razones de espacio- se presenta sólo uno de ellos, la forma WEEK de la primera semana del administrador de desarrollo.

TSPi Week Summary - Form WEEK

Name	Gustavo Arellano	**Date**	15/09/2002
Team	Metasoft	**Instructor**	Hanna Oktaba
Week	1	**Cycle**	1

Weekly Data	**Plan**	**Actual**
Project hours for this week	40.0	0.0
Project hours this cycle to date	0.0	0.0
Earned value for this week	0.0	0.0
Earned value this cycle to date	0.0	0.0
Total hours for the tasks completed this phase to date		

Team Member Weekly Data	**Team Plan Hours**	**Team Actual Hours**	**Team Plan Value**	**Team Earned Value**
Team Leader				
Development Manager				
Planning Manager				
Quality/Process Manager				
Support Manager				
0				
Totals	0.0	0.0	0.0	0.0

Development Tasks Completed	**Plan Hours**	**Actual Hours**	**Earned Value**	**Plan Week**

Tabla 5.14

Con este último documento concluimos la fase de planeación del flujo de elaboración del primer ciclo. A continuación se presenta la tercer y última fase de este flujo. La fase de requerimientos.

5.10 Requerimientos

La fase de requerimientos es la tercera y la última del flujo de elaboración. El objetivo principal es la generación de una especificación de requerimientos. Una especificación de requerimientos (o SRS por sus siglas en inglés *Software Requirments Specification*) es una colección de documentos y modelos que intentan describir de una forma no ambigua un sistema de software a ser construido[1].

Como auxiliar en la especificación de requerimientos se proponen las dos siguientes HRT's:

Referencia: *0013*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Requerimientos*, Ciclo *1*
Fecha de creación:	*Viernes 4 de Enero de 2002*
Fecha límite para ejecución:	*Viernes 4 de Enero de 2002*
Autor:	*GAA*
Origen:	*El proceso de levantamiento de requerimientos debe ser extendido.*
Propuesta:	*Realizar entrevistas incrementales en términos del detalle de los cuestionamientos.*
Justificación:	*Con base en la primera entrevista, se podrán diseñar nuevas entrevistas con cuestionamientos específicos y efectivos en la definición del problema.*

[1] Obtenido de "Building Web Applications with UML" by Jim Conallen. Ed. Addison-Wesley. Cap 8 P 115

Estrategia:	*Realizar inicialmente una entrevista completamente general y posteriormente dos o más entrevistas particulares.*
Beneficios:	*Obtención de un conjunto de especificaciones preciso.*
Desventajas:	*Acumulación de más documentos al expediente del desarrollo.*
Impacto en calidad:	*Funcionalidad. Se obtendrán mas elementos para la definición de los aspectos funcionales del sistema.*
Tecnologías Involucradas:	*N/A*
Responsable de ejecución:	*GAA*
Aprobación:	*MEM, FNL, APA*

Tabla 5.15

Respecto a la forma en la que las entrevistas dirigidas deben ser realizadas, existe una referencia útil (William S. Davis "Tools and techniques for structured systems analysis and design" [15]). A continuación se exponen las tres etapas que Davis propone como componentes de una entrevista:

Preparando la entrevista

La siguiente lista de puntos debe ser considerada de manera previa a la realización de la entrevista.

- Definir el propósito de la entrevista.
- Seleccionar la persona o grupo de personas a ser entrevistado.
- Realizar investigaciones acerca del tópico o temas a ser tratados durante la sesión incluyendo aquellos que no necesariamente estén relacionados con el dominio del problema.
- Enunciar preguntas específicas y no ambiguas dirigidas al grupo a ser entrevistado con base en lo capturado en la entrevista no dirigida.
- Calendarizar la entrevista. Estableciendo mes, día, hora, lugar de reunión y asegurarse de que cada una de las personas que participarán en ésta, se encuentren enterados.

La entrevista por sí misma

Una entrevista se compone de tres partes que deben ser llevadas a cabo en el orden establecido y que a continuación se expone:

- Apertura. Identificarse, compenetrarse, exponer el objetivo de la reunión y definir un protocolo a seguir durante la misma.
- Cuerpo. Llevar a cabo la secuencia descrita en el protocolo de la reunión y en la parte de las preguntas ir de las más generales y abstractas a las más particulares y concretas.
- Cierre. Establecer los pasos a seguir en las siguientes reuniones, solicitar información que se requirió durante la reunión y no se pudo obtener, calendarizar nuevas reuniones.

Seguimiento

Para la parte de seguimiento se recomienda transcribir lo más pronto posible lo capturado durante la entrevista, generar nuevos cuestionamientos y redactar la minuta de lo acontecido durante la reunión.

Referencia: *0014*	**Hoja de recomendaciones técnicas**
Proyecto:	*SIPU*
Fase / Ciclo:	*Requerimientos*, Ciclo *1*
Fecha de creación:	*Viernes 4 de Enero de 2002*
Fecha límite para ejecución:	*Viernes 4 de Enero de 2002*
Autor:	*GAA*
Origen:	*El proceso de levantamiento de requerimientos debe ser extendido.*
Propuesta:	*Realizar grabaciones de audio y si es posible de video de las entrevistas.*
Justificación:	*Con la capacidad de reproducir de manera íntegra las sugerencias o peticiones del cliente será posible tener apego a sus requerimientos.*

Estrategia:	*Para cada entrevista deberá ser generada una cinta ya sea de audio o de video que incluya hora, lugar y fecha del evento. Esta cinta también deberá incluir los nombres de los participantes en dicha entrevista.*
Beneficios:	*Capacidad para poder reproducir lo expuesto por el cliente sin tenerlo que solicitar explícitamente a él. Adicionalmente se tiene un registro para presentar en caso de que el sistema requiera cambios a futuro.*
Desventajas:	*Acumulación de más artefactos al expediente del desarrollo. Incremento en el costo de los insumos por proyecto. Renuencia por parte del cliente a ser grabado.*
Impacto en calidad:	*N/A*
Tecnologías Involucradas:	*N/A*
Responsable de ejecución:	*GAA*
Aprobación:	*MEM, FNL, APA*

Tabla 5.16

Adicionalmente, dentro de TSPi, existen una serie de actividades a llevar a cabo para la generación del documento SRS. En el proceso de generación de este documento obtendremos también una especificación para la elaboración del documento STP (*System testing plan*, por sus siglas en inglés), que es el plan de pruebas para el sistema. Estas actividades son:

- ✓ Generar la descripción de necesidades del sistema
- ✓ Generar diagramas de casos de uso
- ✓ Definir arquitectura y prototipo del sistema
- ✓ Generar e integrar plan de pruebas del sistema

A continuación se presenta el detalle de las actividades (con sus respectivas resultantes) a realizar en ésta fase.

5.10.1 Revisar la descripción de necesidades del sistema

El propósito de ésta actividad es clarificar y unificar el entendimiento acerca de las necesidades del sistema entre todos los miembros del equipo y el cliente. A continuación se exponen las necesidades generales del sistema:

- La coordinación científica para la investigación universitaria requiere de la elaboración de un sistema de consulta para programas universitarios a través de Internet, debiendo ser éste, un sistema poderoso y al mismo tiempo, un versátil motor de búsqueda de información.

- Se requiere que el sistema solicite que los consultantes se identifiquen genéricamente como alumnos, investigadores, gobierno, iniciativa privada y posiblemente otros tipos por definir.

- Será posible, de manera opcional, identificarse a través de una forma de registro detallada que solicitará datos específicos del consultante, como su nombre, apellidos, correo electrónico y demás datos personales que serán almacenados en la base de datos del programa.

De acuerdo a lo que sugiere TSPi, el proyecto será desarrollado en ciclos incrementales. En el primer ciclo se elaborará el núcleo de funcionalidades básicas del sistema. Se establecerá, así mismo, una biblioteca para depositar las funciones y clases reusables, así como todas las constantes que se emplearán en el proyecto.

El producto se dividirá en las siguientes funcionalidades, representadas cada una por una interfaz de usuario:

i. Bienvenida.
ii. Petición de la identidad del consultante (por su actividad).
iii. Petición de información personal del consultante, la cual es para el consultante, opcional.
iv. Datos inválidos del consultante.
v. Interfases para consultas nuevas (6).

vi. Interfases para consultas sobre consultas anteriores (6).
vii. Aviso: cero tuplas resultantes.

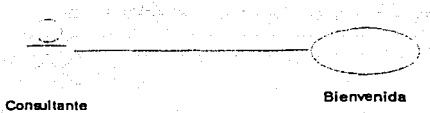
En el primer ciclo se construirán todas las interfaces de usuario sin ninguna funcionalidad detrás, excepto la v y la vii que se desarrollarán completamente.

5.10.2 Generar diagramas de casos de uso.

El propósito de esta actividad es definir actores y escenarios del proyecto en un nivel de abstracción alto.

La notación a usar será UML y para cada caso de uso se presentará una tabla que describe el comportamiento del actor y a continuación la respuesta que éste obtiene del sistema.

5.10.2.1 Caso de Uso: Bienvenida



	Consultante	Sistema	
Bienvenida	Teclea o navega hasta la página de Bienvenida	Muestra mensaje animado de Bienvenida y ofrece una liga o botón para continuar	

Tabla 5.17

5.10.2.2 Caso de Uso: Entrar al Sistema

	Consultante		Sistema	
Entrar al sistema	Selecciona entrar al sistema desde la ventana de bienvenida.		Muestra una ventana de selección de tipo de consultas.	
	Desde un objeto de tipo combo se selecciona que tipo de usuario es él mismo.		Registra su selección y abre la ventana de selección de consultas	E1
E1	El usuario seleccionó la opción vacía (Default)		Solicita seleccionar una opción no vacía (es decir, distinta de la de default)	

Tabla 5.18

5.10.2.3 Caso de Uso: Registro

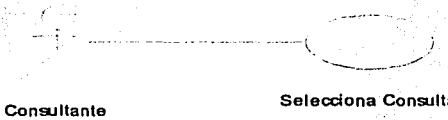
Consultante

Registro

	Consultante		Sistema	
Registro	Selecciona botón de registro de datos personales desde ventana Entrada		Muestra ventana para rellenar datos personales	
	Llena datos y Enter		Recibe datos y los guarda en base de datos	E1
			Manda el control a la venta de tipo de búsqueda	
E1	Datos no validos o incompletos		Muestra ventana de error y lleva a pantalla de captura de datos personales	

Tabla 5.19

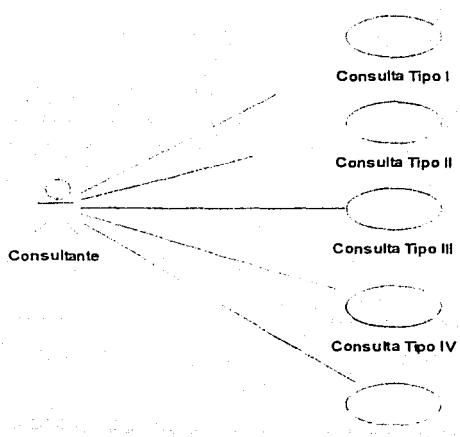
5.10.2.4 Caso de Uso: Seleccionar Consulta



	Consultante	Sistema	
Selecciona tipo consulta	Selecciona desde un objeto de tipo combo el tipo de consulta que desea efectuar	Muestra la ventana asociada a la interfase de consulta seleccionada	

Tabla 5.20

5.10.2.5 Caso de Uso: Define y Ejecuta Consulta



	Consultante	Sistema	
Define y ejecuta consulta	Provee información pertinente a la consulta vía los distintos objetos que la interfaz ofrece.	Muestra resultado de consulta y ofrece realizar una refinación de la misma	E1 E2
E1	Generó una consulta vacía	Muestra un mensaje de advertencia y permite regresar a reformular la consulta	
E2	Datos no validos	Muestra un mensaje de error y permite regresar a reformular la consulta.	

Tabla 5.21

5.10.3 Definir arquitectura y prototipo del sistema.

El propósito de ésta actividad es presentar una propuesta formal de cómo será la vista arquitectónica y funcional del sistema en un nivel de abstracción alto.

5.10.3.1 Arquitectura

Se propone una arquitectura basada en el patrón arquitectónico MVC (Modelo Vista Controlador). Los siguientes esquemas esbozan este patrón tomando en cuenta que el aplicativo será desarrollado con tecnología J2EE.

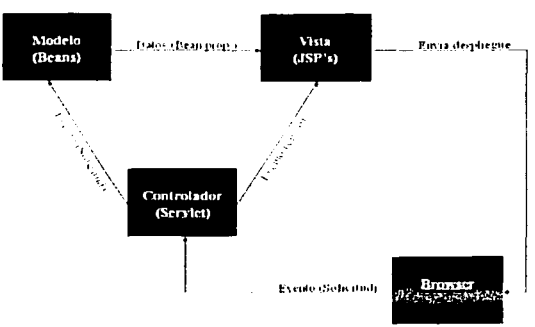


Figura 5.3

El flujo del sistema queda esbozado por el siguiente esquema (Figura 5.4) que en su momento se convertirá en un diagrama de actividades mas detallado. Adicionalmente, cabe mencionar que este esquema facilita el seguimiento y comprensión de lo que en el siguiente apartado se expone: el prototipo del sistema.

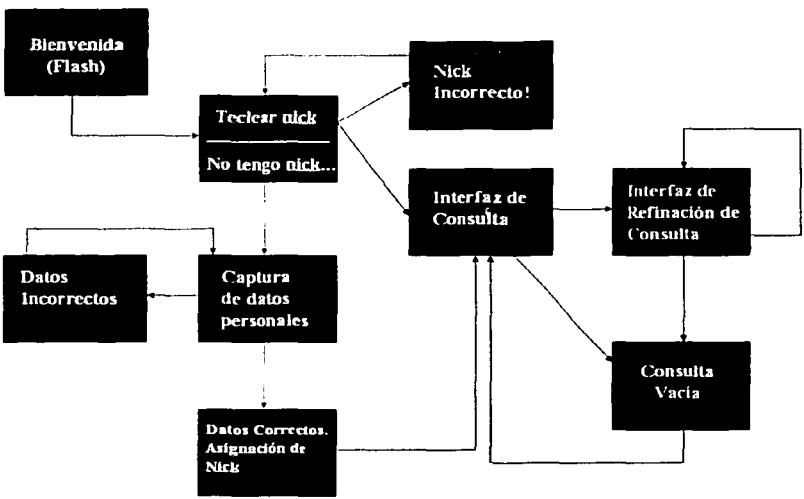
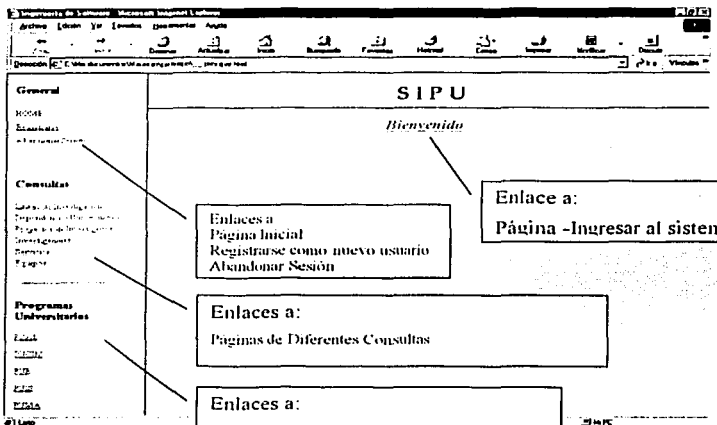
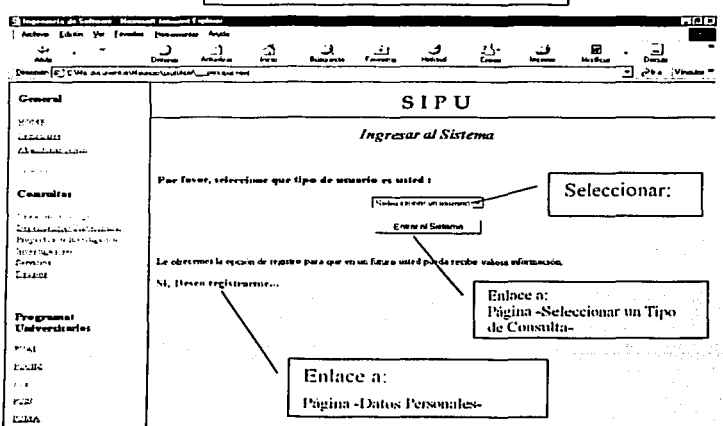


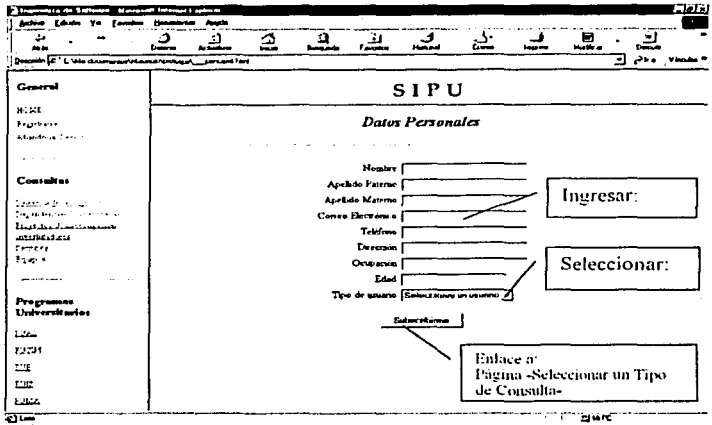
Figura 5.4

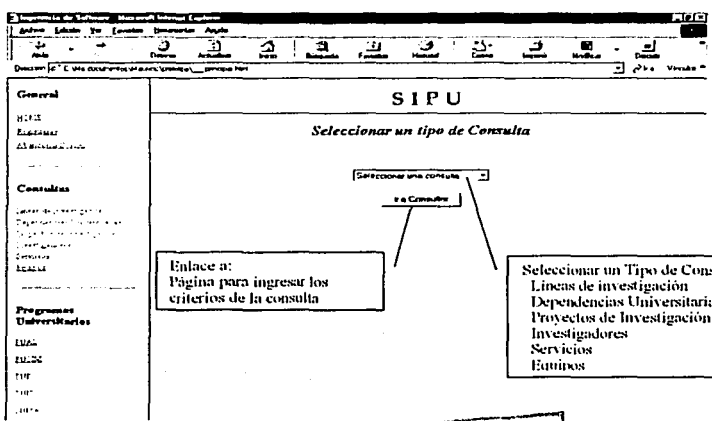
5.10.3.2 Prototipo del sistema

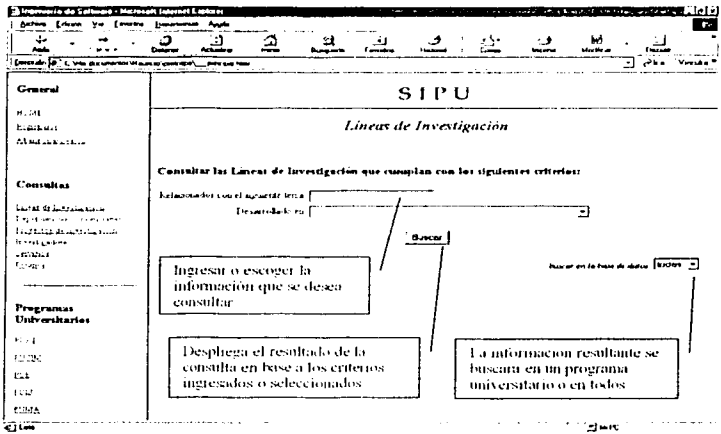
El prototipo del sistema queda descrito por la siguiente colección de imágenes:

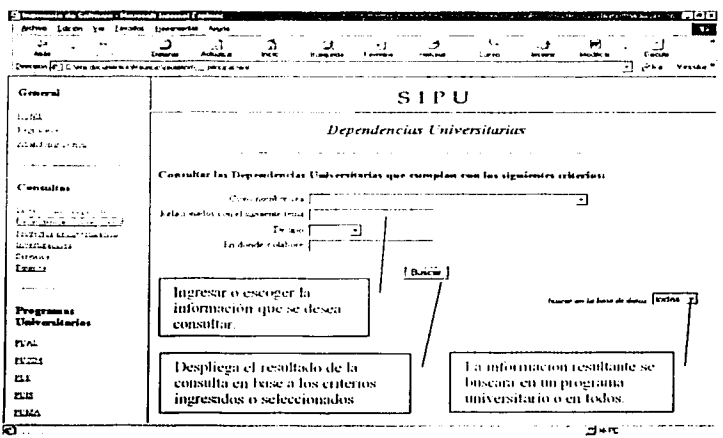


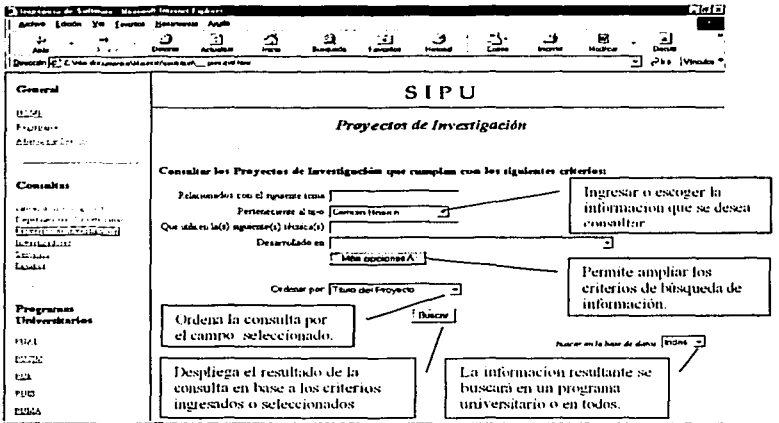


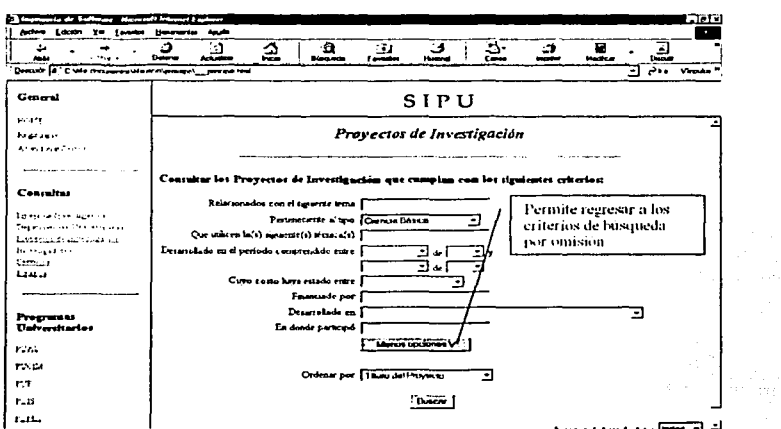

General
Consultas
Programas Universitarios
SIPU
Datos Personales
Nombre
Apellido Paterno
Apellido Materno
Correo Electrónico
Teléfono
Dirección
Ocupación
Edad
Tipo de usuario
Seleccionar un usuario
Ingresar:
Seleccionar:
Enlace a:
Página -Seleccionar un Tipo de Consulta-

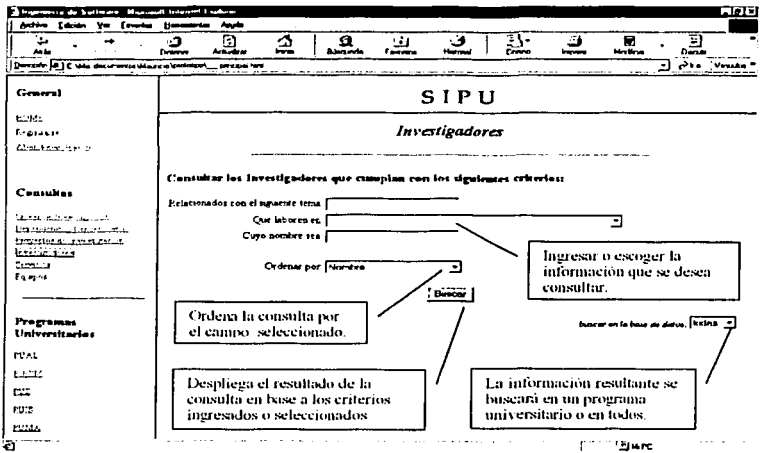

General
Consultas
Programas Universitarios
SIPU
Seleccionar un tipo de Consulta
Seleccionar una consulta
Ir a Consultar
Enlace a:
Página para ingresar los criterios de la consulta
Seleccionar un Tipo de Consulta:
Líneas de investigación
Dependencias Universitarias
Proyectos de Investigación
Investigadores
Servicios
Equipos

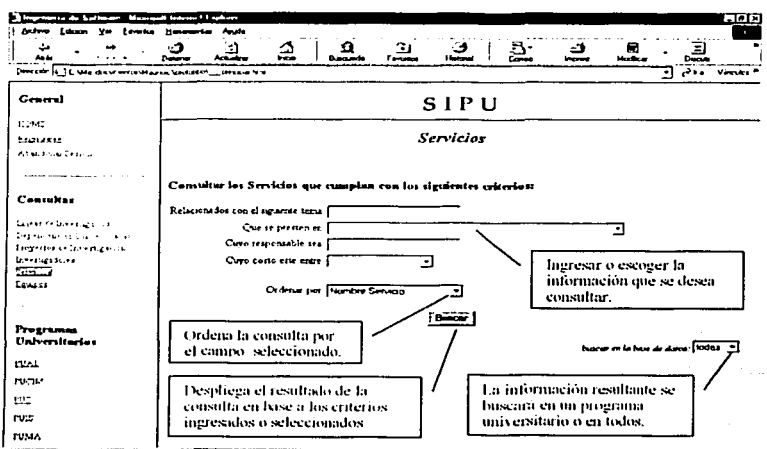


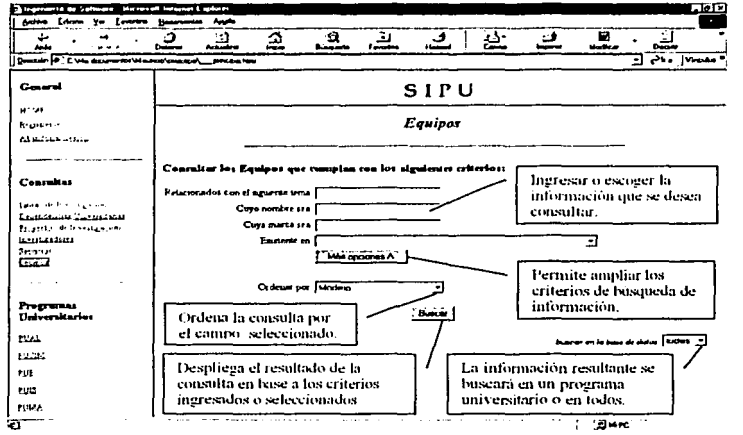


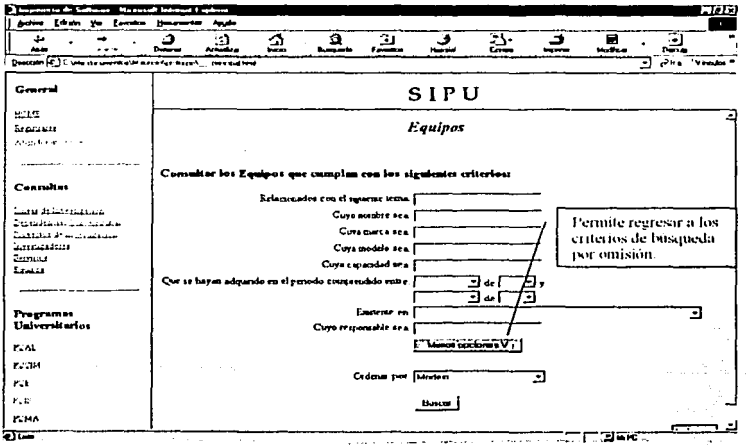

SIPU
Proyectos de Investigación
Consultar los Proyectos de Investigación que cumplan con los siguientes criterios:
Ingresar o escoger la información que se desea consultar
Permite ampliar los criterios de búsqueda de información.
Ordena la consulta por el campo seleccionado.
Despliega el resultado de la consulta en base a los criterios ingresados o seleccionados
La información resultante se buscará en un programa universitario o en todos.


SIPU
Proyectos de Investigación
Consultar los Proyectos de Investigación que cumplan con los siguientes criterios:
Permite regresar a los criterios de búsqueda por omisión


General
Consultas
Programas Universitarios
SIPU
Investigadores
Consultar los Investigadores que cumplan con los siguientes criterios:
Relacionados con el siguiente tema
Que laboren en
Cuyo nombre sea
Ordenar por
Nombre
Buscar
Ingresar o escoger la información que se desea consultar.
Ordena la consulta por el campo seleccionado.
Despliega el resultado de la consulta en base a los criterios ingresados o seleccionados
La información resultante se buscará en un programa universitario o en todos.
todos


General
Consultas
Programas Universitarios
SIPU
Servicios
Consultar los Servicios que cumplan con los siguientes criterios:
Relacionados con el siguiente tema
Que se presten en
Cuyo responsable sea
Cuyo costo este entre
Ordenar por
Nombre Servicio
Buscar
Ingresar o escoger la información que se desea consultar.
Ordena la consulta por el campo seleccionado.
Despliega el resultado de la consulta en base a los criterios ingresados o seleccionados
La información resultante se buscará en un programa universitario o en todos.
todos


SIPU
Equipos
Consultar los Equipos que cumplan con los siguientes criterios:
Relacionados con el siguiente tema
Cuyo nombre sea
Cuya marca sea
Existente en
Más opciones A
Ordenar por
Mostrar
Buscar
Buscar en la base de datos
Todos
Programas Universitarios
PCMA
PUE
PUIS
PUMA
Ingresar o escoger la información que se desea consultar.
Permite ampliar los criterios de búsqueda de información.
Ordena la consulta por el campo seleccionado.
Despliega el resultado de la consulta en base a los criterios ingresados o seleccionados
La información resultante se buscará en un programa universitario o en todos.


SIPU
Equipos
Consultar los Equipos que cumplan con los siguientes criterios:
Relacionados con el siguiente tema
Cuyo nombre sea
Cuya marca sea
Cuyo modelo sea
Cuya capacidad sea
Que se hayan adquirido en el periodo comprendido entre
de
y
de
Existente en
Cuyo responsable sea
Más opciones V
Ordenar por
Mostrar
Buscar
Programas Universitarios
PCMA
PUE
PUIS
PUMA
Permite regresar a los criterios de búsqueda por omisión.

5.10.4 Generar e integrar plan de pruebas del sistema.

A continuación se presenta al plan de pruebas para probar el sistema SIPU. En este plan se presentan una serie de tablas que especifican que tipo de prueba deberá ser realizada en su momento. Estas tablas tienen la siguiente estructura:

- Entrada. Procedimiento necesario para que el ambiente esté preparado para la realización de la prueba.
- Resultante. Comportamiento esperado en el momento de finalizar la prueba.
- Condiciones. Conjunto de condiciones que deben existir al momento de ejecutarse la prueba.
- Procedimiento de prueba. Secuencia de pasos que se deben llevar a cabo para la realización de la prueba.

Conviene mencionar en este punto que estas pruebas no incluyen las pruebas unitarias, que serán realizadas de manera paralela a la codificación de los diferentes elementos constituyentes de cada módulo.

Finalmente, cabe mencionar que el plan de pruebas presentado a continuación fue basado principalmente en los casos de uso identificados hasta el momento:

5.10.4.1 Caso de prueba "*Bienvenida*"

Entrada	Teclear o navegar hasta la página de Bienvenida del Sistema.
Resultante	Se muestra el mensaje animado de Bienvenida y se ofrece una liga o botón para continuar/entrar.
Condiciones	El ingreso al sistema será permitido sólo después de elegir el botón continuar/entrar.
Procedimiento de prueba	1. Ir a la página inicial del sistema.. 2. Verificar que la animación y la página de bienvenida funcionan correctamente y su

	visualización corresponde al diseño original. Utilizar para ello diferentes tipos y versiones de navegadores y diferentes tipos de arquitecturas de computadoras y sistemas operativos. 3. Oprimir el botón continuar/entrar. Lo llevará a la ventana de "Entrada al Sistema".

Tabla 5.22

5.10.4.2 Caso de prueba "Entrar al sistema"

Entrada	Oprimir la opción entrar al sistema desde la ventana de bienvenida.
Resultante	1. Se despliega una ventana donde, desde un objeto de tipo *combo*, el consultante selecciona el tipo de usuario que le corresponde. 2. Se registra la selección. 3. Se muestra una ventana de Selección de Tipo de Consultas.
Condiciones	1. El acceso a esta página es permitido únicamente cuando el consultante provenga de la página de Bienvenida y haya pulsado el botón entrar/continuar. 2. Seleccionar el tipo de consultante para poder acceder a la ventana de selección de tipo de consultas. 3. No seleccionar la opción vacía por defecto.
Procedimiento de prueba	1. Ingresar a esta página web dándole el *url* directo desde el navegador. Comprobar que el sistema remite al consultante a la página de Bienvenida. 2. Ingresar a esta página web desde la página de

	Bienvenida. 3. Seleccionar el tipo de usuario al que pertenece. Comprobar que el sistema lo lleva a la Ventana de Selección de Tipo de Consulta. 4. Seleccionar la opción vacía que se muestra por defecto. Comprobar que el sistema no le permite continuar y que le envía un mensaje de error. 5. Seleccionar Registrar. Comprobar que el sistema lo lleva a la ventana de Registro.

Tabla 5.23

5.10.4.3 Caso de prueba "Registro"

Entrada	1. Seleccionar el botón de registro de datos personales desde la Ventana de Entrada. 2. Llenar datos solicitados y oprimir *Enter* o seleccionar Aceptar.
Resultante	1. Se muestra la ventana para llenar datos personales. 2. Se reciben los datos y se guardan en la base de datos. 3. Se manda el control a la ventana de Tipo de Búsqueda.
Condiciones	Llenar los datos marcados como "Necesarios" completa y adecuadamente
Procedimiento de prueba	1. Seleccionar la opción de Registro desde cualquiera de las ventanas del sistema (excepto desde la ventana de Bienvenida). Comprobar que lo lleva a la ventana de Registro. 2. No insertar datos en los campos marcados

	como necesarios. Comprobar que el sistema devuelve un mensaje de error y le pide que registre los datos correctamente. 3. Registrar los datos correctamente y seleccione Aceptar. Comprobar que el sistema lo lleva a la Ventana de Tipo de Búsqueda.

Tabla 5.24

5.10.4.4 Caso de prueba "Selecciona consulta"

Entrada	Seleccionar el tipo de consulta que se desea efectuar
Resultante	Se muestra la ventana asociada a la interfaz de consulta seleccionada
Condiciones	Seleccionar algún tipo de consulta a realizar
Procedimiento de prueba	1. Seleccionar uno a uno los diferentes tipos de consulta. Verificar que el sistema lo lleva a la página del tipo de consulta elegido. 2. Ingresar a esta página web dándole el *url* directo desde el navegador. Comprobar que el sistema remite al consultante a la página de Bienvenida.

Tabla 5.25

5.10.4.5 Caso de prueba "Define y ejecuta consulta"

Entrada	Proveer la información pertinente a la consulta vía los distintos objetos que la interfaz ofrece
Resultante	Se muestra el resultado de la consulta y se ofrece enlaces para refinar la misma
Condiciones	Ninguna

Procedimiento de prueba	1. Ingresar a esta página web dándole el *url* directo desde el navegador. Comprobar que el sistema remite al consultante a la página de Bienvenida. 2. Hacer diferentes tipos de refinación de búsqueda. Comprobar que el sistema muestra el resultado de la búsqueda solicitada. 3. Seleccionar "Refinar Consulta". Comprobar que el sistema lo lleva a la página de Selecciona Consulta. 4. Efectuar una consulta que asegure no retornar información de la base de datos y comprobar que el sistema muestra una página de aviso donde se indica que no hubo resultados con el criterio ingresado. 5. Dejar pasar un tiempo para comprobar que el sistema lo remite a la página de Ingresar al Sistema, en caso que haya caducado su sesión.

Tabla 5.26

5.10.4.6 Caso de prueba "Salir del sistema"

Entrada	Oprimir la opción Abandonar Sesión desde la ventana principal del sistema
Resultante	Se despliega una ventana donde se confirma que el usuario acaba de cerrar su sesión
Condiciones	Ninguna
Procedimiento de prueba	1. Ingresar a esta link y verificar que se presente la página que indica que el usuario acaba de abandonar la sesión. 2. Intentar seguir utilizando la aplicación, ésta debe enviarle a la página de Ingresar al

	Sistema para seleccionar el tipo de consultante.

Tabla 5.27

5.11 Conclusiones del capítulo V

En la fase de estrategia se definieron a groso modo grandes bloques integrales del sistema completo, la forma en la que se piensa realizar el desarrollo y ciertas medidas preeliminares que servirán de base para el resto del desarrollo.

En la fase de planeación se presentan los artefactos que TSPi propone y se realizan estimados de tamaños y tiempos que se intentarán cumplir a lo largo del desarrollo.

Finalmente, en la fase de requerimientos, se presentan los casos de uso mas representativos y sus realizaciones están fuertemente influenciadas por la tecnología JAVA, que ofrece los elementos suficientes para el adecuado desarrollo de los mismos.

Como consecuencia de una descripción adecuada de los casos de uso, obtenemos las tablas de pruebas del sistema organizadas por casos de uso, en donde incluso se propone el procedimiento de prueba.

6

Construcción

En este capítulo se exponen las fases de Análisis, Diseño, Implementación y Pruebas. La discusión correspondiente a las pruebas está dividida en dos partes. La primer parte se encuentra inmersa en la fase de Implementación, ya que se recomienda (vía un HRT) realizar codificaciones y pruebas del código recién generado de manera inmediata. La segunda parte se refiere al conjunto de pruebas dedicadas a verificar aspectos no funcionales del sistema.

La fase de Análisis no pertenece a TSPi. Tal fase es añadida a este proceso de desarrollo vía una HRT, que se presentará mas adelante y que justifica esta inclusión.

A continuación, se presenta una breve descripción de cada fase en este flujo de trabajo y adicionalmente, se menciona el conjunto de actividades a realizar en cada una de ellas:

Fase de Análisis
- Objetivo:
  - Modelar las reglas de negocio del software que se desarrollará sin incluir detalles técnicos o de implementación.
- Actividades:
  - Revisar requerimientos de software del sistema (Documento SRS)
  - Generar diseño de alto nivel (Documentos HLD)
  - Producir documentación del usuario

Fase de Diseño
- Objetivo:
  - Refinar y extender los diagramas generados en la fase de Análisis incluyendo detalles técnicos con el fin de facilitar la implementación de código en la siguiente fase.
- Actividades:
  - Revisar diseño de alto nivel (Documentos HLD)
  - Generar diseño detallado (Documentos DLD)
  - Generar plan de pruebas del sistema

Fase de Implementación
- Objetivo:
  - Codificar el software solicitado.
- Actividades:
  - Revisar diseño detallado (Documentos DLD)
  - Planear tareas individuales de implementación
  - Codificar, probar y corregir
  - Producir manuales técnicos

Fase de Pruebas del sistema
- Objetivo:
  - Verificar aspectos no funcionales del sistema en general
- Actividades:
  - Realizar pruebas de sistema, corregir pruebas de sistema

Tabla 6.1

La distribución anterior de actividades se debe en gran medida a la inclusión de la fase de Análisis al inicio del flujo de trabajo. Esta distribución queda formalmente establecida vía la siguiente HRT:

Referencia: 0015	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Flujo / Fase / Ciclo:	Construcción. Ciclo I
Fecha de creación:	Lunes 14 de Enero de 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	La inclusión de una nueva fase cambia la secuencia de las siguientes fases.
Propuesta:	Reorganizar la secuencia de actividades de las siguientes fases.
Justificación:	Algunas actividades propuestas en la nueva fase ya estaban consideraras en la de diseño, lo que provoca un corrimiento en cascada de otras actividades en fases posteriores.
Estrategia:	Integrar las actividades de cada fase de acuerdo a la secuencia presentada en la tabla 6.1
Beneficios:	- Mejor organización de las actividades a realizar por fase.
Desventajas:	- Inversión de tiempo al asimilar ésta nueva propuesta
Impacto en calidad:	No aplica
Tecnologías Involucradas:	Ninguna.
Responsable de ejecución:	MEM (Líder de proyecto)
Aprobación:	_____ _____ _____ _____

Tabla 6.2

6.1 Análisis

Como se mencionó al inicio de este capítulo, la fase de Análisis no pertenece al proceso TSPi, que es la base del presente documento. Su inclusión queda justificada a través de la siguiente HRT:

Referencia: 0016	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Flujo / Fase / Ciclo:	Construcción. Ciclo I
Fecha de creación:	Lunes 14 de Enero de 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	La lógica de negocio a ser implementada (que define qué se debe hacer) es independiente de las actividades del diseño detallado (que explican técnicamente cómo se debe hacer) sin embargo, ambos aspectos son complementarios.
Propuesta:	Integrar una nueva fase al flujo de construcción.
Justificación:	- De esta manera habrá una fase dedicada únicamente a la definición de la lógica de negocio sin incluir detalles técnicos o tecnológicos. - Servirá como base o guía para la fase de diseño. - Los documentos resultantes definen el manual para el usuario del sistema, mapeando la lógica de negocio con la funcionalidad del sistema.
Estrategia:	Integrar la fase de análisis de acuerdo a la siguiente secuencia: - Revisar requerimientos de software del sistema (SRS). - Generar diseño de alto nivel (es factible la generación de documentos UML para este efecto). - Producir documentación del usuario.
Beneficios:	- Separación de la lógica de negocio de los detalles técnicos. - Guía para la fase de diseño. - Guía para la generación de manuales de operación.
Desventajas:	- Mayor inversión de recursos de administración - Mayor inversión de tiempo

Impacto en calidad:	- Mantenible. Un preciso diseño de alto nivel facilita la localización de componentes a mantener. - Funcional. En la fase de análisis se modela de manera concreta (pero sin abordar detalles técnicos) la funcionalidad del sistema
Tecnologías Involucradas:	Ninguna.
Responsable de ejecución:	MEM (Líder de proyecto)
Aprobación:	_____ _____ _____ _____

Tabla 6.3

Mas en detalle, en la fase de análisis se propone generar documentos que modelen, a través de un lenguaje común para el equipo de desarrollo (en este caso se sugiere UML), los requerimientos definidos en el flujo de trabajo anterior.

Los documentos generados en esta fase deben explicar de manera clara y sin ambigüedades las reglas de negocio que el software que se desarrolla debe obedecer. Estos documentos no deben incluir información ajena al negocio, o especificaciones técnicas para su implementación.

Los documentos producto de la fase de Análisis podrán, en su momento, ser presentados al cliente y éste deberá ser capaz de entenderlos dada su naturaleza no técnica.

Adicionalmente, a partir de este momento, cada documento deberá formar parte del control de cambios, de manera tal que a éstos se les pueda dar seguimiento y continuidad a lo largo del desarrollo del producto.

Los puntos a desarrollar en las siguientes secciones son:

6.1.1 Revisión del documento de requerimientos del sistema.
6.1.2 Generar diseño de alto nivel.
6.1.2.1 Generar diagramas de actividades.
6.1.2.2 Identificación de clases de Análisis.
6.1.2.3 Generación de diagramas de clases y paquetes.
6.1.2.4 Generación de diagramas de secuencia.
6.1.3 Producir documentación de usuario.

6.1.1 Revisión del documento de requerimientos del sistema

El primer punto a realizar es la revisión de los requerimientos de software del sistema. A partir de este punto se presentarán los documentos resultantes para el caso de estudio presentado en este trabajo.

Con base en el conjunto de requerimientos se tiene el siguiente resumen de funcionalidades:

- Presentar una forma de bienvenida al acceder al URL de la aplicación.
- Solicitar una autentificación para el ingreso al sistema.
- Permitir una caracterización genérica o un registro detallado.
- Ofrecer la elección de un tipo de consulta de entre seis posibles:
  - ✓ Personal académico
  - ✓ Líneas de investigación
  - ✓ Dependencias universitarias
  - ✓ Proyectos de investigación
  - ✓ Servicios
  - ✓ Equipos
- Permitir refinar una consulta.
- Permitir obtener más información acerca de un resultado específico (es decir, poder acceder a información general a partir de un resultado específico).

Lo anterior es la base para la generación de los documentos de alto nivel, ya que ofrece una guía a seguir para la generación de los mismos.

6.1.2 Generar diseño de alto nivel

Se deberá generar una serie de diagramas que modelen los requerimientos anteriormente expuestos. Cabe hacer notar que UML proporciona nueve diagramas; sin embargo, no es necesario generar cada uno de ellos en cada fase. En el presente trabajo no se incluirán, para ninguna fase, los diagramas de objetos ni los de colaboración. La secuencia de creación propuesta para los diagramas en esta fase será la siguiente:

- Diagramas de actividades y estados
- Diagramas de clases
- Diagramas de paquetes
- Diagramas de secuencia

6.1.2.1 Diagramas de actividades y estados

Estos diagramas definen un flujo lógico de negocio basado en las actividades que se llevan a cabo en la operación regular de la organización e incluyen posibles estados a los que se llega vía alguna actividad. Existen referencias (por ej., UML Distilled[9]) que recomiendan la generación independiente de diagramas de actividades y de estados. Sin embargo, la herramienta para diagramación que se utiliza en el presente trabajo no hace distinción entre éstos. El siguiente diagrama de actividades presenta el flujo general de la aplicación **SIPU©**, que es el caso de estudio expuesto en este trabajo:

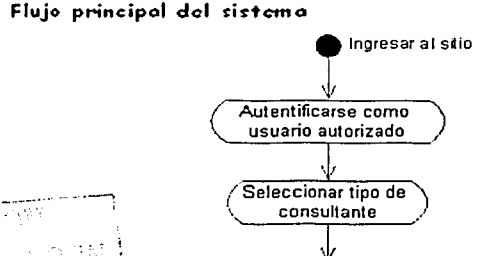


Figura 6.1 (Parte 1)

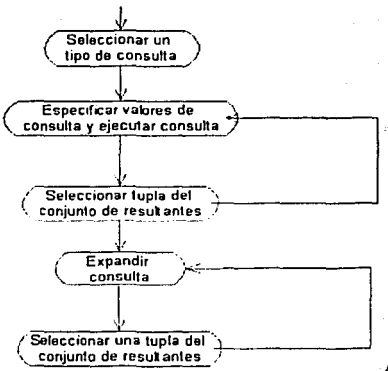


Figura 6.1 (Parte II)

El diagrama anterior se descompone, a su vez, en los siguientes subdiagramas:

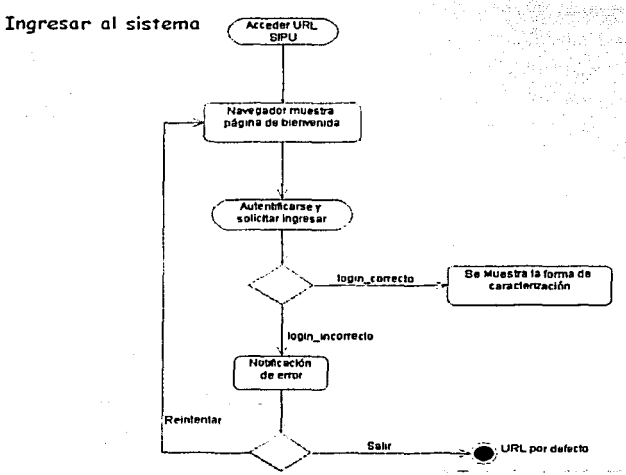


Figura 6.2

La figura 6.2 muestra el diagrama de actividades para la actividad "Ingresar al sistema", correspondiente al caso de uso del mismo nombre. Aquí se pueden apreciar dos reglas de negocio importantes:

a) El sistema permitirá reintentar la autentificación de un usuario un número ilimitado de veces hasta que tal usuario decida salir voluntariamente o ingrese al sistema con la clave correcta.
b) Una vez autenticado, el usuario deberá caracterizarse de acuerdo a la forma que se lo solicitará posterior a su correcta autenticación.

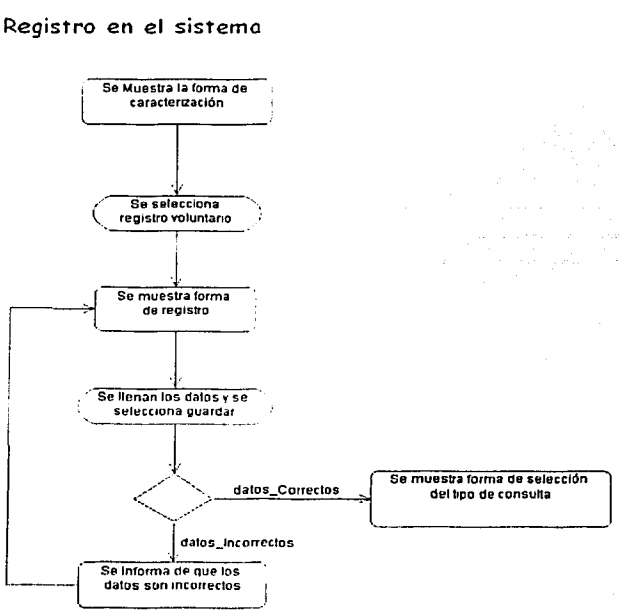


Figura 6.3

La figura 6.3 muestra otro caso de uso, en donde queda diagramada una regla de negocio más: El sistema deberá ofrecer la posibilidad de que un usuario elija, de manera voluntaria, si desea registrarse detalladamente o no. Si el usuario desea realizar su registro, el sistema deberá validar la información dada y presentar un mensaje de error en caso de que ésta sea errónea. Si los datos son correctos, el sistema deberá almacenarlos y continuar con la forma de selección del tipo de consulta.

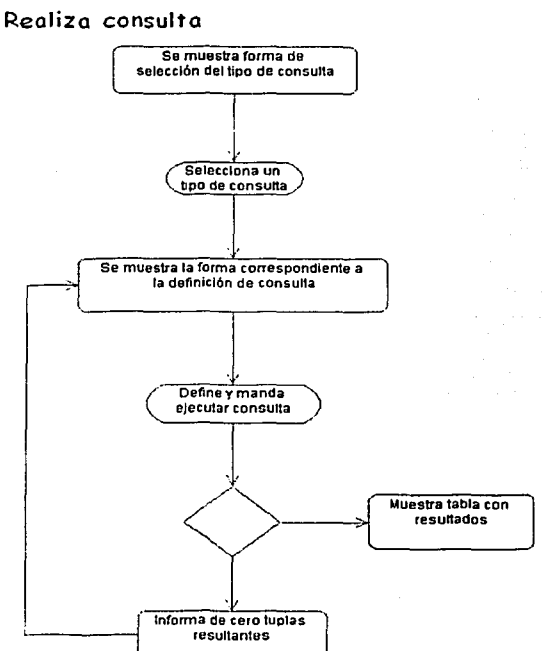


Figura 6.4

En la actividad "Realizar consulta" representada por la figura 6.4, se presenta una secuencia básicamente lineal, salvo el caso en el que el conjunto de resultados es vacío. En este caso, la regla de negocio indica que se debe informar al consultante de tal situación y a continuación, reenviarlo a la forma de definición de consulta correspondiente.

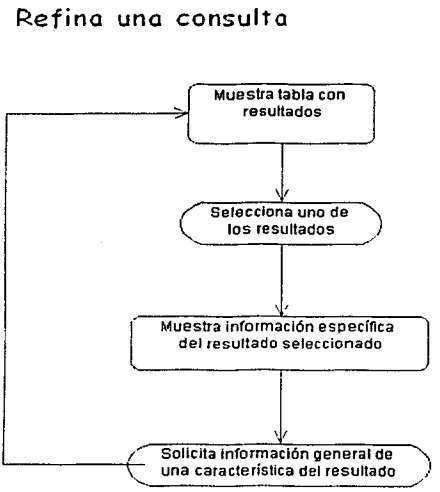


Figura 6.5

Finalmente, se presenta el modelo para los casos de uso "Refina consulta" y "Expande consulta". Posterior a la ejecución de una consulta, el sistema mostrará un conjunto no vacío de resultantes a la misma. El consultante podrá seleccionar un elemento de este conjunto y el sistema mostrará información detallada acerca del elemento. Esta información detallada siempre contendrá ligas a más información y de esta manera será posible expandir consultas.

6.1.2.2 Generación de diagramas de clases

Una de las subactividades más importantes de la fase de Análisis, consiste en identificar y organizar clases en los siguientes tres tipos:

a) Clases de tipo *Boundary*. Son aquellas clases que sirven como interfaz humana y visual entre el sistema y el usuario del mismo.
b) Clases de tipo *Entity*. Son aquellas clases relacionadas con procesos de persistencia o almacenamiento y que generalmente mapean entidades (o tablas) en una base de datos.
c) Clases de tipo *Control*. Son aquellas que contienen la lógica del funcionamiento del sistema, así como la del flujo y control de errores.

En este apartado, se identificarán (y clasificarán) un gran número de clases para nuestro caso de estudio y adicionalmente, se generarán diagramas que muestren las relaciones entre ellas.

A continuación, se presenta la identificación y clasificación antes mencionada:

Clases de tipo *Entity* identificadas:

1. InvestigadoresMD
2. LineasInvestigaciónMD
3. DependenciasMD
4. ProgramasUniversitariosMD
5. EquiposMD
6. PublicacionesMD

Clases de tipo *Boundary* identificadas:

1. Bienvenida
2. Ingreso
3. Error
4. CaracterizacionRegistro
5. SeleccionConsulta
6. DefConsInvestigadores

7. DefConsLineasInvestigacion
8. DefConsDependencias
9. DefConProgramasUniversitarios
10. DefConsEquipos
11. CeroTuplas
12. ResInvestigadores
13. ResLineasInvestigacion
14. ResDependencias
15. ResProgramasUniversitarios
16. ResEquipos
17. DetalleUnInvestigador
18. DetalleUnProgramaUniversitario
19. DetalleUnDependencia
20. DetalleUnEquipo
21. DetalleUnLineaInvestigacion

Clases de tipo *Control* identificadas:

1. ControlDeErrores
2. AutentificadorDeUsuarios
3. LogicaDeFlujo
4. ValidadorDeInformacion

En la siguiente fase, la de diseño, varias de las anteriores clases se dividirán, otras serán extendidas y otras desaparecerán. La organización anterior podrá cambiar y muy probablemente esto ocurra. Esto es debido principalmente a que en la fase de diseño las complejidades técnicas de implementación y la influencia de aspectos tecnológicos modifican la perspectiva de negocio pura.

Por el momento, con la organización que se tiene es posible generar un diagrama de clases que muestre la dependencia entre clases y posteriormente, un diagrama de paquetes. El diagrama de clases es expuesto en la figura 6.6:

Diagrama general de clases

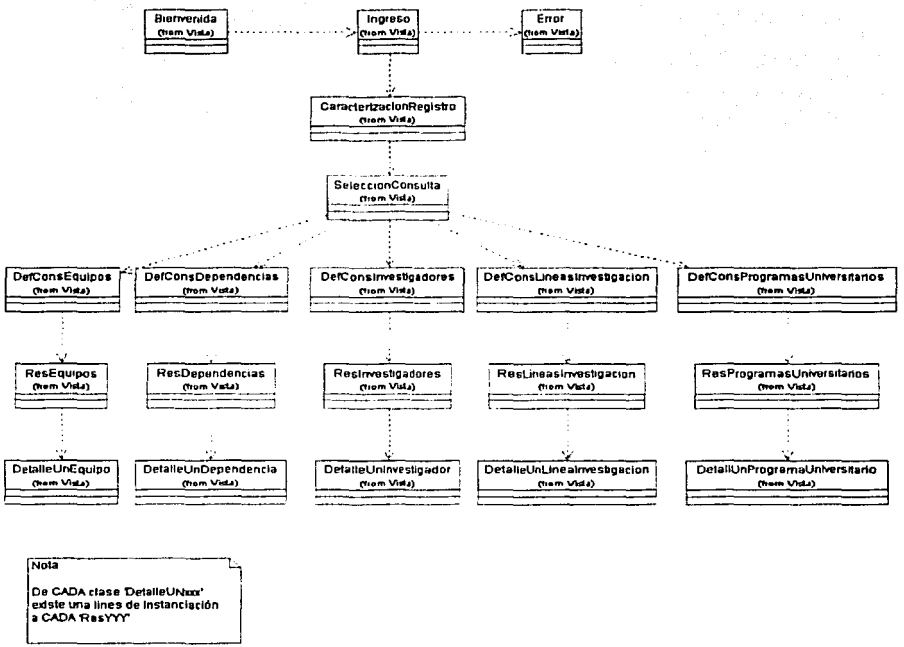


Figura 6.6

6.1.2.3 Generación de diagramas de paquetes

Los diagramas de paquetes nos ayudan a clasificar clases en categorías, generalmente agrupándolas por funcionalidad o misión común. Dentro de cada paquete es factible definir subpaquetes con clases que realizan funcionalidades de propósito común.

Los tres grandes paquetes base, están definidos por la arquitectura propuesta: El paquete de la vista, el del modelo y el del controlador.

A continuación sus diagramas:

Modelo	Vista	Controlador
AutentificadorDeUsuarios ValidadorDeInformacion InvestigadoresMD LineasInvestigaciónMD DependenciasMD ProgramasUniversitariosMD EquiposMD PublicacionesMD	Bienvenida Ingreso Error CaracterizacionRegistro SeleccionConsulta DefConsInvestigadores DefConsLineasInvestigacion DefConsDependencias DefConProgramasUniversitarios DefConsEquipos CeroTuplas ResInvestigadores ResLineasInvestigacion ResDependencias ResProgramasUniversitarios ResEquipos DetalleUnInvestigador DetalleUnProgramaUniversitario DetalleUnDependencia DetalleUnEquipo DetalleUnLineaInvestigacion	ControlDeErrores LogicaDeFlujo

Tabla 6.4

6.1.2.4 Generación de diagramas de secuencia

Los diagramas de secuencia pueden ser definidos como una vista que modela el comportamiento dinámico de un sistema.

A continuación se presentan los diagramas que modelan las dinámicas de nuestro caso de estudio.

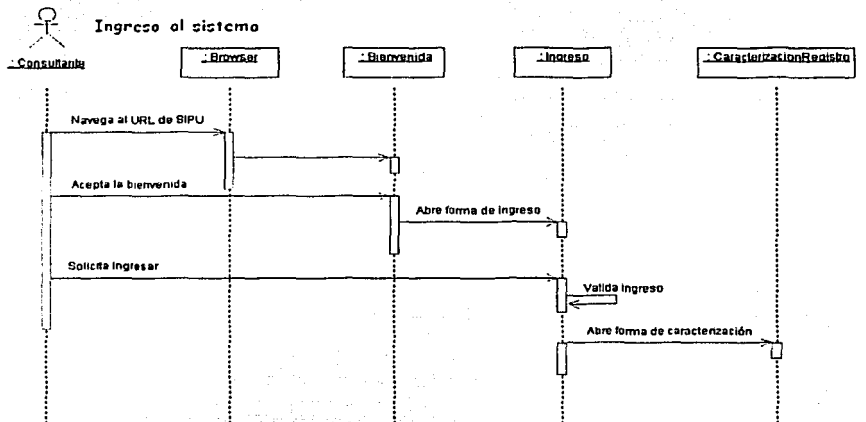


Figura 6.7

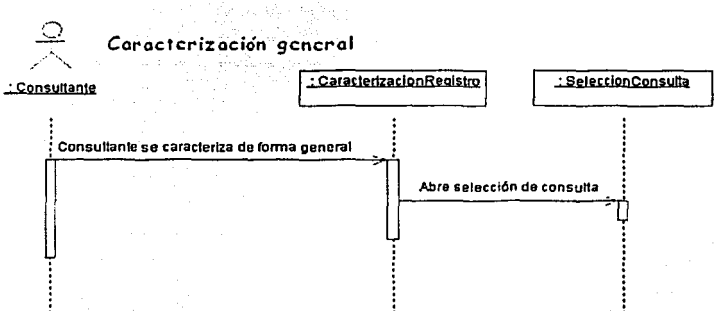


Figura 6.8

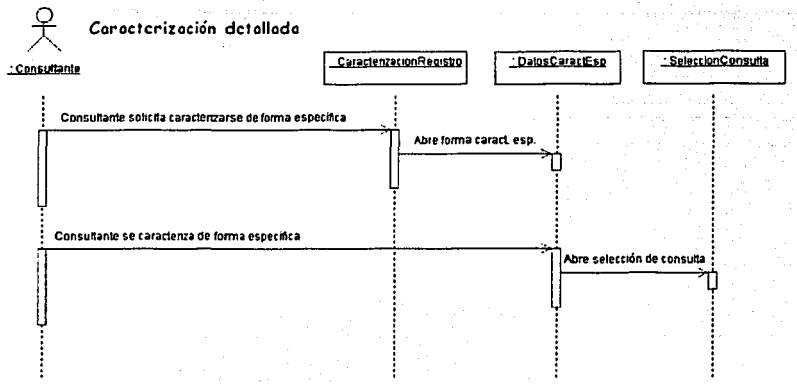


Figura 6.9

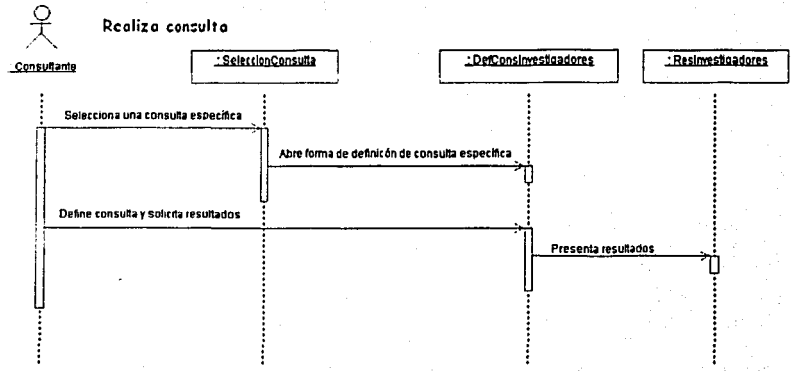


Figura 6.10

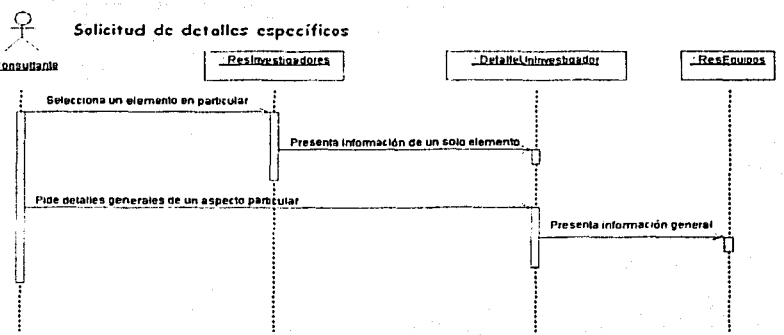


Figura 6.11

6.1.3 Producir documentación de usuario

En este punto del proceso de desarrollo es ya posible generar documentación para el usuario del sistema. La mayor parte de la documentación proviene de los modelos que se han definido previamente. Esta documentación es un nexo (o mapeo) entre la funcionalidad del sistema y la forma en la que la organización opera y debe ser una guía para mostrar cómo ciertos procesos de negocio pueden ser llevados a cabo vía el sistema.

En este punto conviene incluir el prototipo (aún no funcional) desarrollado en la fase de requerimientos. Éste además de ser un esbozo de la interfaz de usuario, contendrá una descripción de la lógica de navegación de la misma y notas o recomendaciones acerca de cómo usarla.

La documentación de usuario asociada al presente caso de estudio se incluye en el apéndice A de este trabajo.

6.2 Diseño

La fase de diseño comprende la generación de documentos que faciliten la implementación del código de un sistema. Diversos diagramas (en UML) son generados y la mayoría de éstos están basados en los que se generaron en la fase de análisis, pero incluyendo ahora la arquitectura del sistema; es decir, considerando aspectos técnicos independientes de la lógica de negocio a implementar.

Cabe mencionar que el diseño depende en mucho de los elementos tecnológicos de los que se dispone, contrariamente a lo que sucede en la fase de análisis.

En la fase de diseño se detallan más (a nivel técnico) los aspectos relacionados con la forma en la que se va a codificar cierto requerimiento. Hasta antes de ésta fase, se ha modelado el sistema con un alto nivel de abstracción y ello promueve la independencia de ambiente, plataforma y lenguaje, sin embargo, y como consecuencia natural de esto, tal modelo no aporta los elementos suficientes para la adecuada implementación (en términos de código) del producto. Un aspecto importante es el hecho de que en esta fase las clases se exponen con atributos, métodos y un indicador de alcance[1] de los mismos.

En esta fase, se extenderán y refinarán clases identificadas en la fase de Análisis. Algunas clases serán excluidas y en otros casos, reemplazadas por dos o más clases. Serán propuestas nuevas clases de soporte no visibles desde la óptica del consultante o desde la lógica de negocio pura.

El volumen de documentación generada en la fase de diseño es muy grande. Es por ello que en este trabajo, para la presente fase, sólo se incluirán los documentos necesarios para ejemplificar el concepto a tratar y no el total de la documentación asociada[2].

[1] Cuando hablamos de un indicador de alcance, nos referimos a los modificadores de los métodos y propiedades de una clase, como lo son public, private, protected, static, syncronyzed y final.

[2] En caso de requerir mayor información al respecto, es posible consultar la carpeta de desarrollo de **SIPUD** que se encuentra en formato impreso y adicionalmente, en formato digital en el disco compacto incluido en este trabajo.

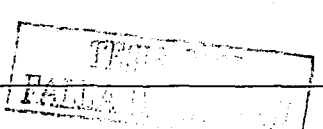
Al igual que en la fase anterior, se presenta ahora un resumen de las actividades a realizar en esta fase:

6.2.1 Revisar diseño de alto nivel
6.2.2 Generar diagramas de diseño
  6.2.2.1 Identificar nuevas clases
  6.2.2.2 Generar diagramas de clases
  6.2.2.3 Generar diagramas de paquetes
  6.2.2.4 Generar diagramas de actividades
  6.2.2.5 Generar diagramas de secuencia
6.2.3 Generar plan de pruebas de integración

6.2.1 Revisar diseño de alto nivel

En esta sección el objetivo es –posterior a la revisión del diseño de alto nivel- definir cuales serán las adiciones y modificaciones a los documentos generados en la fase de Análisis con el fin de facilitar su codificación en la fase Implementación. A continuación se presenta un plan general de cuatro puntos útiles para la realización de este propósito:

i. Detallar cada una de las clases definidas en la fase anterior añadiéndoles métodos y propiedades que requieren para interactuar con las demás clases y dotarlas de funcionalidad por sí mismas.
ii. Se generarán nuevas clases que servirán de soporte a bajo nivel:
   - Servicios de conexión a la base de datos.
   - Registro de transacciones.
   - *Listener* de sesiones.
   - Arrancador automático de servicios.
   - Utilerías de conversión de cadenas SQL entre distintos DBMS's.
   - Controladores de la lógica navegacional.
   - Utilerías de generación de objetos HTML dinámicamente generados.
   - Control y manejo de errores.
iii. Se generará un diagrama detallado de la organización de las entidades en la base de datos.



iv. Implementar un archivo de propiedades que contenga información global a usar a lo largo de toda la aplicación.

Para nuestro caso de estudio, las propuestas realizadas en la fase de Análisis mas las adiciones anteriores serán la base para la codificación del sistema en la siguiente fase. Cabe mencionar que en este caso (caso de estudio SIPU) no fue necesario eliminar elementos propuestos en la fase de Análisis.

En el plan generado en la revisión del diseño de alto nivel, cada uno de los rubros del segundo punto tiene una justificante que es presentada a través de una HRT. Éstas son presentadas a continuación.

Referencia: 0017	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	El acceso concurrente de una gran cantidad de usuarios remotos al sistema requiere de un manejo adecuado de los recursos de conexión a la base de datos.
Propuesta:	Implementar una clase administradora de objetos de tipo *Connection* a la base de datos.
Justificación:	Cada sesión debe utilizar de manera eficiente un número finito de conexiones a la base de datos ya que no es factible crear y destruir objetos de este tipo sin control alguno, debido a su elevado costo en términos del consumo de recursos del sistema.
Estrategia:	Codificar una clase contenedora de objetos de tipo conexión basada en el patrón *singleton* e inicializada al

	inicio del aplicativo (vía el arrancador definido en el HRT 19) cargando un número fijo de conexiones y reciclando su uso.
Beneficios:	• Gran control en el uso de recursos de tipo conexión. • Posibilidad de atención a un gran número de transacciones con un bajo número de conexiones. • Mejor administración y seguimiento en el posible registro de transacciones a la base de datos.
Desventajas:	• Alta complejidad de implementación. • Imposibilidad de realización de transacciones ACID con clientes remotos.
Impacto en calidad:	• Desempeño. • Portabilidad. • Reutilizabilidad. • Escalabilidad.
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.5

Referencia: 0018	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA

Origen:	No se posee un registro de operación del sistema
Propuesta:	Generar una clase para el registro de operación del sistema
Justificación:	Con la implementación de esta clase se incrementa la facilidad del análisis del uso del sistema y puede ser utilizada para realizar procesos de pruebas del mismo.
Estrategia:	Generar un componente que permita escribir en un archivo plano información referente a los distintos eventos, notificaciones, y alertas o errores del sistema. Adicionalmente, será posible utilizar este componente para escribir mensajes genéricos al archivo plano.
Beneficios:	• Facilita la administración y seguimiento del uso del sistema • Apoyo en el proceso de pruebas de código
Desventajas:	Inversión de tiempo en la generación del componente
Impacto en calidad:	• Seguridad • Mantenibilidad
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.6

Referencia: 0019	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de	Viernes 1 de Febrero 2002

creación:	
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	Diversos servicios deben estar disponibles en todo momento, aún sin que exista actividad por parte del cliente
Propuesta:	Implementar un arrancador de servicios
Justificación:	Ciertos servicios que pueden ser levantados 'en demanda' toman demasiado tiempo en su primera ejecución. Por ello, conviene inicializarlos al inicio del aplicativo. Otros servicios deben existir en todo momento y en particular, al iniciar el aplicativo.
Estrategia:	Codificar una clase basada en el patrón *singleton* que será instanciada al inicio del aplicativo por el contenedor del mismo.
Beneficios:	• Disponibilidad de ciertos servicios en todo momento • Mejor desempeño en la atención de peticiones de servicios específicos
Desventajas:	• Inversión de tiempo en codificación. • Complejidad de implementación • Alto consumo de tiempo de proceso al iniciar por primera vez
Impacto en calidad:	• Desempeño • Funcionalidad
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA

Aprobación:	_____ _____

Tabla 6.7

Referencia: 0020	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	La aplicación debe poder ser utilizada con una gran variedad de manejadores de bases de datos
Propuesta:	Crear una clase convertidora de cadenas SQL
Justificación:	La creación de una clase que traduce cadenas entre distintos manejadores de bases de datos permitirá cambiar de manejador sin alterar el código de aplicativo.
Estrategia:	Crear una clase que detecte (vía el archivo de propiedades definido en el HTR 24) el manejador usado y que traduzca adecuadamente las sentencias SQL para que sean entendidas por tal manejador.
Beneficios:	Posibilidad de cambiar de manejador sin impacto al código del aplicativo.
Desventajas:	• El código del traductor deberá ser extendido para cada manejador nuevo que se desee soportar. • Cambios en la sintaxis del SQL de un manejador implican cambios en el código de esta clase traductora

Impacto en calidad:	Portabilidad
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.8

Referencia: 0021	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	El aplicativo está basado en el patrón arquitectónico MVC
Propuesta:	Codificar controladores para la lógica navegacional
Justificación:	Los controladores (según el patrón MVC) tienen la responsabilidad de coordinar la presentación del *layout* adecuado
Estrategia:	Identificar y codificar clases controladoras
Beneficios:	Apego al patrón propuesto
Desventajas:	N/A
Impacto en calidad:	Modularidad

Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.9

Referencia: 0022	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	La aplicación utiliza gran cantidad de objetos cuyo contenido varía en función de la información almacenada en la base de datos
Propuesta:	Implementar utilerías de generación de objetos HTML dinámicamente generados.
Justificación:	Las tareas de actualización de información para objetos cuyo contenido es dinámico sólo pueden ser llevadas a cabo a través de procesos automáticos de carga de información.
Estrategia:	Implementar una clase de utilerías que contenga un método de generación de cadenas con la estructura de objetos HTML, incluyendo el contenido de los mismos. Se recomienda un método para objetos de tipo *comboBox* y otro para objetos de tipo *table*.

Beneficios:	Técnica que ahorrará tiempo de codificación, uniformidad en el desarrollo y modularidad al mismo.
Desventajas:	• Mayor inversión de tiempo de implementación y recursos de proceso por parte del servidor. • Mayor demanda en peticiones a la bases de datos
Impacto en calidad:	• Mantenibilidad • Escalabilidad
Tecnologías Involucradas:	HTML, JAVA
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.10

Referencia: 0023	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	El sistema despliega una gran variedad de avisos al usuario y al programador.
Propuesta:	Implementar una clase para el manejo, control y despliegue de errores.
Justificación:	De esta manera se tendrá una mayor y mejor organización del control de errores para su detección, corrección y

	seguimiento.
Estrategia:	Crear una clase que reciba objetos de tipo *Exception* y que les de tratamiento, permitiendo realizar un desplegado agradable en pantalla o deteniendo la ejecución del aplicativo. Adicionalmente, esta clase podrá comunicarse con la clase del registro de transacciones y podrá opcionalmente guardar información.
Beneficios:	El sistema se hará mas entendible debido a la forma en que se desplegarán los mensajes y adicionalmente será mas fácil de mantener, dada la modularidad de esta clase.
Desventajas:	Mayor inversión en el tiempo de codificación de esta clase.
Impacto en calidad:	Mantenibilidad Usabilidad
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.11

Referencia: 0024	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha limite para ejecución:	De inmediato

Autor:	GAA
Origen:	El aplicativo debe regirse con base en un conjunto de parámetros fijos.
Propuesta:	Implementar un archivo que contenga la definición de tales parámetros.
Justificación:	Dado que los parámetros a utilizar serán fijos a lo largo de la vida de la aplicación y que estos pueden sufrir cambios sin alterar el código del sistema, la mejor alternativa es la implementación de un archivo plano que contenga estos parámetros.
Estrategia:	Crear un archivo plano que contenga parejas del tipo nombre_parámetro, valor_parámetro con información a utilizar en el aplicativo.
Beneficios:	Facilidad de realización de cambios a los parámetros generales del sistema desde un solo archivo central.
Desventajas:	Se incrementa el nivel de abstracción del código que utiliza valores del archivo de parámetros.
Impacto en calidad:	• Mantenibilidad • Escalabilidad
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.12

Referencia: 0024	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	Cada sesión debe ser identificada por un indicador fácilmente manejable y el contenedor de JSP's ofrece uno muy complejo.
Propuesta:	Implementar un generador de identificadores de sesión automático
Justificación:	Con esta propuesta se satisface el origen del problema.
Estrategia:	Codificar una clase generadora de identificadores de sesión, que le serán asignados a cada nueva sesión y dados de baja al término de esa sesión para que luego sean reutilizados.
Beneficios:	Obtención de un identificador único de sesiones manejable y reutilizable.
Desventajas:	Mayor inversión en el tiempo de codificación de esta clase.
Impacto en calidad:	N/A
Tecnologías Involucradas:	Java
Responsable de ejecución:	GAA
Aprobación:	

Tabla 6.13

Como es de esperar, la inclusión de estas clases al modelo del sistema influye en la definición de los nuevos diagramas de diseño. De esta manera, los diagramas del Diseño serán extensiones de los del Análisis, permaneciendo estos últimos intactos o con muy pocas modificaciones. En la siguiente sección se presentan ejemplos de tal extensión.

6.2.2 Generar diagramas de diseño (Diseño detallado o DLD)

Los diagramas de diseño a generar son básicamente del mismo tipo que en el análisis, sin embargo, en ésta fase también se generarán los diagramas de instalación. A continuación se presenta (a través de tablas semánticamente equivalentes a los diagramas de UML para la definición de clases) la extensión al conjunto de clases identificadas en la fase de Análisis. El formato de la presentación propuesta se formaliza a través de la HRT 0025.

Clases pertenecientes al paquete de interfaz humana.

Los siguientes métodos (ejemplificados con la exposición de la clase SeleccionaConsulta) deben ser implementados para cada una de las clases pertenecientes al paquete de interfaz humana:

SeleccionConsulta

Descripción: Clase que permite seleccionar, de entre un conjunto de opciones disponibles, el tipo de consulta que se desea ejecutar.

Paquete: IH
Ancestro: N/A
Implementa: N/A
Parámetros públicos:
Void
Parámetros privados:
Void
Métodos públicos:
Constructor

	Parámetros	
	Void	
open		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	Void	
build		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	Void	
submit		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	Void	

Tabla 6.14

De la misma manera que en la tabla anterior, a continuación se presentan los métodos asociados a cada una de las clases correspondientes al manejo de datos ejemplificado con la clase 'InvMD'.

InvMD	
Descripción: Clase que permite la inserción, actualización, eliminación y búsqueda de la información referente a la tabla Investigadores de la base de datos.	
Paquete:	MD
Ancestro:	BaseMD
Implementa:	N/A

Parámetros públicos:		
	Void	
Parámetros privados:		
	sesionId	alfanumérico
	bolsa	vector
Métodos públicos:		
Constructor		
	Parámetros:	
	cnn	Conexión
	idp	InvDP
Insert		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	Void	
update		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	Void	
delete		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	Void	
find		
	Alcance:	público
	Valor de retorno:	InvDP
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	llave	entero

Tabla 6.15

Resta presentar la definición de las nuevas clases propuestas al inicio de la fase. Debido a la extensión y número de éstas, sólo serán presentados algunos ejemplos que aclaren el concepto. Si se desea obtener una referencia completa, es posible consultar la carpeta de desarrollo del proyecto **SIPU©**.

PoolConnection

Repositorio de objetos de tipo conexión (que ofrece en demanda) y que recicla en su uso. Permite la ejecución de transacciones ACID y soporta la petición concurrente de un número fijo máximo de usuarios de la clase.

Paquete: tools
Ancestro: N/A
Implementa: N/A
Parámetros públicos:

MAX_ALOWED	entero
TIMEOUT	entero largo

Parámetros privados:

bolsa	vector

Métodos públicos:

Constructor

Parámetros	
Void	

init

Alcance:	público
Valor de retorno:	void
Método de clase:	NO
Sincronía:	NO
Arroja excep:	NO
Parámetros:	
numConn	entero
dbms	alfanumérico
drv	alfanumérico
usr	alfanumérico
psw	alfanumérico

getPoolConnection

Alcance:	público
Valor de retorno:	Conexión
Método de clase:	NO
Sincronía:	SI
Arroja excep:	NO
Parámetros:	

	abc	alfanumérico
returnPoolConnection		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	SI
	Arroja excep:	SI
	Parámetros:	
	cnn	Conexión
pushCnn		
	Alcance:	público
	Valor de retorno:	void
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	idSesion	entero
	cnn	Conexión
popCnn		
	Alcance:	público
	Valor de retorno:	Conexión
	Método de clase:	NO
	Sincronía:	NO
	Arroja excep:	NO
	Parámetros:	
	idSesion	entero

Tabla 6.16

SessionListener

Clase que lleva el control de las sesiones que acaban de ser creadas por peticiones remotas y que destruye recursos utilizados por estas sesiones cuando ya no son requeridos de manera automática.

Paquete: tools
Ancestro: HttpServlet
Implementa: HttpSessionListener
Parámetros públicos:

MAX_ALOWED	entero
TIMEOUT	entero largo

Parámetros privados:

sesionId	alfanumérico
bolsa	vector

Métodos públicos:

Constructor

Parámetros: Void

getSubID

Alcance:	público
Valor de retorno:	alfanumérico
Método de clase:	NO
Sincronía:	NO
Arroja excep:	NO

Parámetros: Void

returnSubID

Alcance:	público
Valor de retorno:	Void
Método de clase:	NO
Sincronía:	NO
Arroja excep:	NO

Parámetros:

id	alfanumérico

getCounterInfo

Alcance:	público
Valor de retorno:	alfanumérico
Método de clase:	SI
Sincronía:	NO
Arroja excep:	NO

Parámetros: Void

secuencial

Alcance:	público
Valor de retorno:	entero largo
Método de clase:	NO
Sincronía:	NO
Arroja excep:	NO

Parámetros: Void

sessionCreated

Alcance:	público
Valor de retorno:	void
Método de clase:	NO
Sincronía:	NO
Arroja excep:	NO

Parámetros:

evt	HttpSessionEvent

sessionDestroyed

Alcance:	público
Valor de retorno:	void
Método de clase:	NO
Sincronía:	NO
Arroja excep:	NO
Parámetros:	
evt	HttpSessionEvent

Tabla 6.17

El formato de presentación de la definición de las anteriores clases tiene un justificante, que es presentado a través de la siguiente HRT:

Referencia: 0025	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	Se requiere de mayor información (en términos técnicos) para el apoyo a la fase de Implementación.
Propuesta:	Generar un formato de especificación para clases.
Justificación:	Sin la necesidad de una herramienta de diagramación, se pueden generar las definiciones de las clases de manera más explícita, facilitando su posterior implementación.
Estrategia:	Realizar las definiciones de las clases a través de formatos especiales, con una estructura que permita convertirlos posteriormente en diagramas de UML.
Beneficios:	No habrá dependencia de una herramienta diagramadora. Mayor claridad en la definición de las clases.

Desventajas:	Consumo de tiempo al asimilar un nuevo formato de definición de clases.
Impacto en calidad:	N/A
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.18

De la misma forma que en la HRT anterior, a continuación se presenta otra que sugiere un formato semánticamente equivalente a los diagramas de actividades de UML y que será utilizado en la siguiente sección:

Referencia: 0026	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo 1
Fecha de creación:	Viernes 1 de Febrero 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	El tiempo de elaboración de diagramas de actividades es alto, debido a la forma en que una herramienta de diseño permite su diagramación.
Propuesta:	Generar un formato de especificación para actividades.
Justificación:	Sin la necesidad de una herramienta de diagramación, se

	pueden especificar las actividades a realizar de manera más simple, facilitando su comprensión y posterior implementación.
Estrategia:	Realizar las especificaciones de las actividades a través de formatos especiales con una estructura que permita convertirlas posteriormente en diagramas de UML.
Beneficios:	No habrá dependencia de una herramienta diagramadora. Mayor claridad en la definición de las clases. Menor consumo de espacio y mejor visualización.
Desventajas:	Consumo de tiempo al asimilar un nuevo formato de especificación de actividades.
Impacto en calidad:	N/A
Tecnologías Involucradas:	N/A
Responsable de ejecución:	GAA
Aprobación:	_____ _____ _____ _____

Tabla 6.19

Diagramas de actividades

Los diagramas de actividades en la fase de Diseño son refinamientos de los que se generaron en la fase del Análisis y su objetivo es facilitar la implementación de las mismas en la fase de Implementación. Con base en la HRT 26, a continuación se presenta una descripción del formato que representará dichos diagramas y que está dividido en dos secciones:

Sección 1 (Administrativa)

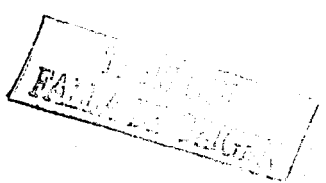
Actividad:	Número de la actividad a describir.
Procedencia:	Actividad que hace referencia a la actividad que se presenta. Si no existe procedencia, el valor a capturar es 0.
Fecha:	Fecha de elaboración
Etapa:	Fase y ciclo en que la actividad que se presenta fue elaborada.
Autor:	Acrónimo del miembro del equipo que elaboró la hoja.
Objetivo:	Descripción general. Fin con el que se elabora la actividad.

Sección 2 (Descriptiva)

Columna 1: Consecutivo de actividades
(Para cada actividad inicia en 1.1)
Columna 2: Actor que realiza la actividad
Columna 3: Descripción de la actividad
Columna 4: Referencia a la actividad descrita.
(Si no aplica, se captura "N/A")

En la tercera columna es posible utilizar las palabras reservadas IF, ELSE y THIS las dos primeras para indicar que existe una decisión. La palabra reservada THIS indicará un salto a una sub actividad del mismo diagrama.

Con el fin de ejemplificar este nuevo formato, a continuación se presenta un diagrama de actividades UML (Ingresar al sistema) y la hoja propuesta correspondiente:



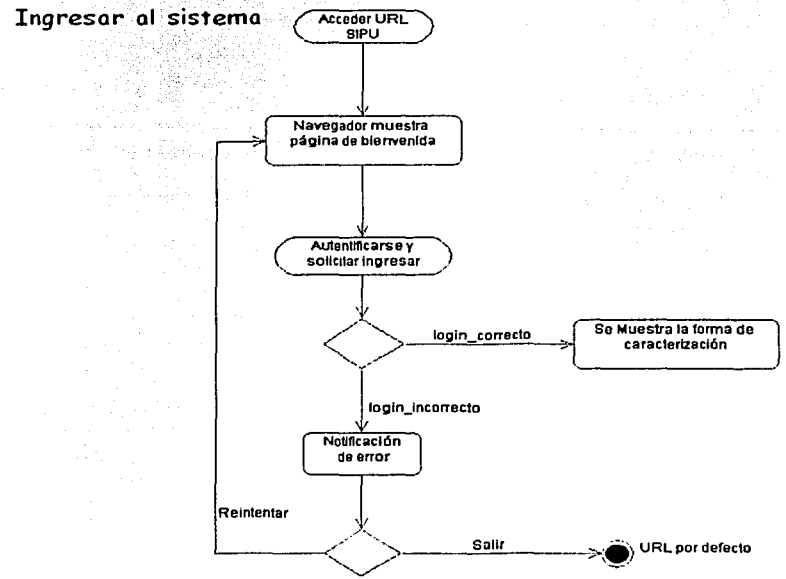


Figura 6.12

Hoja de Actividades	
Actividad:	2.1
Procedencia:	0
Fecha:	Lunes 4 de Febrero de 2002
Proyecto:	SIPU
Etapa:	Diseño, Ciclo I
Autor:	GAA
Objetivo:	Describir el proceso de ingreso al sistema de consulta

1.1	Usuario	Acceder al URL de SIPU	N/A
1.2	Usuario	Autentificarse y solicitar ingreso	3.1
1.3	Sistema	IF (Identidad correcta)	6.1.2
1.3.1	Sistema	Llamar Caracterizacion.open	4.1
		ELSE	
1.3.2	Sistema	Llamar Error.open	13.1
1.4	Usuario	IF (Usuario desea reintentar)	N/A
1.4.1	Sistema	THIS(1.1)	N/A
		ELSE	
1.4.2	Sistema	Ir al URL por defecto	N/A

Tabla 6.20

Como se puede apreciar, este formato es muy compacto y explícito. Semánticamente es equivalente a un diagrama de actividades de UML y es elaborado con una herramienta que permita la inclusión de ligas, éste permitirá navegar entre las distintas hojas de actividades que se representan (en el formato anterior) como palabras subrayadas. Adicionalmente, en un formato como éste, es posible incluir información referente a los valores de retorno que arroja una actividad y usarlos en el mismo formato, como por ejemplo en los condicionales de IF.

El número de diagramas de actividades que se generaron en la fase de diseño del proyecto **SIPU**© fue grande. Por este motivo, en el presente documento sólo se presentarán algunos ejemplos de estos diagramas, pero utilizando el formato propuesto que, como se mencionó anteriormente, es semánticamente equivalente. Si se desea obtener una referencia completa de la diagramación UML para el proyecto **SIPU**© es posible consultar la carpeta del proyecto.

El primero de los ejemplos se refiere a la forma en la que un usuario se caracteriza en el sistema:

Hoja de Actividades	
Actividad:	3.7
Procedencia:	1.7

Fecha:		Lunes 4 de Febrero de 2002	
Proyecto:		SIPU	
Etapa:		Diseño, Ciclo I	
Autor:		GAA	
Objetivo:		Describir el proceso de caracterización de un usuario en el sistema	
1.1	Sistema	Se invoca el método 'open' de la clase 'CaracterizaUsuario'	N/A
1.2	Sistema	Se despliega un objeto que permite la selección de un tipo de usuario y se despliega una liga ofreciendo el registro personal de manera opcional	N/A
1.3	Usuario	IF (Se selecciona "registro personal")	N/A
1.3.1	Sistema	Llama RegistroPersonal.open	7.1
		ELSE	
1.3.2	Usuario	IF (Seleccionar un tipo de usuario)	
1.3.2.1	Sistema	Llama SeleccionarConsulta.open	3.4
		ELSE	
1.3.2.2	Usuario	Abandonar sesión	4.1

Tabla 6.21

El segundo ejemplo se refiere a la forma en la que el sistema calcula y despliega los datos resultantes de una consulta del tipo "Equipo":

Hoja de Actividades	
Actividad:	2.9
[illegible]:	4.3
Fecha:	Lunes 4 de Febrero de 2002
Proyecto:	SIPU
Etapa:	Diseño, Ciclo I
Autor:	GAA
Objetivo:	Describir el proceso que calcula y despliega los datos Resultantes de una consulta de tipo "Equipo".

1.1		Acceder al URL de SIPU	N/A
1.2		Autentificarse y solicitar ingreso	3.1
1.3		IF (Identidad correcta)	6.1.2
1.3.1		Caracterizarse	4.1
		ELSE	
1.3.2		Aceptar Notificación de error	13.1
1.4		IF (Usuario desea reintentar)	N/A
1.4.1		THIS(1.1)	N/A
		ELSE	
1.4.2		Ir al URL por defecto	N/A

Tabla 6.22

En este punto, después de haber realizado todos los formatos de actividades necesarios, es posible iniciar con la codificación de los mismos. Sin embargo, y como una referencia más de apoyo a la codificación, es posible generar diagramas de secuencia que es lo que se expone a continuación.

Diagramas de secuencia

Al igual que los diagramas de actividades, los diagramas de secuencia son refinamientos de los que se generaron en la fase de Análisis. Nuevamente, se exponen sólo algunos ejemplos que muestren este refinamiento.

El primer ejemplo se refiere al refinamiento del diagrama representado por la figura 6.10 de la fase de Análisis correspondiente a la secuencia "Mostrar Resultados", pero aplicado a la entidad de Investigador.

Los diagramas de secuencia no fueron modificados en ningún sentido, ya que la semántica de los mismos es suficiente para los propósitos de este trabajo.

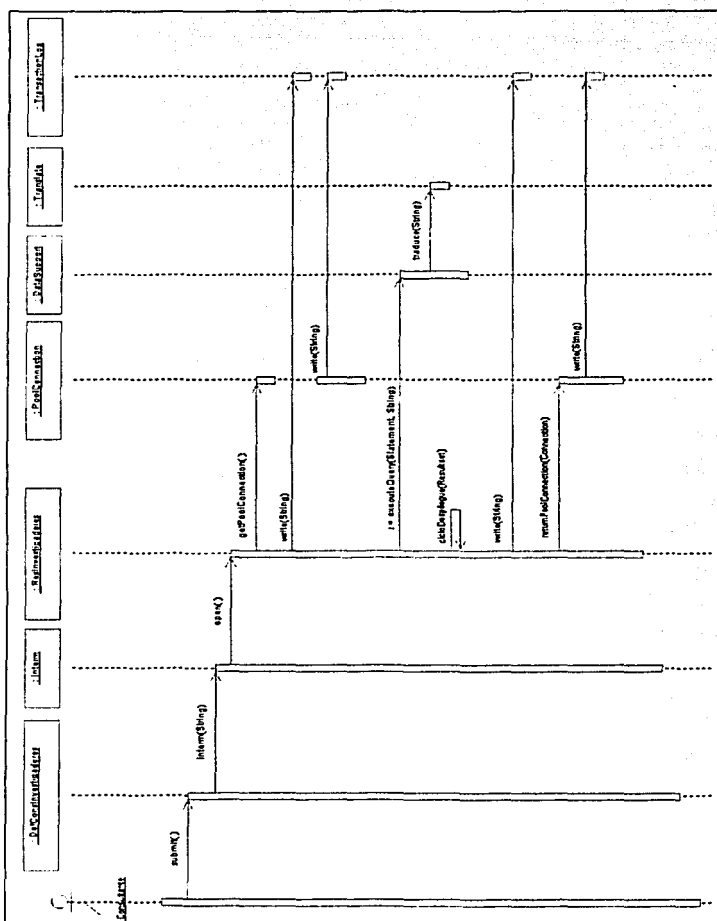


Figura 6.13 Despliegue de la hoja de resultados de la consulta "investigadores"

El segundo ejemplo se refiere al refinamiento del diagrama representado por la figura 6.9 de la fase de Análisis correspondiente a la secuencia "Registro Voluntario".

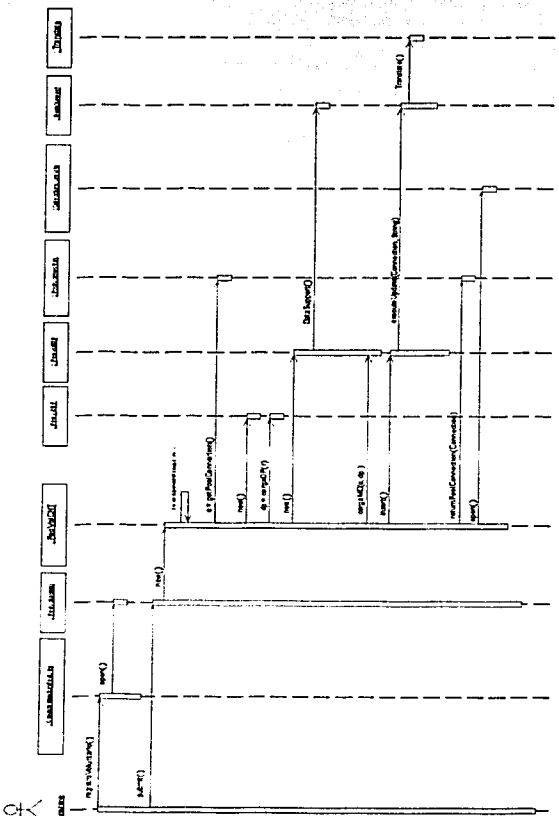


Figura 6.14 "Registro voluntario de un usuario al sistema"

Como se puede apreciar en los diagramas de secuencia anteriores, las flechas (a diferencia de lo que ocurría en los diagramas de secuencia de la fase de Análisis) están etiquetadas con los métodos que corresponden a las clases a las que apuntan y los mensajes son menos abstractos. Así mismo, el detalle es muy técnico e incluye aspectos relacionados con la tecnología que se esté usando.

Resta generar los diagramas de instalación que, para el caso de estudio presentado en este trabajo, son muy simples. Estos nos ayudarán a entender la distribución física de componentes en la arquitectura de hardware.

Diagramas de instalación

Para la aplicación **SIPU©**, el diagrama de instalación es bastante simple. Básicamente, la aplicación se encuentra instalada en un servidor central que posee una base de datos (o bien se conecta a ella) y da servicio (a través de Internet) a usuarios remotos que tienen instalado un navegador:

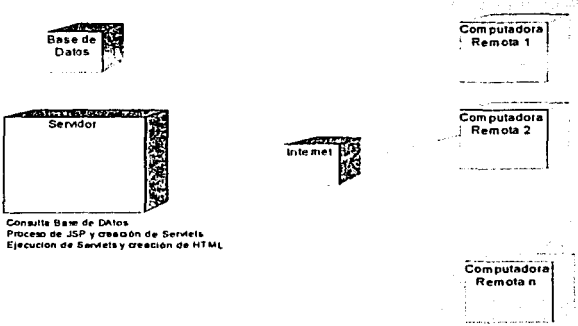


Figura 6.15

La última parte de la fase de Diseño se refiere a la generación de un plan de pruebas, que será aplicado durante la fase de implementación.

6.2.3 Generar plan de pruebas unitarias y de integración

Para el caso de estudio **SIPU©**, el plan de pruebas unitarias y el plan de integración fueron integradas simultáneamente. Esto es debido a que la propuesta (Ver HRT 0027) global de pruebas, consiste en generar ambientes base a los que se integrarán nuevas clases, generando así un nuevo ambiente para integrar y probar las siguientes. Este proceso se repetirá hasta que se obtengan los resultados propuestos en la fase de estrategia para el ciclo. A continuación se presenta la HRT mencionada:

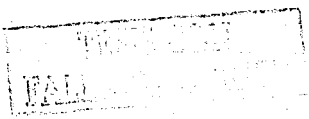
Referencia: 0027 **Hoja de recomendaciones técnicas**

Proyecto:	SIPU
Fase / Ciclo:	Diseño, Ciclo I
Fecha de creación:	Viernes 8 de Febrero de 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	Ha resultado inoperante generar para cada clase otra que la pruebe.
Propuesta:	Probar conjuntos de clases relacionadas entre si, en ambientes incrementales que eventualmente conformarán el ambiente de producción final.
Justificación:	- Esta forma de probar paquetes de clases se apega al proceso realizado en la vida real. - Con esta técnica se va construyendo un ambiente que, eventualmente será el de producción.
Estrategia:	- Establecer ambiente base que incluye la instalación del hardware necesario, el sistema operativo, el lenguaje de desarrollo y el servidor de aplicaciones. - El segundo conjunto de clases al ser unido con el primero genera un nuevo ambiente, al que se unirá el tercer nivel para continuar con este proceso, hasta llegar al último nivel, que será el ambiente de producción completo. - En cada nuevo ambiente, se prueban las clases que se acaban de integrar y si se desea, también es posible realizar mas

	pruebas sobre clases integradas previamente.
Beneficios:	- Se prueban repetitivamente las mismas clases una y otra vez. - Al final del proceso se obtiene el ambiente completo probado
Desventajas:	- Esta técnica no se apega a la tradicional prueba de unidades. - Redundancia en la ejecución de pruebas.
Impacto en calidad:	- Funcional. - Robusto.
Tecnologías Involucradas:	N/A.
Responsable de ejecución:	FNL (Administrador de Soporte)
Aprobación:	_____ _____ _____ _____ _____ _____

Tabla 6.23

Para poner en práctica lo anterior, es necesario generar una secuencia de desarrollo e integración que puede ser representada a través de una gráfica como la siguiente:



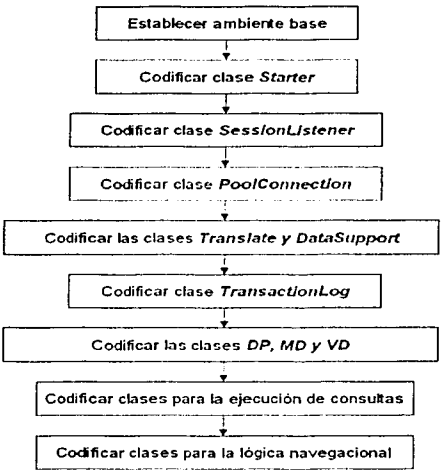


Figura 6.16

El primer nivel (establecer ambiente base) incluye la instalación del hardware necesario, el sistema operativo, el lenguaje de desarrollo y el servidor de aplicaciones. En este punto, la codificación será prácticamente nula, debido a que generalmente no es necesario invertir recursos para probar el ambiente base.

El segundo nivel (codificar clase Starter) al ser unido con el primero genera un nuevo ambiente, al que se unirá el tercer nivel (Codificar la clase SessionListener) para continuar con este proceso, hasta llegar al último nivel.

En cada nuevo ambiente, se prueban las clases que se acaban de integrar y si se desea, también es posible realizar mas pruebas sobre clases integradas previamente.

Cada nivel define un bloque que contiene una o más clases. Esto indica que las clases que componen el bloque deben ser generadas de manera simultánea para ser probadas como una unidad.

Para ejemplificar lo anterior, en el bloque "Codificar clases DP, MD y VD" de la figura 6.16, (para la entidad Investigadores) se propone la codificación de las siguientes partes simultáneamente:

- InvestigadoresDP
- InvestigadoresMD
- InvestigadoresCNT
- InvestigadoresJSP
- InvestigadoresHTML

Cada una de estas partes se complementa con la otra y la mejor manera de probar su funcionamiento es en conjunto. Lo mismo debe ocurrir para el resto de entidades, definidas por las clases de dominio de problema.

A continuación, se presenta el plan de pruebas unitario incremental para cada uno de los niveles que fue utilizado en el caso de estudio (proyecto **SIPU©**) que se presenta en este trabajo:

1 Probar ambiente base	
Prueba	Resultante
Verificar que estén correctamente instalados el lenguaje a utilizar, los programas de soporte a la aplicación como manejador de bases de datos, contenedor de JSP's y editores de código.	Cada elemento funciona correctamente.

2 Probar la clase Starter	
Prueba	Resultante
Iniciar el contenedor de JSP	La terminal muestra un mensaje indicando que el método Init de la clase Starter se ha invocado.

3 Probar la clase HomeMadeSessionListener	
Prueba	Resultante

Generar una clase que acceda a cualquier recurso de la aplicación. Esperar un tiempo mayor al definido como 'timeout' en el archivo de propiedades.	La terminal muestra un mensaje indicando que una nueva sesión acaba de ser creada. La terminal muestra un mensaje indicando que la sesión con cierto ID acaba de ser destruida.

4 Probar la clase PoolConnection	
Prueba	Resultante
Instanciar la clase PoolConnection tomando una conexión del pool, usándola y regresándola a éste. Por cada acción, enviar un mensaje a la terminal indicando el tamaño del pool.	La base de datos se debe haber afectado como consecuencia de las acciones realizadas con el objeto de tipo Connection. Adicionalmente, la terminal debe reportar el tamaño inicial del pool, una disminución en 1 del mismo y finalmente el tamaño original del pool.

5 Probar la clase DataSupport	
Prueba	Resultante
Instanciar la clase DataSupport usar la clase PoolConnection, y ejecutar una acción que afecte a la base de datos	La base de datos se debe haber afectado como consecuencia de las acciones realizadas con el objeto de tipo DataSupport

6 Probar las clases de dominio de problema, de manejo de datos y sus correspondientes controladores	
Prueba	Resultante
Generar una clase cliente que utilice estas clases y solicitar guardar, recuperar y modificar la información presentada	La información debe, en cada caso, ser guardada, recuperada y modificada adecuadamente

7 Probar la lógica de navegación	
Prueba	Resultante
Navegar entre las distintas vistas de la aplicación.	La navegación resulta ser coherente y acorde a la lógica de negocio

8 Probar los métodos de ejecución de consultas	
Prueba	Resultante
Crear una clase de prueba que posea en	La aplicación muestra una resultante con

'código duro' los parámetros de una consulta. Invocar los distintos métodos de ejecución de consultas con estos parámetros.	datos que efectivamente corresponden a la consulta solicitada.

9 Probar ingreso exitoso al sistema	
Prueba	Resultante
En la clase inicial capturar un usuario y una contraseña válidos.	La aplicación muestra una forma de caracterización.

10 Probar ingreso no exitoso al sistema	
Prueba	Resultante
En la clase inicial capturar un usuario y una contraseña no válidos.	La aplicación muestra una forma de error.

Con esto concluye la sección 6.2, que se refiere a la fase de Diseño. Con base en los documentos generados, será posible realizar una implementación eficiente en la siguiente fase.

6.3 Implementación y pruebas

Con base en los documentos generados en las fases de análisis y de diseño, deberá ser posible iniciar la codificación de los subproductos que integrarán el sistema completo. Los documentos del análisis serán una guía para la lógica de negocio y los documentos del diseño para la implementación técnica, en términos de arquitectura, plataforma y lenguaje.

Para esta fase fueron de gran utilidad los formatos propuestos equivalentes a los diagramas de actividades, ya que estos definieron la estructura base del código implementado.

La cuidadosa revisión del diseño detallado nos permitirá asignar trabajo de implementación con un adecuado balance de cargas.

Finalmente, uno de los subproductos de la implementación será el manual técnico, que contiene información relacionada más con la plataforma e instalación del producto que con la lógica de negocio.

Basado en el proceso de desarrollo TSPi, las principales actividades a realizar en esta fase son los siguientes:

- Revisar el diseño detallado y planear tareas individuales de implementación.
- Codificar, diseñar y aplicar pruebas unitarias y corregir código.
- Producir manuales técnicos.

6.3.1 Revisar diseño detallado y planear tareas individuales de implementación

De los documentos generados en la fase anterior, para el caso de estudio **SIPU©**, se tiene el siguiente resumen de tareas de implementación:

Codificar clases de dominio de problema
InvestigadoresDP
LineasInvestigacionDP
ProyectosInvestigacionDP
ServiciosDP
DepUnivDP
EquiposDP

Codificar clases de manejo de datos
InvestigadoresMD
LineasInvestigacionMD
ProyectosInvestigacionMD
ServiciosMD
DepUnivMD
EquiposMD

Codificar clases controladoras (para DP)
InvestigadoresCNT
LineasInvestigacionCNT
ProyectosInvestigacionCNT

ServiciosCNT
DepUnivCNT
EquiposCNT

Codificar clases controladoras (soporte)
DataSupport
Global
HomeMadeSessionListener
Starter
TransactionLog
PoolConnection
Translate

Codificar archivos JSP (Envolventes)
BienvenidaJSP
IngresoSistemaJSP
RegistroVoluntarioJSP
BarraAccesoRapidoJSP
SeleccionaConsultaJSP
ConsultaLineasInvestigaciónJSP
ConsultaDepUnivJSP
ConsultaProyectosInvJSP
ConsultaInvetigadoresJSP
ConsultaServiciosJSP
ConsultaEquiposJSP
ResManyLineasInvestigaciónJSP
ResManyDepUnivJSP
ResManyProyectosInvJSP
ResManyInvestigadoresJSP
ResManyServiciosJSP
ResManyEquiposJSP
ResOneLineaInvestigaciónJSP
ResOneDepUnivJSP
ResOneProyectoInvJSP
ResOneInvestigadorJSP
ResOneServicioJSP

ResOneEquipoJSP

Codificar archivos JSP (Servicios indep.)
CeroTuplasJSP
ErrorLoginJSP
SesionTerminadaJSP

Codificar archivos HTML
Bienvenida
IngresoSistema
RegistroVoluntario
BarraAccesoRapido
SeleccionaConsulta
ConsultaLineasInvestigación
ConsultaDepUniv
ConsultaProyectosInv
ConsultaInvetigadores
ConsultaServicios
ConsultaEquipos
ResManyLineasInvestigación
ResManyDepUniv
ResManyProyectosInv
ResManyInvestigadores
ResManyServicios
ResManyEquipos
ResOneLineaInvestigación
ResOneDepUniv
ResOneProyectoInv
ResOneInvestigador
ResOneServicio
ResOneEquipo
CeroTuplas
ErrorLogin
SesionTerminada

Tabla 6.24

Con base en la información presentada en la tabla 6.24, se realizó el siguiente plan de tareas individuales de implementación:

	Nombre	Tareas
MEM	Ing. Mauricio Espinoza	Generación de archivos HTML
GAA	Mat. Gustavo Arellano	Generación de clases de soporte a bajo nivel.
FNL	Ing. Francisco Noguez	Generación del resto de las clases

Tabla 6.25

En la distribución anterior, las dos tareas que requieren de mayor atención son la generación de las clases de soporte de bajo nivel por su complejidad de elaboración y la última tarea, que es la generación del resto de las clases. Sin embargo, esta última tarea aunque puede parecer muy extensa es, en realidad, mecánica y probablemente muy simple. Esto último lo justifica la HRT 29 que será presentada en la sección 6.3.3 de este capítulo.

En la siguiente sección se describe la codificación de las clases que integrarán el sistema.

6.3.2 Codificar, aplicar pruebas y corregir código

Como se mencionó anteriormente, la primera etapa de este proceso consiste en establecer el ambiente inicial. Para probar este ambiente basta iniciar el sistema base y ejecutar los servicios que éste tiene.

La generación de las siguientes clases de soporte al proyecto **SIPU©** deberá seguir la especificación formal de las mismas e implementar los algoritmos necesarios para su correcto funcionamiento:

- Starter
- SessionListener
- PoolConnection

- DataSupport
- Translate
- TransactionLog

El código de estas clases podrá ser consultado en el disco compacto que se incluye en este trabajo.

Para la codificación de los paquetes de clases definidos en la sección anterior conviene hacer notar que cada paquete tiene una estructura similar, que varía solamente en función de la clase de dominio de problema base. Esto nos lleva a sugerir que, con base en el primer paquete, se pueden generar los demás de manera mecánica. La siguiente HRT propone esto:

Referencia: 0028 **Hoja de recomendaciones técnicas**

Proyecto:	SIPU
Fase / Ciclo:	Implementación, Ciclo I
Fecha de creación:	Viernes 22 de Febrero de 2002
Fecha limite para ejecución:	De inmediato
Autor:	GAA
Origen:	La estructura de los paquetes de clases que presentan información para su edición y visualización es similar.
Propuesta:	Implementar un proceso automático de generación de código basado en un machote base.
Justificación:	- La generación automática de código es menos propensa a fallas de codificación. - El tiempo de codificación y pruebas se reduce en gran medida al generar código de manera automática. - La integración de código está asegurada, ya que éste es generado con ese propósito.
Estrategia:	Implementar en una hoja de cálculo un proceso automático de generación de código Java (clases DP, MD, CNT y JSP) que reciba como entrada el nombre y tipo de dato de cada uno de los atributos

	de una clase de dominio de problema.
Beneficios:	- Simplicidad en el desarrollo y la implementación. - Ahorro de tiempo de codificación. - Facilidad de administración, actualización y mantenimiento. - Minimización de errores de codificación. - Alto grado de integración. - Este proceso es reutilizable.
Desventajas:	- Inversión de tiempo en la implementación de este proceso automático de generación de código. - Se requiere de conocimiento de la hoja de cálculo y su lenguaje para esta implementación.
Impacto en calidad:	- Mantenible. - Robusto. - Escalable. - Modular.
Tecnologías Involucradas:	Hojas de cálculo.
Responsable de ejecución:	FNL (Administrador de Soporte).
Aprobación:	

Tabla 6.26

De lo anterior, a continuación se presenta un ejemplo que muestra la utilidad de esta propuesta:

Base de datos: **SIPU©**
Tabla: Usuarios
Campos:

idUsuario	INTEGER
nombre	VARCHAR(20)
apPaterno	VARCHAR(20)
apMaterno	VARCHAR(20)
escolaridad	INTEGER
edoCivil	INTEGER
fuma	BOOLEAN

bebe BOOLEAN
sobrepeso BOOLEAN

La generación automática de código de la hoja electrónica arrojó los siguientes resultados:

```
/*
 * Proyecto:     Avalúos Inmobiliarios
 * Paquete:      Infonavit
 * Módulo:       UsuariosDP.java
 * Tipo:         clase
 * Autor:        GAA
 * Fecha:        lunes 13 de enero de 2003
 * Descripción: .
 *                 Generación automática de código.
 *                 Clase de dominio de problema para
 *                 la entidad 'Usuarios'
 *
 * Historia:     .
 *               Creacion 130103_1939
 *
 */


package tesisDP;

public class UsuariosDP {
    private int         idUsuario;
    private String      nombre;
    private String      apPaterno;
    private String      apMaterno;
    private int         escolaridad;
    private int         edoCivil;
    private boolean     fuma;
    private boolean     bebe;
    private boolean     sobrePeso;

    // Metodos set y get para idUsuario:
    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }
    public int getIdUsuario() {
        return this.idUsuario;
    }

    // Métodos set y get para nombre:
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getNombre() {
```

```
        return this.nombre;
    }

    // Métodos set y get para apPaterno:
    public void setApPaterno(String apPaterno) {
        this.apPaterno = apPaterno;
    }
    public String getApPaterno() {
        return this.apPaterno;
    }

    // Métodos set y get para apMaterno:
    public void setApMaterno(String apMaterno) {
        this.apMaterno = apMaterno;
    }
    public String getApMaterno() {
        return this.apMaterno;
    }

    // Métodos set y get para escolaridad:
    public void setEscolaridad(int escolaridad) {
        this.escolaridad = escolaridad;
    }
    public int getEscolaridad() {
        return this.escolaridad;
    }

    // Métodos set y get para edoCivil:
    public void setEdoCivil(int edoCivil) {
        this.edoCivil = edoCivil;
    }
    public int getEdoCivil() {
        return this.edoCivil;
    }

    // Métodos set y get para fuma:
    public void setFuma(boolean fuma) {
        this.fuma = fuma;
    }
    public boolean getFuma() {
        return this.fuma;
    }

    // Métodos set y get para bebe:
    public void setBebe(boolean bebe) {
        this.bebe = bebe;
    }
    public boolean getBebe() {
        return this.bebe;
    }

    // Métodos set y get para sobrePeso:
    public void setSobrePeso(boolean sobrePeso) {
```

```
        this.sobrePeso = sobrePeso;
    }
    public boolean getSobrePeso() {
        return this.sobrePeso;
    }

}// Fin de clase UsuariosDP
```

La clase UsuariosDP implementa los métodos get y set para cada uno de sus atributos, considerando tanto sus valores de retorno, como los tipos de datos correspondientes a los parámetros formales de la función.

La siguiente clase, UsuariosMD también es generada de manera automática por la hoja electrónica:

```
/*
 * Proyecto:     Avalúos Inmobiliarios
 * Paquete:      Infonavit
 * Módulo:       UsuariosMD.java
 * Tipo:         clase
 * Autor:        GAA
 * Fecha:        lunes 13 de enero de 2003
 * Descripción: .
 *                 Generación automática de código.
 *                 Clase de manejo de datos para
 *                 la entidad 'Usuarios'
 *
 * Historia:     .
 *               Creación 130103_1939
 *
 */

package tesisMD;

import java.sql.*;
import tools.*;
import tesisDP.*;

public class UsuariosMD extends SupportMD {
    Connection conn=null;
    UsuariosDP obj=null;
    String str="";

    public UsuariosMD(){
    }

    public void init(Connection conn, UsuariosDP obj){
        this.conn = conn;
```

```
        this.obj = obj;
    }

    public void insert(){
        str = "";
        str = str + "INSERT INTO Usuarios (";
        str = str + "idUsuario, ";
        str = str + "nombre, ";
        str = str + "apPaterno, ";
        str = str + "apMaterno, ";
        str = str + "escolaridad, ";
        str = str + "edoCivil, ";
        str = str + "fuma, ";
        str = str + "bebe, ";
        str = str + "sobrePeso) ";
        str = str + "VALUES(";
        str = str + obj.getIdUsuario() + ", ";
        str = str + comillas(obj.getNombre()) + ", ";
        str = str + comillas(obj.getApPaterno()) + ", ";
        str = str + comillas(obj.getApMaterno()) + ", ";
        str = str + obj.getEscolaridad() + ", ";
        str = str + obj.getEdoCivil() + ", ";
        str = str + obj.getFuma() + ", ";
        str = str + obj.getBebe() + ", ";
        str = str + obj.getSobrePeso() + "); ";
        ds.executeUpdate(conn, str);
    }

    public void update(){
        str = "";
        str = str + "UPDATE Usuarios SET ";
        str = str + "nombre= " + comillas(obj.getNombre()) + ",";
        str = str + "apPaterno= " + comillas(obj.getApPaterno()) + ",";
        str = str + "apMaterno= " + comillas(obj.getApMaterno()) + ",";
        str = str + "escolaridad= " + obj.getEscolaridad() + ",";
        str = str + "edoCivil= " + obj.getEdoCivil() + ",";
        str = str + "fuma= " + obj.getFuma() + ",";
        str = str + "bebe= " + obj.getBebe() + ",";
        str = str + "sobrePeso= " + obj.getSobrePeso() + " ";
        str = str + "where idUsuario= " + obj.getIdUsuario();
        ds.executeUpdate(conn, str);
    }

    public void delete(){
        str = "";
        str = str + "DELETE FROM Usuarios ";
        str = str + "where idUsuario= " + obj.getIdUsuario();
        ds.executeUpdate(conn, str);
    }

    public boolean find(){
        str = "";
        str = str + "SELECT * FROM Usuarios ";
```

```
        str = str + "where idUsuario= " + obj.getIdUsuario();
      return (ds.size(conn, str)>0);
    }

    public UsuariosDP load(){
        UsuariosDP tmp = new UsuariosDP();
        str = "";
        str = str + "SELECT * FROM Usuarios ";
        str = str + "where idUsuario= " + obj.getIdUsuario();
        try{
            Statement stmt = conn.createStatement();
            ResultSet rs = ds.executeQuery(stmt, str);
            if (rs.next()){
                tmp.setIdUsuario(   gInt(rs.getString("idUsuario")));
                tmp.setNombre(      gStr(rs.getString("nombre")));
                tmp.setApPaterno(   gStr(rs.getString("apPaterno")));
                tmp.setApMaterno(   gStr(rs.getString("apMaterno")));
                tmp.setEscolaridad(gInt(rs.getString("escolaridad")));
                tmp.setEdoCivil(    gInt(rs.getString("edoCivil")));
                tmp.setFuma(        gBool(rs.getString("fuma")));
                tmp.setBebe(        gBool(rs.getString("bebe")));
                tmp.setSobrePeso(   gBool(rs.getString("sobrePeso")));
            }
          else{
                tmp.setIdUsuario(   obj.getIdUsuario());
                tmp.setNombre(      "");
                tmp.setApPaterno(   "");
                tmp.setApMaterno(   "");
                tmp.setEscolaridad(0);
                tmp.setEdoCivil(    0);
                tmp.setFuma(        false);
                tmp.setBebe(        false);
                tmp.setSobrePeso(   false);
            }
        }
        catch(Exception e){
            TransactionLog.write("UsuariosMD:find " + e.toString());
        }
        finally{
            return tmp;
        }
    }

}// UsuariosMD
```

Esta clase implementa los métodos insert, update, delete, find y load, que permiten la búsqueda y actualización de la información que provee la clase de dominio de problema y posterior a su generación, sólo requiere de algunas precisiones que se refieren a la forma en cómo la clase deberá encontrar una tupla en la base de datos. Cabe mencionar que las clases de manejo de datos

soportan el retorno de valores por defecto, que es uno de los puntos en donde se debe modificar el código de éstas, para definir el valor de retorno deseado.

En este punto conviene mencionar que originalmente sólo se habían propuesto para esta clase los métodos INSERT, UPDATE, DELETE y FIND, que regresaba (en caso de encontrarlo) un objeto de tipo UsusariosDP o null en caso de no encontrarlo. Sin embargo, las dos siguientes HRT's influyeron en la codificación final de las clases de manejo de datos:

Referencia: 0029	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Implementación, Ciclo I
Fecha de creación:	Viernes 22 de Febrero de 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	El método FIND indica que se realizará una búsqueda y esta sugiere una respuesta boleana.
Propuesta:	Modificar el método find para que retorne un valor boleano. Y compensar la disminución de funcionalidad con otro método.
Justificación:	- La semántica del método find se fortalece. - El nuevo método que retoma la funcionalidad que tenía find también refuerza la semántica del mismo.
Estrategia:	Modificar el método find para que retorne un falso en caso de no encontrar una coincidencia y verdadero en caso de encontrarla. Codificar un método 'load' que retorne el objeto de tipo DP correspondiente y en caso de no existir, que lo retorne con valores por defecto definidos en el mismo método 'load'.
Beneficios:	- Mejora en la semántica de los métodos. - Obtención de valores por defecto de manera automática. - Mejor organización del código.
Desventajas:	- Inversión de tiempo en la implementación de este mecanismo - Se requiere de más cambios manuales en la definición de los

	valores de retorno por defecto.
Impacto en calidad:	- Mantenible. - Escalable. - Modular.
Tecnologías Involucradas:	JAVA.
Responsable de ejecución:	GAA (Administrador de Desarrollo).
Aprobación:	_____ _____ _____ _____

Tabla 6.27

Referencia: 0030	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Implementación, Ciclo I
Fecha de creación:	Viernes 22 de Febrero de 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	Las clases de manejo de datos requieren de varios métodos comunes
Propuesta:	Descender las clases de manejo de datos de otra que contenga los métodos comunes.
Justificación:	- De esta forma los métodos comunes estarán disponibles para toda clase del paquete MD
Estrategia:	Codificar una clase llamada 'SupportMD' que contendrá métodos de conversión de tipos de datos, y funciones de soporte como comillas, numerales para las fechas y generación de espacios complementarios y formato.
Beneficios:	- Ahorro en la cantidad de codificación de una clase MD - Apoyo a la generación automática del código

	- Mejor organización del código. - Uniformidad en la forma de generar código. - Evasión de errores de conversión de tipo.
Desventajas:	- Inversión de tiempo en la implementación de este mecanismo.
Impacto en calidad:	- Mantenible. - Robusto. - Modular.
Tecnologías Involucradas:	JAVA.
Responsable de ejecución:	GAA (Administrador de Desarrollo).
Aprobación:	_____ _____ _____ _____

Tabla 6.28

Finalmente, se presenta el código generado para la clase controladora correspondiente y para el JSP resultante, que también fue generado de manera automática:

```
/*
 * Proyecto:     Avalúos Inmobiliarios
 * Paquete:      Infonavit
 * Módulo:       UsuariosCNT.java
 * Tipo:         clase
 * Autor:        GAA
 * Fecha:        lunes 13 de enero de 2003
 * Descripción: .
 *                 Generación automática de código.
 *                 Clase de controladora para
 *                 la entidad 'Usuarios'
 *
 * Historia:     .
 *               Creación 130103_1939
 *
 */
```

```
package tesisCNT;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.ServletContext;

import tools.*;
import java.sql.*;
import tesisDP.*;
import tesisMD.*;
import java.io.*;


public class Usuarios_CNT extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException {

        ServletContext servletContext = getServletContext();
        HttpSession session = request.getSession();

        // En este controller se usarán las clases de
        // dominio de problema y de manejo de datos siguientes
        UsuariosDP  cargador    = new UsuariosDP();
        UsuariosMD  cargadorMD  = new UsuariosMD();

      // Llave del objeto:
      String idU = (String) session.getAttribute("idU");
        cargador.setIdUsuario(Integer.parseInt(idU));


      // Objetos de tipo 'text':
        cargador.setNombre( request.getParameter("nombre") );
        cargador.setApPaterno( request.getParameter("apPaterno") );
        cargador.setApMaterno( request.getParameter("apMaterno") );


      // Objetos de tipo 'check':
      String tmp="";
      tmp = request.getParameter("fuma") + "";
        cargador.setFuma( !tmp.equals("null") );
      tmp = request.getParameter("bebe") + "";
        cargador.setBebe( !tmp.equals("null") );
      tmp = request.getParameter("sobrePeso") + "";
        cargador.setSobrePeso( !tmp.equals("null") );


      // Objetos de tipo 'combo':
        cargador.setEscolaridad(
Integer.parseInt(request.getParameter("escolaridad")) );
```

```
      // Objetos de tipo 'radio':
        cargador.setEdoCivil(
Integer.parseInt(request.getParameter("edoCivil")) );


        // Guardando los datos almacenados en el objeto 'cargador'
        PoolConnection pool = (PoolConnection)
servletContext.getAttribute("pool");
        Connection cnn = null;
        try{
            // Pidiendo una conección al pool
            cnn = pool.getPoolConnection();

            // Guardando
            cargadorMD.init(cnn, cargador);
            if(cargadorMD.find()){
                cargadorMD.update();
            }
            else{
                cargadorMD.insert();
            }
        }
        catch(Exception e){
            System.out.println("Usuarios_CNT.jsp: " + e.toString());
        }
        finally{
            // Regresando la conección
            if (cnn!=null) pool.returnPoolConnection(cnn);
        }

        response.setContentType("text/html");
        try{
            PrintWriter out = response.getWriter();
            out.println("<html><body><form method='post'
action='../contenido/Usuarios.jsp?valor=" + idU + "'>");
            out.println("v1: " + idU + "<BR>");
            out.println("v2: " + request.getParameter("name") + "<BR>");
            out.println("v3: " + request.getParameter("apPat") + "<BR>");
            out.println("v4: " + request.getParameter("apMat") + "<BR>");
            out.println("v5: " + request.getParameter("escol") + "<BR>");
            out.println("v6: " + request.getParameter("edoC") + "<BR>");
            out.println("v7: " + request.getParameter("fuma") + "<BR>");
            out.println("v8: " + request.getParameter("bebe") + "<BR>");
            out.println("v9: " + request.getParameter("sPeso") + "<BR>");
            out.println("");
            out.println("<input type='submit' value='OK'>");
            out.println("</form></body></html>");
        }
        catch(Exception e){
            System.out.println("Error en escritura a terminal");
        }
    }// Fin de DoPost
```

```
}// Fin de la clase Usuarios_CNT
```

El JSP resultante, también fue influenciado por la siguiente HRT:

Referencia: 0027	**Hoja de recomendaciones técnicas**
Proyecto:	SIPU
Fase / Ciclo:	Implementación, Ciclo I
Fecha de creación:	Viernes 22 de Febrero de 2002
Fecha límite para ejecución:	De inmediato
Autor:	GAA
Origen:	La parte de la vista (HTML) debe ser independiente del código que está asociado a ella.
Propuesta:	Generar, para cada HTML, un JSP 'envolvente' que provea los recursos necesarios para el despliegue de los datos que el HTML mostrará
Justificación:	- Separación entre el diseño estético y el código de despliegue de datos.
Estrategia:	Codificar, para cada HTML, un JSP que contenga código jScript generado dinámicamente con JAVA que cargue a los objetos HTML.
Beneficios:	- Automatización de la carga y despliegue de información - Independencia entre el diseño estético y código
Desventajas:	- Inversión de tiempo en la implementación de este mecanismo - Dependencia de JScript, que es diferente para cada navegador
Impacto en calidad:	- Mantenible. - Robusto - Modular
Tecnologías Involucradas:	JAVA y JScript
Responsable de ejecución:	GAA (Administrador de Desarrollo)

Aprobación:	_____ _____ _____ _____

Tabla 6.29

A continuación se presenta el JSP generado automáticamente y basado en la HRT anterior:

```
/*
 * Proyecto:     Avalúos Inmobiliarios
 * Paquete:      Infonavit
 * Módulo:       Usuarios.jsp
 * Tipo:         clase
 * Autor:        GAA
 * Fecha:        lunes 13 de enero de 2003
 * Descripción: .
 *                 Generación automática de código.
 *                 JSP resultante para
 *                 la entidad 'Usuarios'
 *
 * Historia:     .
 *               Creación 130103_1939
 *
 */

<%@page language="java" %>
<%@page import="java.sql.*" %>
<%@page import="tesisDP.*" %>
<%@page import="tesisMD.*" %>
<%@page import="tools.*" %>
<jsp:useBean id="pool" class="tools.PoolConnection" scope="application"/>


<%
    // Usado para crear el(los) combo(s) dinámico(s)
    DataSupport dataSupport = new DataSupport();

    // Sólo requerire crear un único combo:
    String escolaridad="";

    // En este JSP se usarán las clases de dominio
    // de problema y de manejo de datos siguientes
    UsuariosDP  cargador    = new UsuariosDP();
    UsuariosMD  cargadorMD  = new UsuariosMD();

    // La variable 'desactiva' sirve para
    // activar o desactivar los objetos HTML
    String desactiva="false";
    String mensaje="";
```

```
    int idU = Integer.parseInt(request.getParameter("valor"));
    session.setAttribute("idU", idU + "");
    cargador.setIdUsuario(idU);

    Connection cnn = null;
    try{
        // Obtengo una conección del pool de conecciones
        cnn = pool.getPoolConnection();

        // Creando el combo dinámicamente.
        // Primero, definir el conjunto de resultados a utilizar
        // Despues, llamar a la funcion que genera la cadena
        // que representará al combo
        // Finalmente, asignarla a la variable que se utilizará
        // en el cuerpo del HTML
        String qry = "select distinctrow valor, texto from escolaridad
order by texto";
        escolaridad = dataSupport.createOptionListSQL(cnn, qry, "valor",
"texto", "escolaridad", "Seleccione una opción", true);

        // El registro correspondiente a la vivienda puede o no existir,
        // si no existe, 'cargador' se llenará con datos 'por default':
        cargadorMD.init(cnn, cargador);
        cargador = cargadorMD.load();

        if (cargadorMD.find()){
            mensaje = "Si encontrado";
        }
        else{
            mensaje = "No encontrado";
        }
    }
    catch(Exception e){
        System.out.println("Error en Usuarios.jsp: " + e.toString());
    }
    finally{
        // SIEMPRE se debe regresar la conección al pool de conecciones
        if (cnn!=null) pool.returnPoolConnection(cnn);
    }
 >

<%@ include file="Usuarios.html" %>

<!--
 *
 * Código jScript de control de carga de información.
 *
 * Este código sólo debe de gestionar la carga de datos
 * en las cajas de texto y su propiedad "disabled"
 *
 -->
<SCRIPT language=JavaScript1.2 type=text/javascript>
```

```
    function dis(state, ruta, valor){
        ruta.value = valor;
        if(ruta.type=="checkbox" || ruta.type=="radio"){
            if (ruta.value=="false")
                ruta.checked=false;
            else
                ruta.checked=true;
        }
        if(ruta.type=="hidden"){
            ruta.disabled=false;
        }
        else{
            if(ruta.disabled==false){
                ruta.disabled=state;
            }
        }
    }

    // Función encargada de rellenar a los objetos del formulario
    function fill(obj, state){
        dis(state, obj.nombre,       "<%= cargador.getNombre() %>");
        dis(state, obj.apPaterno,    "<%= cargador.getApPaterno() %>");
        dis(state, obj.apMaterno,    "<%= cargador.getApMaterno() %>");
        dis(state, obj.escolaridad, "<%= cargador.getEscolaridad() %>");

        dis(state, obj.edoCivil[<%= cargador.getEdoCivil() %>],
      < = cargador.getEdoCivil() %>);

        dis(state, obj.fuma,         "<%= cargador.getFuma() %>");
        dis(state, obj.bebe,         "<%= cargador.getBebe() %>");
        dis(state, obj.sobrePeso,    "<%= cargador.getSobrePeso() %>");
    }

</SCRIPT>
```

El único archivo que no se genera de manera automática es el archivo HTML que se presenta a continuación:

```
<html>
      <body onLoad=fill(window.document.Usuarios,<%= desactiva %>); >
            <form method='post' name='Usuarios'
                  action='../servlet/tesisCNT.Usuarios_CNT'>
                  <input type='hidden' name='FormName' value='Usuarios.jsp'>

                  Nombre: <input type='text' name='nombre'><BR>
                  Apellido Paterno: <input type='text' name='apPaterno'><BR>
                  Apellido Materno: <input type='text' name='apMaterno'><BR>

                  Escolaridad: < = escolaridad %><BR>
```

```
                    <input type='radio' name='edoCivil' value=0>Soltero<BR>
                    <input type='radio' name='edoCivil' value=1>Casado<BR>
                    <input type='radio' name='edoCivil' value=2>Divorciado<BR>
                    <input type='radio' name='edoCivil' value=3>Juntado<BR>

                    <input type='checkbox' name='fuma' >Fuma<BR>
                    <input type='checkbox' name='bebe' >Bebe<BR>
                    <input type='checkbox' name='sobrePeso' >Sobrepeso<BR>

                    Mensaje: <%= mensaje %>

                    <input type='submit' name='ok' value='ok'>
                </form>
        </body>
</html>
```

Este último archivo puede estar basado en un XML que si se puede generar automáticamente, pero que actualmente el proceso automático no hace.

6.3.3 Producir manuales técnicos

En general, un manual técnico debe contener como mínimo:

- Código fuente
- Infraestructura de programación
- Instalación de ambiente de desarrollo
- Solución de problemas comunes
- Referencias técnicas

Lo anterior no se incluye debido a que no es el objetivo de esta sección (y la gran extensión de este documento). Sin embargo, puede ser consultado en la carpeta de desarrollo del proyecto **SIPU©** o en forma electrónica en el CD que se incluye en este trabajo.

6.4 Pruebas de Sistema

Las actividades a realizar en esta parte se enlistan a continuación:

- Realizar pruebas del sistema
- Corregir pruebas del sistema

En esta parte, se deberán probar aspectos no funcionales de la aplicación, como:

- ✓ Robustez del sistema
- ✓ Eficiencia en la utilización de recursos
- ✓ Seguridad de la información
- ✓ Usabilidad del sistema ante usuarios reales

Para llevar a cabo las pruebas de este tipo de aspectos, se propone utilizar herramientas de monitoreo de desempeño del sistema. Se recomienda que el ambiente de pruebas sea, en la medida de lo posible, similar al ambiente de producción.

Las herramientas que se proponen para la realización de estas pruebas se enlistan a continuación:

Herramienta	Descripción
Racional Robot	Para la automatización de procesos de prueba que requieran de una secuencia larga de ejecución.
Racional Purify	Para el monitoreo de los recursos que el sistema consume en tiempo de ejecución.
Racional Secure	Para el análisis del intercambio de los paquetes de información entre clientes y el servidor de aplicaciones.

Tabla 6.30

Con esto concluye la sección de pruebas de sistema y el flujo de trabajo de Construcción. Cabe mencionar que durante el desarrollo del caso de estudio presentado en este trabajo no fue posible realizar estas pruebas debido a que no se disponía del conjunto de herramientas mencionadas en la tabla 6.5.1 y

por ello, no existe documentación referente a la prueba de las características no funcionales del sistema.

6.5 Conclusiones del capítulo VI

El flujo de construcción es el de mayor extensión y complejidad en un proceso de desarrollo. En él se define y genera el principal entregable: El producto de software. Adicionalmente, se generan productos secundarios pero de igual manera imprescindibles en el desarrollo. Varios de estos productos están dirigidos al cliente (principalmente los casos de uso y los documentos de la fase de Análisis) y otros, al equipo de implementadotes de código (como los documentos de la fase de diseño).

En la fase de Implementación, se generaron un gran número de recomendaciones técnicas, debido principalmente a dos factores: desconocimiento de la tecnología a emplear (que se puede traducir en una falta de información tecnológica previamente documentada) y al esfuerzo por automatizar la generación de código y así facilitar la tarea del equipo de implementadotes de código.

Se propuso una forma de realizar pruebas sobre ambientes incrementales y de esta manera dejar para el final las pruebas de aspectos no funcionales como robustez, usabilidad y eficiencia.

Al final de este flujo de trabajo se debe tener el producto de software y sus subproductos debidamente probados y listos para ser extendidos (o en su caso, instalados en el ambiente de producción) para el nuevo ciclo de desarrollo.

7

Transición

7.1 Introducción

El flujo de transición es el último flujo del proceso de desarrollo presentado en este trabajo. Contiene una sola fase: la fase de Post mortem también conocida como fase de Mejora continua.

En la fase de Post mortem se revisa que el equipo de trabajo haya terminado todas las tareas y registrado todos los datos requeridos. El Post mortem proporciona una manera estructurada de aprendizaje y mejora, porque se evalúa el desempeño personal y del equipo. Cada ciclo de desarrollo de TSPi finaliza con una fase de Post mortem.

PSP y TSPi utilizan la forma de Mejora al Proceso (PMP). En la misma se registran ideas de mejora, las cuales incluyen las mejores prácticas personales sobre el proceso de trabajo en equipo, mejora en las herramientas, cambios al

proceso y cualquier otro aspecto que esté relacionado con la mejora del desempeño en el trabajo.

En el presente trabajo se substituye la forma PMP (Post mortem process) por un conjunto de HRT's, procurando seguir la misma línea de operación propuesta al inicio de este documento.

Adicionalmente, se presentan una serie de datos estadísticos referentes al contenido de las HRT's generadas a lo largo del desarrollo. Estos datos nos ayudarán a visualizar el impacto de las HRT's en el desarrollo desde el punto de vista técnico y tecnológico.

A continuación se presentan las actividades a realizar en la fase de Post mortem (o Mejora continua) y los productos que arrojan este tipo de actividades:

- Revisar datos del proceso
- Evaluar roles de colegas
- Revisar productos
- Realizar reporte general de HRT's

A continuación, se presenta el detalle de cada una de estas actividades y los productos generados en las mismas para el caso de estudio presentado en este trabajo.

7.2 Revisar datos del proceso

El propósito es comparar los datos obtenidos con los datos planeados y evaluar la calidad del producto. Lo anterior se hace a través de la evaluación de los siguientes aspectos:

- Desempeño actual comparado con el plan generado.
- Lecciones aprendidas en el ciclo actual.
- Mejoras que pueden realizarse en los siguientes ciclos.
- Reportes de tamaños, tiempos y defectos.
- Informe del estado de la configuración.

- Ajuste a la planeación.
- Identificación de nuevos riesgos.

Los productos de entrada son:

- Productos de Pruebas
- Forma SUMP
- Forma SUMQ

No hay productos de salida. El responsable de esta actividad es el Administrador de calidad.

7.3 Evaluar roles de colegas

El propósito es evaluar la efectividad en el desempeño de cada uno de los roles (Cada miembro del equipo evalúa el desempeño de cada uno de los demás roles). Los pasos a seguir son los siguientes:

Registrar para cada rol:

- Dónde se encontraron problemas
- Formas de mejorar el proceso y nuevas metas para ciclos posteriores
- Realizar los reportes de rol de acuerdo a la ayuda y eficacia del mismo
- Registrar los datos obtenidos en la forma PEER

Participan todos los miembros del equipo y el responsable de que esta actividad se lleve a cabo es el líder de proyecto.

Los artefactos de salida son las formas PEER, que se presentan a continuación:

Evaluación del Administrador de Calidad

Actividades Principales:

1. **Asegurarse que todos los miembros del equipo reporten y usen apropiadamente los datos del proceso de TSPi**

Esta ha sido una tarea compartida con el líder de proyecto y con el administrador de planeación. Esto contribuyó a que este objetivo se cumpliera en forma y tiempo.

2. **Asegurarse de que el equipo siga fielmente el TSPi y producir un producto de calidad**

   Únicamente en las primeras etapas del primer ciclo fue necesario establecer los estándares, a fin de que todo el equipo presentara la documentación uniformemente y con calidad. No ha sido necesario hacer hincapié en este aspecto, en el ciclo 2 se efectuó una revisión completa de toda la documentación para cumplir satisfactoriamente este objetivo.

3. **Asegurarse que todas las inspecciones del equipo son bien realizadas y reportadas**

   Este objetivo se cumplió en su totalidad en el ciclo 2. Hubo un retardo en este aspecto debido a la alta complejidad del sistema, así como su tamaño.

4. **Asegurarse que todas las reuniones del equipo son reportadas con exactitud y que los reportes son colocados en la carpeta del proyecto.**

Todas las juntas que formalmente se han celebrado han quedado debidamente documentadas. Sin embargo, éste no ha sido el único medio de comunicación, ya que durante la fase de implementación se efectuaron comunicaciones vía mail, girando en torno al integrante del equipo que fungia como integrador del sistema

Tabla 7.1

Evaluación del Administrador de desarrollo

Actividades Principales:

1. **Guiar al equipo a producir la estrategia de desarrollo:**

   Se propuso una estrategia que fue aprobada y practicada exitosamente por cada miembro del equipo.

2. **Guiar al equipo al producir la primer estimación de tamaño y tiempo de desarrollo del producto:**

   Se produjeron métricas para tamaño y tiempo, que resultaron ser sumamente exactas.

3. **Guiar el desarrollo de la especificación de requerimientos:**

   Se definió y aplicó un cuestionario de especificación de requerimientos que se enriqueció en el momento de la entrevista con el cliente con preguntas adicionales que fueron de gran utilidad.

4. **Guiar al equipo al producir el diseño de alto nivel:**

   Se propuso y aprobó un diseño de alto nivel que fue complementado con comentarios y propuestas de los integrantes de Metasoft.

5. **Guiar al equipo al producir la especificación del diseño:**

   Esta tarea se cumplió satisfactoriamente. Cada miembro del equipo colaboró en la producción de la especificación del diseño en igual proporción.

6. **Guiar al equipo al implementar al producto:**

   Se organizó un curso de tecnologías y técnicas de desarrollo para así facilitar la implementación de código individual. Se asignaron trabajos de implementación de código personal.

7. **Guiar al equipo en el desarrollo de la construcción, integración y planes de prueba de sistema:**

   Cada módulo fue solicitado de manera tal que su integración con otros módulos fuera sencilla, inmediata y transparente. Se definieron los planes de prueba.

8. **Guiar al equipo en el desarrollo de materiales de prueba y correr las pruebas:**

   Se solicitaron a los integrantes de Metasoft realizar una serie de pruebas acorde al plan de pruebas.

9. **Guiar al equipo en la producción de documentación del**

producto para el usuario:

Uno de los miembros de METASOFT fue el encargado de generar esta documentación.

10. **Participar en la producción del reporte de desarrollo del ciclo:**

Se colaboró con todo lo solicitado para contribuir con el reporte del desarrollo del ciclo.

11. **Actuar como un ingeniero de desarrollo**

Durante todo el desarrollo del segundo ciclo, se ha contribuido con la integración de código, así como de los procesos que localmente prueban que éste es correcto

Tabla 7.2

Evaluación del Administrador de Planeación

Actividades Principales:

1. **Motivar al equipo para planear las tareas a desarrollar en el siguiente ciclo.**

   Debido a la alta motivación del equipo en la realización de este proyecto, la planeación de las tareas a desarrollar se dio siempre de forma voluntaria y anticipada por cada uno de los miembros de mi equipo y por el equipo en conjunto.

2. **Motivar al equipo para planear los tiempos y horarios para el siguiente ciclo de desarrollo.**

   Cada miembro del equipo periódicamente determinó las horas por semana que dedicaría al proyecto. Conscientes todos los miembros de la importancia de la planeación debido al poco tiempo con el que se cuenta para el desarrollo del proyecto, hubo siempre entusiasmo y cooperación en cuanto a la planeación de tiempos para el primer ciclo.

3. **Motivar al equipo para generar el plan del equipo consolidado.**

Esta tarea siempre tuvo alta prioridad entre los miembros del equipo, quienes siempre estuvieron de acuerdo en la importancia que representa el plan general del equipo. A partir de la etapa de desarrollo, el administrador de calidad generó semanalmente una "lista de tareas asignadas a cada miembro del equipo", identificando las habilidades y aptitudes de cada uno para la asignación más adecuada de las tareas y buscando siempre, dentro de lo posible, balancear la carga de trabajo con estos mismos criterios. A partir de esta asignación de tareas semanales se pudo obtener el plan personal detallado para cada ingeniero.

4. **Comparar el proceso del equipo con el plan trazado.**

   Esta tarea se realizó al finalizar el primer ciclo y se entregó a tiempo al instructor, el reporte de avances así como el consolidado final del equipo.

5. **Participar en la generación de reportes del ciclo.**

   Esta tarea se cumplió satisfactoriamente. Cada miembro del equipo realizó a tiempo los reportes que le fueron asignados, de acuerdo a su rol o de acuerdo a la actividad que el equipo en conjunto estaba realizando en ese momento. Por mi parte, participe activamente en la generación de reportes cada vez que así correspondía.

6. **Actuar como ingeniero de desarrollo**

Cuando se hizo necesario también se cumplió la tarea de ingeniero de desarrollo, procurando efectuar las tareas asignadas a tiempo y cumpliendo los estándares que el equipo había establecido

Tabla 7.3

Evaluación del Líder de Proyecto

Actividades Principales:

1. **Motivar a los integrantes a desarrollar sus tareas:**

   Esta no fue una tarea complicada y prácticamente no fue

necesario en ninguna ocasión efectuar ningún tipo de motivación, puesto que todos los integrantes del equipo efectuaban sus tareas con responsabilidad y entusiasmo, buscando el beneficio del equipo en general.

2. **Cada semana, efectuar una reunión del equipo:**

   Desde el inicio de la construcción del sistema, esta política se ha venido manejando apropiadamente, una de las razones es que todo el grupo esta a tiempo completo. A pesar de que existieron ocasiones en que no me fue posible asistir a estas reuniones, el resto del grupo procuro reunirse para evaluar:

   Las tareas cumplidas

   Llenado de formas en el tiempo establecido

   Verificar el estado del Proyecto

   Asignar tareas para la siguiente semana

3. **Semanalmente, reportar el estado del equipo al instructor**

   Aunque esta tarea era una responsabilidad personal, siempre se prefirió escoger a la persona mas idónea de acuerdo a la fase que estábamos atravesando; indicando el estado del equipo al instructor y resto de compañeros.

   Con respecto a la entrega de la carpeta del proyecto, siempre se entrego a tiempo, solo hubo un caso excepcional en que se decidió postergar la entrega hasta dos días después para completar la información.

   No fue necesario en ninguna ocasión buscar asesoramiento del instructor por falta de cumplimiento de tareas del resto del equipo.

4. **Ayudar al equipo en la asignación de tareas y resolución de otros asuntos**

   Esta tarea no fue cumplida apropiadamente.

   El Administrador de calidad fue la persona encargada de asignar tareas al resto del equipo y el Administrador de Desarrollo en la mayoría de casos fue la persona encargada de resolver los

problemas técnicos.

5. **Mantener la carpeta del proyecto**

   Esta tarea se cumplió satisfactoriamente. Inclusive se creo un repositorio compartido donde cada miembro del equipo podía depositar allí sus versiones creadas.

6. **Actuar como ingeniero de desarrollo**

Cuando se hizo necesario, también se cumplió la tarea de ingeniero de desarrollo, procurando efectuar las tareas asignadas a tiempo y cumpliendo los estándares que el equipo había establecido

Tabla 7.4

7.4 Revisar productos

El propósito de ésta actividad es revisar el cumplimiento de estándares, detectar y corregir defectos en los productos. Para llevar a cabo esta actividad es necesario imprimir los productos a revisar, tomar un inciso de la lista de verificación y dedicarse a su revisión en todo el producto, resaltar con marcador los defectos encontrados, realizar el proceso anterior hasta el final de la lista Entregar al propietario del producto para realizar la corrección.

Los artefactos de entrada son:

- Lista de verificación
- Estándar de documentación
- Formas PEER
- Reporte del ciclo

Los documentos de salida (o productos de Post mortem) son:

- Formas PEER
- Reporte del ciclo

El responsable de la realización de estas actividades es el administrador de calidad. A continuación se presenta el reporte del ciclo:

Reporte del Ciclo

Para el primer ciclo de desarrollo se definieron las siguientes actividades a cumplir:

1) Definir presentación de formas HTML con hojas de estilo

Esta tarea se cumplió satisfactoriamente

2) Usar el manejador de base de datos seleccionado por el Cliente

Todas las sentencias SQL's fueron transformadas para que funcionen en el manejador de Base de Datos proporcionado por el cliente.

3) La aplicación se logró hacer funcionar en cualquier explorador de Internet

Se hicieron las investigaciones necesarias para encontrar las funciones apropiadas para que la aplicación trabaje de manera semejante tanto en Internet Explorer como Netscape.

Además de las tareas que se pretendían cubrir en este ciclo de desarrollo, se cumplieron los objetivos propuestos al iniciar la aplicación, estos son:

Producir un producto de calidad

1. Funciones de requerimientos incluidas al finalizar el proyecto: 100%

Mantener un proyecto bien administrado

1. Error en la estimación del tamaño del producto < 20%
2. Error en la estimación de horas de desarrollo < 20%
3. Porcentaje de datos registrados e incluidos en la carpeta del proyecto 100%

Finalizar el primer ciclo en la fecha acordada Número de días de retraso / adelanto en la finalización del ciclo < 4

Tabla 7.5

A continuación se presenta el último producto de la fase de post mortem para el primer ciclo de TSPi. La forma PEER:

Forma PEER

For each role, evaluate the work required and the relative difficulty in % during this cycle.		
Role	Work Required	Role Difficulty
Team Leader	**20%**	**20%**
Development Manager	**25%**	**20%**
Planning Manager	**20%**	**20%**
Quality/Process Manager	**25%**	**20%**
Support Manager	**10%**	**20%**
Total Contribution (100%)	**100%**	

Rate the overall team against each criterion. Circle one number from 1 (inadequate) to 5 (superior)					
Team spirit	1	2	3	4	5
Overall effectiveness	1	2	3	4	5
Rewarding experience	1	2	3	4	5
Team productivity	1	2	3	4	5
Process quality	1	2	3	4	5
Product quality	1	2	3	4	5

Rate role for overall contribution. Circle one number from 1 (inadequate) to 5 (superior)					
Team Leader	1	2	3	4	5
Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

Rate each role for helpfulness and support. Circle one number from 1 (inadequate) to 5 (superior).

Team Leader	1	2	3	4	5
Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

Rate each role for how well it was performed. Circle one number from 1 (inadequate) to 5 (superior).					
Team Leader	1	2	3	4	5
Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

7.5 Reporte de HRT's

En esta sección, se presenta un consolidado general de las HRT's generadas a lo largo del primer ciclo del proceso de desarrollo TSPi extendido, propuesto en el capítulo 2 de este trabajo. Los objetivos son los siguientes:

- Consolidar el repositorio de conocimientos adquiridos para el beneficio de la organización.
- Ofrecer una referencia de consulta para la resolución de problemas técnicos y toma de decisiones en proyectos posteriores.
- Promover la integración y reuso de las HRT's en el proceso de desarrollo de una organización, vía la presentación de datos estadísticos que muestran los beneficios de la inclusión de este artefacto.

Para lo anterior, conviene mencionar que en el ciclo 1 se generaron 31 HRT's distribuidas de la siguiente manera:

- Por estado:
  - HRT's aceptadas 95%

- HRT's rechazadas 03%
- HRT's pendientes o sin responder 02%

- Por objetivo:
  - HRT's que impactan calidad 32%
  - HRT's técnicas 67%
  - HRT's para la toma de decisiones 21%

  La suma de porcentajes en el rubro anterior es mayor al 100% debido a que varias recomendaciones (tanto técnicas como para la toma de decisiones) impactan a la calidad.

- Por independencia de proyecto:
  - HRT's específicas para SIPU 17%
  - HRT's independientes de SIPU 83%

Dado el punto anterior, podemos ver que, con base en los documentos producidos al responder una HRT, el conocimiento así generado puede ser de utilidad en otros proyectos, ya que estas recomendaciones funcionaron y fueron utilizadas previamente de manera exitosa. Adicionalmente, este conocimiento puede ser incrementado o enriquecido con desarrollos subsecuentes.

Los datos anteriores muestran que las HRT's recopilan la experiencia y los conocimientos de personas que pueden incluso ya no pertenecer a la organización. Sin embargo, de cualquier manera este conocimiento será accesible a nuevos integrantes de la misma, reduciendo así costos en términos de capacitación y contratación de personal altamente especializado.

Cabe mencionar que, si se posee una adecuada administración en la actualización de la base de conocimientos, se tendrá la garantía de que la organización estará utilizando en todo momento técnicas de vanguardia, tanto en el aspecto tecnológico como en el de la toma de decisiones.

Finalmente, la realización de HRT's ofrece un apoyo en la planeación de tiempos para nuevos proyectos, ya que varias de las actividades que se incluyen en un plan de desarrollo pueden estar ya resueltas y documentadas en

una HRT, permitiendo así proponer tiempos muy precisos en la implementación de las mismas y ahorrando tiempo y esfuerzo en su elaboración. A continuación se presenta la tabla 7.4 que describe el volumen de HRT's generadas en cada fase y en seguida se justifica la presentación de la misma:

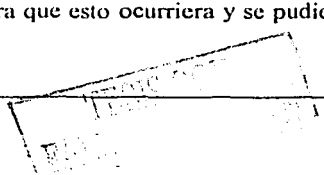
Flujo de trabajo	Fase	# HRT's	Porcentaje
Concepción	Factibilidad	3	9,68
	Lanzamiento	1	3,23
Elaboración	Estrategia	3	9,68
	Planeación	0	0,00
	Requerimientos	3	9,68
Construcción	Análisis	5	16,13
	Diseño	5	16,13
	Implementación	7	22,58
	Pruebas	3	9,68
Transición	Post mortem	1	3,23

Tabla 7.6

La tabla anterior muestra que en nuestro caso de estudio (**SIPU©**) el mayor número de recomendaciones fue generado en la fase de implementación. Esto es natural debido a que en el momento en el que se desarrolló el proyecto, no se tenía conocimiento de ningún tipo de tecnología o implementación de la misma.

De igual manera, el esfuerzo en las fases de análisis y diseño fue grande, debido principalmente a la falta de experiencia en la abstracción y modelación de los requerimientos.

Las fases de factibilidad y requerimientos presentan un menor número de recomendaciones y esto es debido a que el proyecto fue escogido de manera previa –por los instructores- para que esto ocurriera y se pudieran cumplir las expectativas académicas.



Finalmente, las fases que incluyeron un menor número de recomendaciones fueron las de estrategia, lanzamiento, post mortem y planeación, ya que el equipo de desarrollo, en esas fases, se apegó fielmente a la metodología TSPi y no hubo necesidad de generar demasiadas recomendaciones.

A continuación se presenta una gráfica útil en la visualización del conjunto de datos presentados en la tabla 7.6:

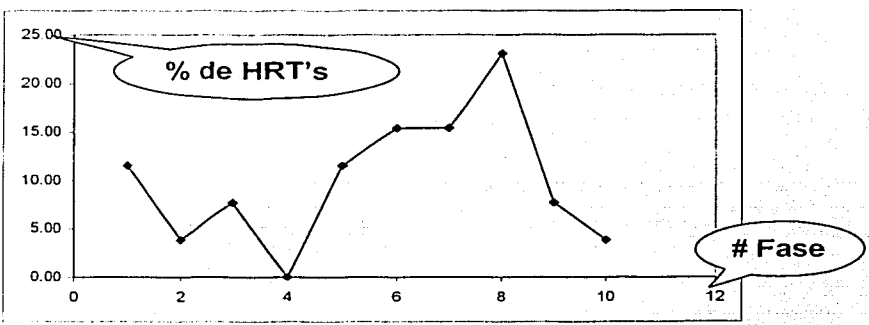


Figura 7.1

Para proyectos similares al caso de estudio **SIPU©** en dónde la tecnología requerida sea la misma (J2EE), se espera que el porcentaje para la fase de implementación disminuya como consecuencia de la reutilización del conocimiento generado a través de las HRT's.

De igual manera se espera que (para proyectos posteriores) el porcentaje de HRT's para las fases de análisis y diseño disminuya, si el uso o finalidad del software es similar al del caso de estudio.

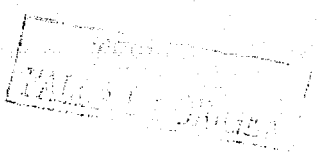
En general se espera que, después de la realización de varios proyectos, las HRT's de los desarrollos subsiguientes puedan ser de utilidad para otros y así llegar a generar gráficas como la de la figura 7.1, pero con una curva asintótica al eje horizontal.

Con ésta última hipótesis concluye la sección final del capítulo 7 dedicado al flujo de transición.

7.6 Conclusiones del capítulo VII

En esta fase, se deberá hacer un análisis en retrospectiva de la forma en la que se llevaron a cabo las actividades a lo largo del ciclo, identificando posibles fallas en el seguimiento, administración y realización del desarrollo. Esto con el fin de documentar los puntos a mejorar en el próximo ciclo o en su caso, el próximo desarrollo.

Adicionalmente se propone realizar un conteo de las recomendaciones técnicas generadas a lo largo del ciclo con el fin de mantener una estadística de uso de las mismas y así poder medir adecuadamente el impacto de estas en el desarrollo.



8

Siguientes ciclos

8.1 Introducción

En el presente capítulo se presentará una guía a seguir en los siguientes ciclos basada en la estructura del proceso de desarrollo que se presentó en el capítulo 2 y que es una modificación del proceso TSPi. La estructura antes mencionada se presenta a continuación:

Flujo de Concepción
- Fase de Factibilidad
- Fase de Lanzamiento

Flujo de Elaboración
- Fase de Estrategia
- Fase de Planeación
- Fase de Requerimientos

Flujo de Construcción
- Fase de Análisis
- Fase de Diseño

Fase de Implementación y pruebas unitarias
Fase de Pruebas de integración
Flujo de Transición
Fase de Mejora Continua

8.2 Flujo de Concepción

En los siguientes ciclos se esperan realizar pocas actividades en el flujo de concepción. Dado que el proyecto ya ha iniciado y el equipo está formado, se debe procurar mantener esta formación y realizar pocas consideraciones con respecto a la viabilidad del proyecto.

8.2.1 Fase de Factibilidad

La principal actividad en ésta fase consiste en evaluar si las condiciones que determinaron la viabilidad del proyecto en el primer ciclo, persisten.

Las actividades de esta fase para los siguientes ciclos son cinco:

	Objetivos de esta fase para el primer ciclo	**Actividades a realizar en el siguiente ciclo**
1	Establecer el alcance del proyecto y los límites de éste, incluyendo una visión operacional, criterios de aceptación y que es lo que se desea tener como producto final y que no.	Afinar el alcance del proyecto. Es deseable no extender la funcionalidad del mismo.
2	Discriminar los casos de uso críticos para el sistema, los escenarios de operación primarios que van a definir los aspectos mayores del diseño.	Confirmar que no hay mas casos de uso que los identificados en el primer ciclo.
3	Exhibir al menos una arquitectura candidata para alguno de los escenarios primarios.	Consolidar la elección de la arquitectura aplicándola a mas escenarios.
4	Realizar un estimado general del tiempo y costo para el siguiente ciclo.	Confirmar que el desarrollo se encuentra en tiempo y que el costo no ha cambiado.

5	Estimar riesgos potenciales.	Identificar nuevos riesgos y generar un plan de mitigación de los mismos

Tabla 8.1

8.2.2 Fase de Lanzamiento

Para la fase de Lanzamiento, el principal aspecto a considerar es el caso de que en el equipo se requiera reemplazar algún recurso humano y el secundario es el que se requiera renovar o actualizar las licencias de productos o equipos que se han estado utilizando en el desarrollo.

En caso de que se requiera reemplazar algún recurso humano, también se deberá definir el rol del mismo dentro del equipo de desarrollo y, naturalmente, reasignar tareas al nuevo recurso.

8.3 Flujo de Elaboración

Las actividades a realizar en el flujo de Elaboración son básicamente las mismas para el primer y para el segundo ciclo. La diferencia consiste en que en el segundo ciclo ya están definidas las metas personales y de equipo. Además, la metodología de trabajo y la definición de la organización y distribución de actividades ya se ha establecido.

8.3.1 Fase de Estrategia

A continuación se presentan las actividades de la fase de Estrategia para ciclos posteriores, y un breve comentario acerca de lo que procede para ellas en los ciclos subsiguientes.

	Actividades de la fase de estrategia para los siguientes ciclos.	**Guía para realizar la actividad**
1	Revisar estrategia.	Basarse en la estrategia definida en el ciclo anterior.

2	Actualizar estrategia de desarrollo.	Incluir (o en su caso, eliminar) nuevos criterios estratégicos.
3	Producir actualizaciones de estimados de tamaño y tiempo.	Recalcular tiempos y tamaños con base en la información del ciclo anterior.
4	Identificar riesgos.	Documentar riesgos que no habían sido detectados en el ciclo anterior.
5	Documentar estrategia.	N/A
6	Revisar y actualizar plan de configuración.	N/A

Tabla 8.2

8.3.2 Fase de Planeación

Para la fase de planeación, al igual que en el primer ciclo, se generarán los siguientes documentos:

- Forma SUMS
- Forma TASK
- Forma SCHEDULE
- Forma SUMQ
- Planes Individuales

Cada uno de estas formas contendrá información actualizada – muy similar- a la ejemplificada en los documentos de la fase de Planeación del capítulo cinco, pero ahora deberá estar apegada a los tamaños y tiempos que se requieran para el nuevo ciclo.

8.3.3 Fase de Requerimientos

La siguiente es una lista de las actividades a realizar en la fase de Requerimientos para los ciclos posteriores al primero:

	Actividad	Descripción
1	Revisión del enunciado del problema.	El administrador de desarrollo guía al equipo a reexaminar el enunciado del

		problema y formular nuevas cuestiones acerca de: • Las funciones a ser realizadas por esta nueva versión del producto. • Cómo estas funciones van a ser usadas.
2	Clarificación del enunciado del problema.	El administrador de desarrollo realiza preguntas al cliente, quién las responde frente al equipo de desarrollo.
3	Actualización de tareas.	El administrador de desarrollo guía al equipo a: • Identificar los cambios a los requerimientos que se deben hacer. • Actualizar las ubicaciones de los componentes funcionales.
4	Redistribución de tareas.	El líder de proyecto ayuda a distribuir las tareas entre los miembros del equipo y establece un compromiso de entrega con ellos.
5	Actualización de documentación.	Cada miembro del equipo actualiza su SRS y se lo entrega al administrador de desarrollo. El administrador de desarrollo consolida los diferentes SRS's.
6	Revisión y actualización del plan de pruebas del sistema.	El administrador de desarrollo guia al equipo en la revisión y actualización del plan de pruebas para el sistema.
7	Actualización de inspecciones.	El administrador de calidad guía al equipo a: • Inspeccionar la forma SRS y el plan de pruebas del sistema (ver forma INS) • Identificar preguntas y problemas • Definir quién, cuándo y cómo va a resolver las preguntas. • Documentar la inspección vía la

		forma INS.
8	Actualización de requerimientos.	El administrador de desarrollo genera el documento final de SRS y verifica el seguimiento de el enunciado del problema.
9	Revisión del la forma SRS por el usuario.	El administrador de desarrollo entrega una copia del SRS al cliente y posteriormente, el equipo arregla cualquier problema identificado.
10	Actualizar línea base.	El administrador de soporte genera la línea base para la forma SRS actualizada.

Tabla 8.3

8.4 Flujo de Construcción

Para los ciclos posteriores al primero, en el flujo de Construcción, se generarán los mismos documentos que en el primer ciclo, pero ahora incluyendo las nuevas adiciones identificadas durante la fase anterior, la de requerimientos.

8.4.1 Fase de Análisis

A continuación se presentan las actividades a realizar en la fase de análisis incluyendo los artefactos resultantes de la misma y el rol asignado a tal actividad:

	Actividad	**Rol**	**Artefacto resultante**
1	Revisar la forma SRS	LP	N/A
2	Refinar diagramas de casos de uso	LP	Diagramas de casos de uso
3	Generar diagramas de actividades	LP	Diagramas de actividades
4	Definir nuevas clases y generar diagramas de clases asociados.	AD	Diagramas de clases

5	Rehacer diagramas de paquetes.	AD	Diagramas de paquetes
6	Rehacer diagramas de secuencia.	AD	Diagramas de secuencia

Tabla 8.4

En este punto, es conveniente hacer notar que los diagramas anteriores intentan modelar las reglas de negocio plasmadas en el documento SRS, y que la forma de implementación de las mismas, no está contemplada en los diagramas o artefactos resultantes. Esa responsabilidad recae en la fase de diseño.

8.4.2 Fase de Diseño

Como se mencionó en la sección anterior, en la fase de diseño se pretende indicar, vía diagramas de UML, la forma en que debe ser realizada la implementación técnica de las reglas de negocio definidas en el documento SRS.

La mayoría de los documentos o artefactos que se generarán en ésta fase son diagramas de UML (básicamente los mismos que en la fase anterior) que incluirán un alto nivel de detalle en términos técnicos y altamente influenciados por la tecnología empleada.

El objetivo de esta fase (al igual que en los demás ciclos) es facilitar la implementación de código en la siguiente fase: la fase de Implementación y pruebas. Con base en los diagramas generados en la fase de diseño debería ser posible lograr una implementación prácticamente directa y sin errores.

8.4.3 Fase de Implementación y pruebas unitarias

Actividades del flujo de Implementación y pruebas:

	Actividades	Descripción
1	Planeando la implementación	El AD dirige al equipo de trabajo a: • Definir y planear las tareas de implementación (SUMP, SUMQ).

2	Asignación de tareas	El LE ayuda a asignar las tareas entre los integrantes y a fijar las fechas de terminación.
3	Diseño detallado	Los ingenieros producen el diseño detallado. • Revisan el diseño utilizando métodos de revisión de diseño esmerados. • Completan las formas LOGD (registro de defectos) y LOGT (registro de tiempo).
4	Plan de pruebas unitarias	Los ingenieros producen el plan de pruebas unitarias.
5	Desarrollo de pruebas	Los ingenieros se basan en el guión de pruebas unitarias (PU) para desarrollar los casos de pruebas unitarias, procedimientos de pruebas y datos de pruebas.
6	Inspección de diseño detallado	El AQ dirige al equipo en la inspección del diseño detallado de cada componente (guión INS (inspección), formas INS (reporte de inspección) y LOGD (registro de defectos).
7	Código	Los ingenieros producen el código fuente de cada componente. • Se hace la revisión de código usando una lista de verificación personal. • Se compila y arregla el código hasta que compile sin un error. • Se completan las formas LOGD y LOGT.
9	Inspección de código	El AQ dirige al equipo en la inspección de código de cada componente (guión INS, formas INS y LOGD).

10	Pruebas unitarias	Los ingenieros basándose en el guión de pruebas unitarias conducen dichas pruebas y completan las formas REGD y REGT.
11	Revisión en la calidad de componentes	El AQ revisa los datos de cada componente para determinar si reúne los criterios de calidad del equipo. Si es así, el componente es aceptado para las pruebas de integración. De lo contrario, el AQ recomienda que: • El producto sea inspeccionado nuevamente y que se apliquen las modificaciones pertinentes. • El producto se descomponga en piezas y sea desarrollado de nuevo.
12	Liberación de componentes	Cuando los componentes son inspeccionados e implementados en forma satisfactoria, los ingenieros lo entregan al Administrador de configuración. El Administrador de configuración registra los componentes en el sistema de administración de configuración.

Tabla 8.5

Se consideran los siguientes criterios de éxito:

- Componentes terminados, inspeccionados y bajo control de configuración.
- Las formas INS de diseño e inspecciones de código completadas.
- Planes de pruebas unitarias y materiales de apoyo realizadas.
- Las formas SUMP, SUMQ, SUMT, LOGD y LOGT actualizadas.
- La carpeta del proyecto actualizada.

Con esto concluye la fase de Implementación y prueba unitarias para ciclos posteriores al primero.

8.4.4 Fase de Pruebas de integración

Actividades del flujo de Pruebas de integración:

	Actividad	Descripción
1	Desarrollo de pruebas	El AD dirige el desarrollo de las pruebas. El LE ayuda en la asignación de tareas para desarrollar las pruebas entre los integrantes. • Se definen procesos y procedimientos requeridos en la construcción. • Se desarrollan procedimientos y facilidades de pruebas de integración y del sistema. • Se mide el tamaño y tiempo de ejecución de cada prueba. • Se revisan los materiales de pruebas y se corrigen errores.
2	Construcción	El equipo construye el producto y revisa que esté completo. • Se verifica que las partes necesarias estén disponibles. • Se construye el producto y se pone a disposición para la prueba de integración. • El propietario registra los defectos en LOGD.
3	Integración	El AD dirige al equipo en el desarrollo de las pruebas de integración. • Se revisa que el producto y las pruebas de integración estén completos. • Se registran todas las actividades de pruebas en la forma LOGP. • El propietario del producto registra los defectos en LOGD.

4	Pruebas del sistema	El AD dirige al equipo en el desarrollo de las pruebas del sistema. • Se prueba el producto en condiciones normales y de estrés. • Se prueba el producto en cuanto a instalación, conversión y recuperación. • Se aplican pruebas de regresión al sistema. • Se registran todas las actividades de pruebas en la forma LOGP. • El propósito del producto es registrar defectos en LOGD.
5	Documentación	El AD dirige al equipo en: • La actualización de la documentación del usuario y las tareas. • La asignación de esas tareas como parte de la documentación del equipo. • La revisión del esquema con el equipo de pruebas. • La generación del borrador de los documentos del usuario para ciclos posteriores. • La revisión, corrección y generación de la documentación del usuario.

Tabla 8.6

Criterios de éxito:

- Un sistema, del segundo o de ciclos posteriores, integrado y probado.
- Las formas LOGD y LOGP de todas las pruebas.
- La documentación del usuario revisada y actualizada.
- Los datos de tiempo, tamaño y defectos registrados.

8.5 Flujo de Transición

En este flujo – que únicamente se compone de la fase de Mejora continua - se realiza una reflexión en retrospectiva de los productos, actividades y resultantes del proceso llevado a cabo.

8.5.1 Fase de Mejora continua

Actividades a realizar para la fase de Mejora continua:

	Actividades	Descripción
1	Revise los datos del proceso	El AC dirige al equipo a: • Analizar los datos del proyecto e identificar áreas con problemas. • Evaluar las propuestas de mejora establecidas en las formas PMP de ciclos anteriores. Se identifica dónde es necesario realizar mejoras, de acuerdo a: • Liderazgo, planeación, proceso, calidad o soporte. • Toma de acciones y responsabilidades en equipo. • Mejoras por parte del instructor. Se preparan y entregan las formas PMP con las sugerencias indicadas.
2	Evalúe el desempeño del rol	El LP dirige la evaluación sobre la efectividad de los roles, las actividades del instructor y las facilidades de soporte. • Dónde generaron mayor beneficio. • Dónde se pueden hacer mejoras.
3	Prepare el reporte del ciclo n	El LP dirige al equipo en la generación del reporte del ciclo n. • Se asignan actividades a cada integrante para generar tal reporte. • Se establecen acuerdos para entregar el documento asignado. • Se integra, revisa y corrige el reporte terminado.

4	Prepare las Evaluaciones de Roles	Cada ingeniero entrega la forma EEC completada • En cuanto a dificultad y contribución de cada rol. • Con porcentajes que deberían sumar cien por ciento. • La efectividad de cada rol, calificada desde 1 (uno = inadecuado) hasta 5 (cinco = superior).

Tabla 8.7

Los criterios de éxito son los siguientes:

- El ciclo de desarrollo ha generado un producto de alta calidad y la documentación necesaria.
- El producto terminado está bajo control de configuración.
- Los datos del proceso han sido evaluados y las formas PMP entregadas.
- Las evaluaciones del equipo entre colegas se realizaron y entregaron (EEC).
- El reporte del ciclo n se terminó y entregó.
- Las formas RESPL y RESCA han sido completadas, tanto para el sistema y sus partes componentes.
- La carpeta del proyecto ha sido actualizada.

Con esto concluye la fase de Mejora continua (o post mortem) de TSPi y el flujo de transición del proceso de desarrollo.

8.6 Conclusiones del capítulo VIII

La mayoría de las actividades a realizar en los ciclos posteriores al primero son similares a éste y en general, se reducen para las fases de Factibilidad y Lanzamiento, se mantienen para las fases de Análisis, Diseño e Implementación y se incrementan para las fases de Requerimientos, Implementación y Pruebas.

La mayoría de las actividades a realizar propuestas están documentadas en este trabajo y adicionalmente, un detalle de las mismas puede ser consultado en TSPi[1].

Conclusiones

Introducción

Debido a la gran diversidad que existe en el desarrollo de software, es difícil establecer criterios de implementación estándares para todo proyecto. Cada desarrollo posee características únicas que lo hacen diferente de cualquier otro.

Una forma de adquirir capacidad de desarrollo es a través de la experiencia y la acumulación de elementos o técnicas reutilizables en nuevos proyectos que conllevarán a la generación de software de mayor calidad de manera administrada, controlada y repetible.

Propósito

En este trabajo se presenta una propuesta que extiende al proceso de desarrollo al incluir en él un nuevo artefacto cuyo propósito es incrementar la capacidad

de desarrollo de la organización y la generación de productos de software apegados a un estándar de calidad.

Este propósito se verifica al observar el cumplimiento de los objetivos primarios de la propuesta:

- **Definir el artefacto 'Hojas de Recomendaciones Técnicas'.** Esto fue realizado en el capítulo III de este documento. Se propuso, para la elaboración de las mismas, una sección administrativa con datos útiles en para el control y seguimiento de las mismas; y una sección de datos particulares que detallan la recomendación.

- **Proponer la integración de las mismas al proceso de desarrollo de software y justificar el porqué de ésta propuesta.** En la sección 3.1 del capítulo III se propone la integración de las Hojas de Recomendaciones Técnicas y esta propuesta se justifica a través de las responsabilidades de las mismas:

  - ✓ Apoyar al plan de mitigación de riesgos generado en alguna fase, o mitigar riesgos que se identificaron posterior a dicho plan.
  - ✓ Servir como base o referencia para la toma de decisiones en el desarrollo del sistema.
  - ✓ Servir como referencia técnica en posteriores desarrollos, minimizando así tiempos y costos, reutilizando el conocimiento generado previamente.
  - ✓ Incrementar la capacidad de desarrollo de la organización reduciendo la dependencia de recursos humanos especializados.

- **Hacer evidente el impacto en la calidad de software que se genera haciendo uso de buenas técnicas de desarrollo en conjunción con el conocimiento de tecnologías de punta y mostrar que las HRT's influyen en el proceso de toma de decisiones durante el desarrollo de un sistema.** Lo anterior se desprende de la estructura de las HRT's, que incluye apartados dedicados a justificar los beneficios, desventajas, e impacto en calidad que inyectan a un proyecto, así como también hay

apartados que definen la propuesta y su estrategia de implementación. Esto es justificado con mayor detalle en el apartado "Ejemplos".

- **Ilustrar vía un caso de estudio la influencia de esta propuesta en la calidad del producto final.** Esto fue realizado a lo largo de los capítulos IV, V, VI y VII al incluir en cada uno de ellos todas las HRT's que surgieron durante el desarrollo del proyecto SIPU. Con esto se intenta hacer evidente que cada una de las propuestas impactó de cierta manera al desarrollo y a la calidad (en términos del ISO/IEC 9126-1) del mismo. Para esto último es posible verificar el apartado *impacto en calidad* que cada propuesta contiene.

De esta manera, podemos concluir que la integración del contexto técnico y tecnológico en un proceso de desarrollo permitirá desarrollar software de manera cada vez más sistemática evitando rehacer trabajo y reduciendo la dependencia de recursos humanos especializados. Con esta técnica se espera que, para proyectos posteriores, el porcentaje de HRT's para las fases de análisis, diseño e implementación disminuya, si el uso o finalidad del software es similar en estos nuevos desarrollos.

En general se espera que, después de la realización de varios proyectos (de distintos tipos) las HRT's de los desarrollos subsiguientes puedan ser de utilidad para otros y así llegar a generar cada vez menos HRT's. Dicho de otra manera, esta generación de recomendaciones deberá decrecer conforme la organización vaya liberando más sistemas y eventualmente, será mínimo el número de recomendaciones a generar, dado el acervo adquirido.

Ejemplos

Para sustentar lo anterior, las HRT's se han probado en la práctica exitosamente con cuatro desarrollos que poseen características similares llamados A, B, C y D respectivamente. El siguiente es un resumen de la información recabada.

i. En los cuatro proyectos se utilizaron el 100% de las clases propuestas en las HRT's del proyecto "A" como "de soporte a bajo nivel".

ii. El equipo de desarrollo para cada proyecto fue diferente, pero el proceso de desarrollo que integró esta propuesta fue el mismo, lo que muestra que la misma es independiente del conocimiento y experiencia de desarrolladores especializados.

iii. Si el número de HRT's generadas para el proyecto X es n(X), entonces se ha observado que:

$$n(A) > n(B) > n(C) > n(D)$$

iv. Si el número de HRT's reutilizadas para el proyecto X es r(X), entonces se ha observado que:

$$r(A) < r(B) < r(C) < r(D)$$

De lo anterior, se puede observar que la disciplinada inclusión de recomendaciones técnicas al conjunto de documentos que integran un desarrollo, facilita la reutilización de las mismas en desarrollos posteriores y reduce la dependencia que existe entre una organización y los recursos humanos especializados de ésta.

Otro aspecto importante a considerar a favor de este proceso de adquisición de conocimientos es que reduce de manera importante la curva de aprendizaje e investigación por parte de los nuevos recursos humanos, que se vayan integrando a la organización.

Lo anterior repercute directamente en la calidad del producto, ya que este mecanismo permite analizar y en su caso, actualizar o mejorar una propuesta con el fin de generar, posteriormente, productos superiores.

Adicionalmente, este mecanismo es independiente de cualquier proceso de desarrollo y por ello es fácilmente integrable a la labor de desarrollo de cualquier organización.

Trabajos a futuro

En la práctica se ha demostrado que esta estrategia de inclusión de recomendaciones técnicas al curso del desarrollo ha sido exitosa y se propone como trabajo a futuro recabar estadísticas de más desarrollos que reafirmen esta propuesta.

Una actividad interesante a futuro, sería la refinación del aplicativo AHRTe, para que se incrementen sus capacidades de operación así como la habilidad de interactuar con otros programas afines. Una inclusión interesante sería la capacidad de poder intercambiar información técnica entre organizaciones.

A

Manual de usuario de SIPU

A.1 Introducción

La Coordinación de la Investigación Científica de la UNAM a través de la Secretaría de Investigación y Desarrollo tiene como objetivo, entre otros, desarrollar proyectos de investigación con carácter prioritario para México.

Con este propósito la Secretaría de Investigación y Desarrollo cuenta con las siguientes áreas: Dirección de Programas Universitarios, Dirección para el Desarrollo de la Investigación, Coordinación de Plataformas Oceanográficas y Coordinación de Servicios de Gestión y Cooperación Académica.

En particular, la Dirección de Programas Universitarios (DPI) tiene asignada la tarea de canalizar y coordinar actividades de investigación interdisciplinarias. Con este objetivo, surge la necesidad de contar con una base de datos que contenga toda la información relacionada con los proyectos

de investigación que se desarrollan en la UNAM, y en particular, aquella información relacionada con los temas y líneas de investigación abordados en cada uno, así como los recursos con los que actualmente se cuenta para llevarlos a cabo.

Además de los proyectos de investigación asignados directamente a cada centro de investigación, la DPI dirige cinco Programas Universitarios encargados de coordinar los recursos de investigación (instituciones e investigadores) con que cuenta la UNAM.

Estos Programas que actualmente se encuentran conformados se enumeran a continuación:

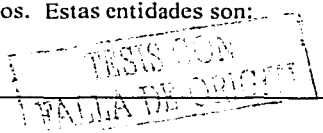
PUAL	Programa Universitario en Alimentos
PUCIM	Programa Universitario de Ciencia e Ingeniería de Materiales
PUE	Programa Universitario de Energía
PUIS	Programa Universitario de Investigación en Salud
PUMA	Programa Universitario de Medio Ambiente

A.2 El Módulo de Consultas

El sistema de cómputo denominado Sistema de Información para Programas Universitarios (**SIPU©**) auxiliará a la DPI y a cada uno de sus Programas en su labor.

El **SIPU©** cuenta con una base de datos que contiene toda la información mencionada anteriormente. El mantenimiento y alimentación de la información contenida en dicha base de datos se encuentra bajo la responsabilidad directa de cada Programa Universitario.

En particular, el Módulo de Consultas por Internet del **SIPU©** brindará a sus usuarios toda la información contenida en la base de datos de forma lógica y estructurada. La manera como lleva a cabo este objetivo es por medio de consultas por criterios de filtrado para cada una de las entidades principales contenidas en la base de datos. Estas entidades son:



- ✓ Líneas de Investigación
- ✓ Proyectos de Investigación
- ✓ Dependencias Universitarias
- ✓ Investigadores
- ✓ Equipos
- ✓ Servicios

Adicionalmente, este sistema amplía sistemáticamente la información mostrada en sus páginas por medio de ligas de Internet, las cuales implican nuevas peticiones de consulta que atenderá el sistema.

Por último, los consultantes de este sistema podrán dejar un registro de sus datos personales, a fin de que el Programa Universitario en cuestión mantenga un contacto más estrecho con los interesados.

A.3 Requerimientos

Debido a que esta aplicación funciona a través de Internet, se requiere tener instalado un navegador y una conexión a Internet. Los navegadores probados para esta aplicación son Microsoft Internet Explorer 5 y Netscape 4. Adicionalmente deberá verificarse que el navegador elegido tenga activada su capacidad para interpretar código JavaScript (esta opción se encuentra activada por omisión).

Nota: Las imágenes del sistema que se presentan en este manual pueden diferir de las actuales.

A.4 Uso de la aplicación

En esta sección se describe la forma en la que la aplicación puede ser utilizada. Esta descripción se basa en la operación cotidiana del sistema y está organizada de manera que primero se presenten las funcionalidades generales y después, las particulares.

A.4.1 Entrada al Sistema de Consultas

Cada Programa Universitario poseerá una liga hacia esta aplicación. Sin embargo, será posible ingresar directamente a ésta, escribiendo la dirección URL en donde se encuentre instalada.

En cualquiera de estos casos se presentará la página de presentación del Módulo de Consultas de **SIPU©** (Figura 1).

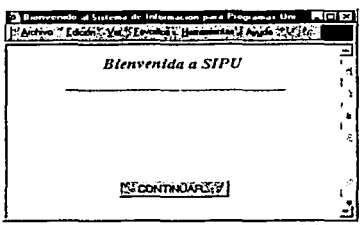


Figura 1. Página de presentación del Módulo de Consultas

La siguiente página requiere del usuario que se caracterice mediante las opciones que se ofrecen en el cuadro de lista de consultantes; o bien, que se registre, para lo cual se le pedirá información personal necesaria para establecer comunicación en un futuro si así lo desea (Figura 2).

Una vez realizada esta operación (caracterización o registro del usuario), se mostrará la página principal de consultas (Figura 4)

A.4.2 Registro de Usuarios

Si así lo decide, el consultante podrá registrarse al entrar a la página inicial (Figura 2), o bien mediante la liga que se ofrece en la porción izquierda de la página principal de consultas (Figura 4). Los datos que se marcan como obligatorios en el formulario de registro son indispensables para poder registrar al usuario, de lo contrario, no se permitirá ejecutar esta operación (Figura 3).

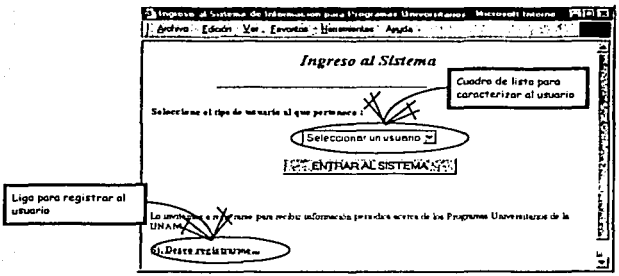


Figura 2. Página de caracterización / registro de usuarios

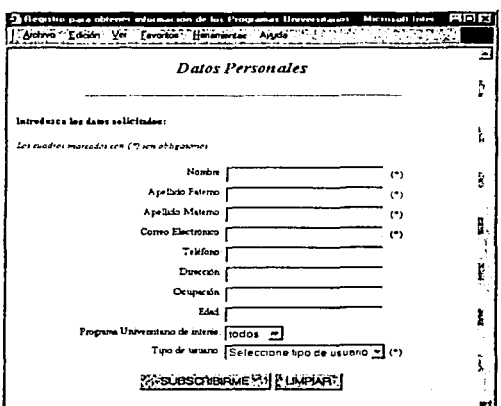


Figura 3. Formulario de registro de consultantes

A.4.3 Página principal de consultas

En esta página pueden seleccionarse las diferentes opciones de consulta que tiene disponibles el sistema (Figura 4). Adicionalmente, muestra ligas para

finalizar la sesión de consulta, regresar a la página inicial del sistema, o bien navegar por las páginas de cada uno de los Programas Universitarios.

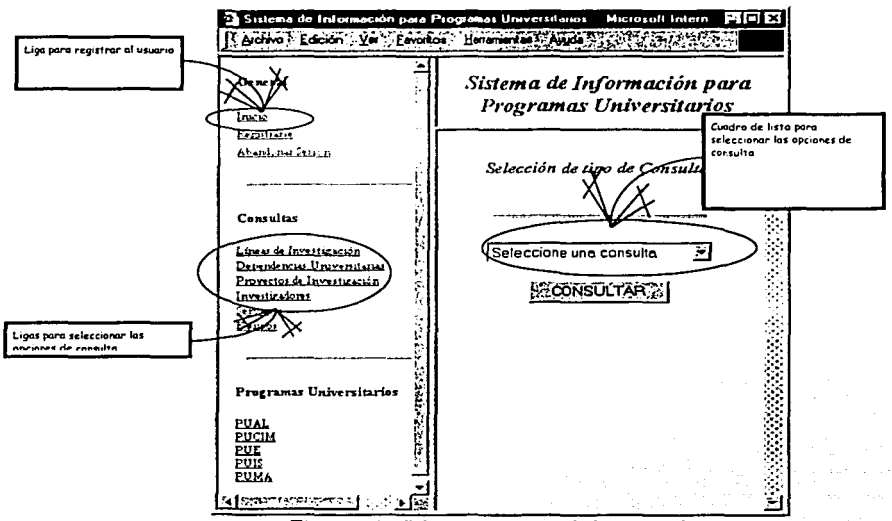


Figura 4. Página principal de consultas

A.4.4 Páginas de Consulta por Criterios de Filtrado

El acceso a cada una de estas páginas se realiza desde la página principal de consultas (Figura 4). La selección de cada página de consulta puede realizarse por medio del cuadro de lista, o bien, a través de las ligas de la porción izquierda.

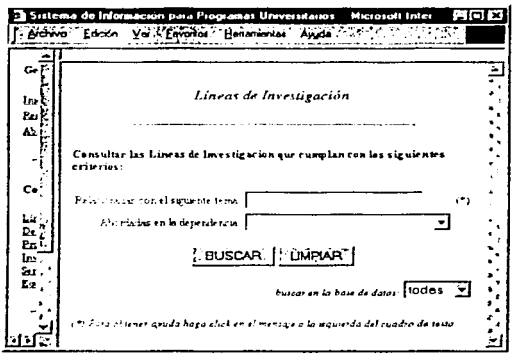


Figura 5. Página de consulta de Líneas de Investigación

Las páginas de consulta por criterios de filtrado disponibles son: Líneas de investigación, Dependencias Universitarias, Proyectos de Investigación, Investigadores, Servicios y Equipos (Figura 5 a 10)

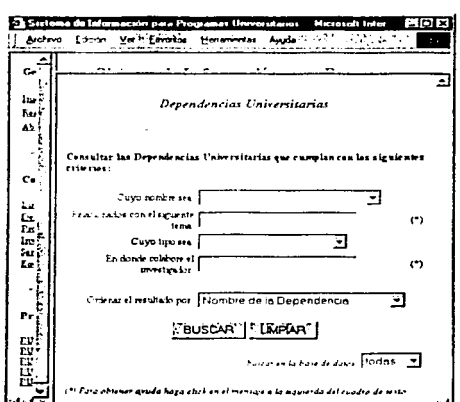


Figura 6. Consulta de Dependencias Universitarias

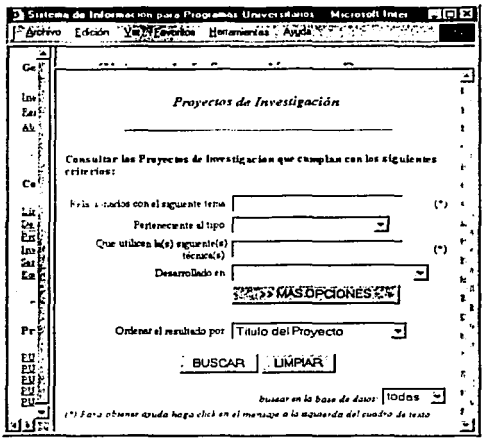


Figura 7. Consulta de Proyectos de Investigación

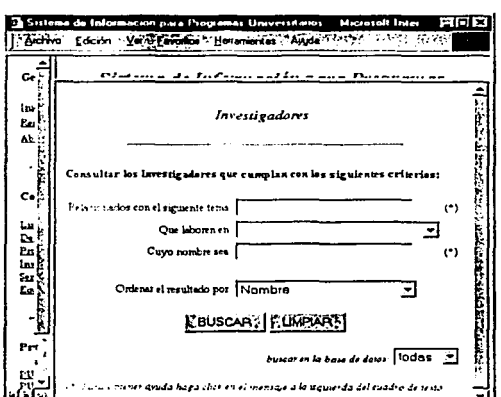


Figura 8. Página de consulta de Investigadores por criterios de filtrado

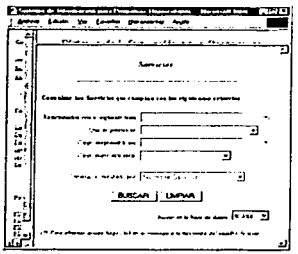


Figura 9. Página de consulta de Servicios

A.4.5 Criterios de filtrado adicionales

Las páginas de consulta de Proyectos de Investigación (Figura 7) y Equipos (Figura 10) pueden mostrar opciones adicionales (más particulares) de criterios de filtrado (Figura 11). Para mostrar estas opciones adicionales se debe pulsar el botón con la leyenda ">> MAS OPCIONES" (Figura 10)

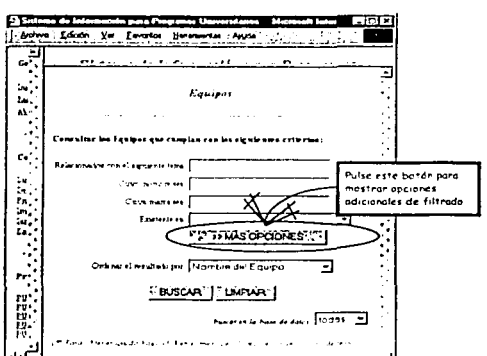


Figura 10. Página de consulta de Equipos

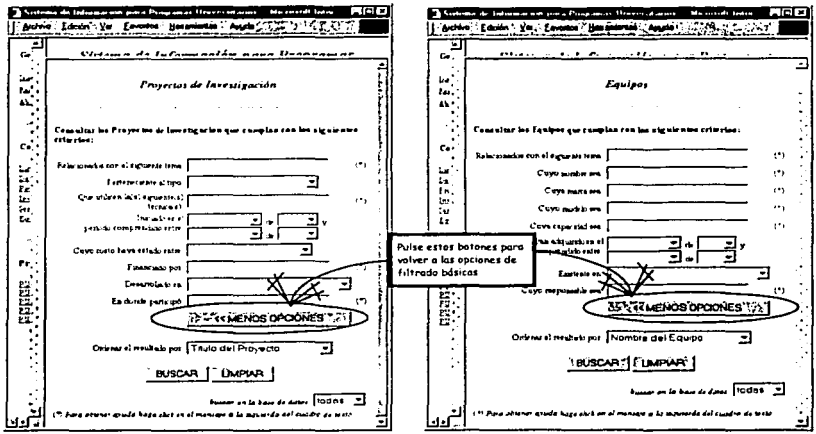


Figura 11. Páginas de consulta con opciones adicionales de filtrado (Proyectos de Investigación y Equipos)

A.4.6 Ejecución de las consultas

Las opciones de filtrado que se dejan vacías implican que una vez concluida la consulta, los resultados obtenidos no fueron filtrados por esos criterios. Por ejemplo, si se decide dejar en blanco el cuadro de texto "Cuyo nombre sea:" de la página de consulta de Investigadores (Figura 8), se estará indicando al sistema que se desean aquellos investigadores con CUALQUIER nombre.

Cuando se seleccionan varios criterios de filtrado, los resultados obtenidos obedecerán simultáneamente a todos los criterios seleccionados. Por ejemplo, si en la página de consulta de Investigadores (Figura 8) se escribe `Gómez` en el cuadro de texto "Cuyo nombre sea:", y se selecciona `Instituto de Ingeniería` en el cuadro de lista "Que laboren en:"; se obtendrán todos los investigadores que laboren en el Instituto de Ingeniería y que, además, alguno de sus apellidos (o ambos) sea Gómez.

Consecuentemente, si no se indica ningún criterio de filtrado, se obtendrán todas las entidades disponibles en la base de datos. Por ejemplo, si se dejan en blanco todas las opciones de filtrado en la página de consulta de Investigadores (Figura 8), se obtendrán todos los investigadores de la base de datos.

Adicionalmente, los resultados pueden ordenarse por diferentes características de las entidades buscadas (ver el cuadro de lista "Ordenar el resultado por:" en cada una de las páginas de consulta, Figura 5 a 11).

A.4.7 Consultas por Tema

Todas las páginas de consulta por criterios de filtrado (Figura 5 a 11) disponen de la opción "Relacionados(as) con el siguiente Tema:". En particular, cada uno de los temas que se indiquen en este cuadro de texto, se buscarán en los atributos que se enumeran a continuación de cada una de las entidades siguientes:

Investigadores

- Campos de interés o especialidades del investigador.
- Título o resumen de la línea de investigación de éste.
- Palabras clave o título de sus publicaciones.
- Palabras clave, título, resumen, objetivo, metodología o técnicas utilizadas por los proyectos que ha dirigido o en los que ha participado.
- Título, laboratorio o descripción de los servicios de los cuales es responsable.
- Nombre, descripción, función, modelo o marca de los equipos de los cuales es responsable.

Dependencias Universitarias

- Campos de interés, especialidades, líneas de investigación (título o resumen) o publicaciones (palabras clave o título) de los investigadores asociados con la dependencia universitaria.

- Palabras clave, título, resumen, objetivo, metodología, técnicas utilizadas o líneas de investigación (título o resumen) abordadas por los proyectos que se han desarrollado en esa dependencia universitaria.
- Título, laboratorio o descripción de los servicios que brinda.
- Nombre, descripción, función, modelo o marca de los equipos que posee.

Proyectos de investigación
- Palabras clave, título, resumen, objetivo, metodología, técnicas utilizadas o líneas de investigación (título o resumen) abordadas por el proyecto de investigación.

Líneas de Investigación
- Palabras clave, título, resumen, objetivo, metodología o técnicas utilizadas por los proyecto de investigación que abordan la línea de investigación.
- Título o resumen de la línea de investigación.

Servicios
- Título, laboratorio o descripción del servicio.
- Nombre, descripción, función, modelo o marca de los equipos empleados para brindar el servicio.

Equipos
- Título, laboratorio o descripción de los servicios que emplean este equipo.
- Nombre, descripción, función, modelo o marca del equipo.

A.4.8 Ayudas

No todos los cuadros de texto admiten el mismo tipo de texto. Para cada cuadro de texto existe una página de ayuda, la cual puede mostrarse si se hace click con el mouse sobre el texto a la izquierda de cada cuadro de texto (Figura 13).

Adicionalmente, existe una ventana de ayuda para el caso de los cuadros de lista relacionados con un período de tiempo (páginas de consulta de Equipos y Proyectos de Investigación, Figura 11).

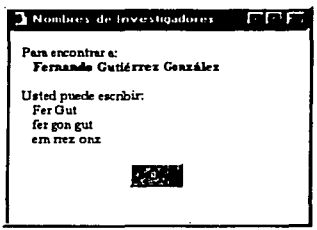


Figura 13. Ejemplo de una página de ayuda (Nombre de un Investigador)

A.4.9 Resultados de Consulta

Los resultados de cualquier consulta pueden mostrarse de dos formas básicas:

- Tabla con N entidades y Documento con UNA entidad.
- Tabla con N entidades

Cuando el resultado de una consulta muestra una o varias entidades, éstas se agrupan en forma tabular (Figura 14). En la parte superior de esta página se indica el tipo de entidades mostradas en la tabla, así como el número total de entidades que conforman el resultado. Si no se muestra el total de las entidades en la tabla, se ofrecen páginas adicionales (en forma de ligas) para mostrar el resto de las entidades.

Si este tipo de página proviene de una página de consulta por criterios de filtrado, se muestra un botón con la leyenda "<- Refinar Consulta" para dar la opción al usuario de modificar la consulta que produjo la tabla de resultados mostrada.

Cada una de las entidades que aparecen en la tabla se indican en forma de ligas. De esta forma, si se hace click con el mouse en alguna de ellas, se ampliará la información relacionada con la elegida.

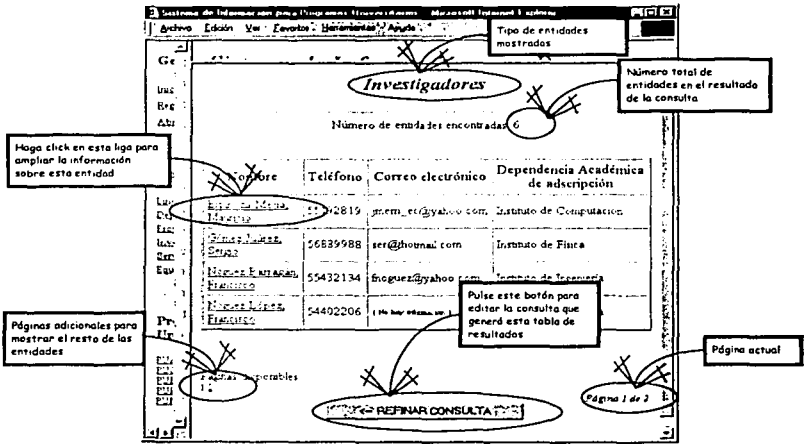


Figura 14. Ejemplo de una página de resultado de una consulta (tabla con N entidades)

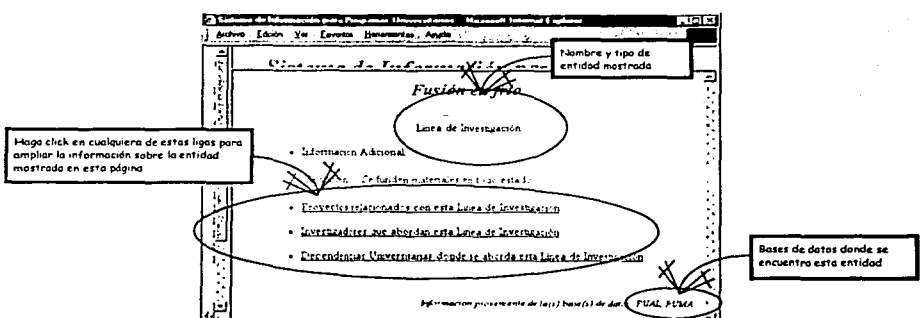


Figura 15. Ejemplo de una página de resultado de una consulta (Documento con UNA entidad)

En este tipo de página se encuentra TODA la información relacionada con la entidad indicada. Adicionalmente, la información que puede extenderse se ofrece como liga (Figura 15). Al final de la página se indica con qué base de datos esta ligada la entidad mostrada.

Notar que el sistema amplía sistemáticamente la información mostrada en sus páginas por medio de ligas, las cuales implican nuevas peticiones de consulta que atenderá el sistema.

A.4.10 Información no existente en la base de datos

En ocasiones no se encuentra la información solicitada, debido a que no está presente en la base de datos;. o bien, la consulta formulada no produce ninguna entidad que satisfaga todos los criterios de filtrado establecidos. En estos casos el sistema mostrará una página con un mensaje relativo a esta situación (Figura 16)

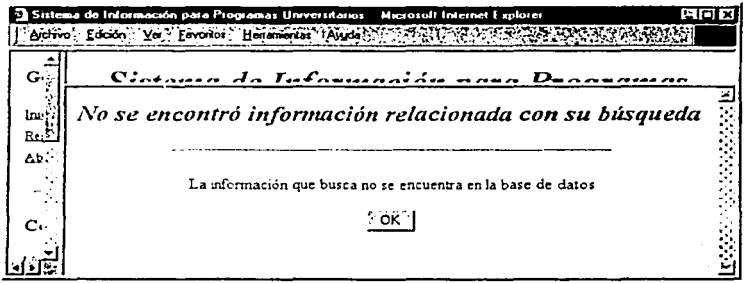


Figura 16. Página de resultado relacionada con una consulta que no produce ninguna entidad

Por otro lado, si una entidad no posee toda la información que podría contener, se mostrará en cada caso la leyenda "{No hay información}".

A.4.11 Salida del Sistema de Consultas

La configuración del sistema de consultas establece que la sesión de consulta de cualquier usuario expirará después de un período de inactividad de 20 minutos. Sin embargo, es posible provocar intencionalmente la finalización de la sesión de consulta, al hacer click en la liga "Abandonar Sesión" ubicada en la porción izquierda de la página principal de consultas (Figura 4)

B

Instalación y configuración de SIPU

B.1 Introducción

Para que la aplicación **SIPU©** funcione adecuadamente, es necesario instalar y configurar el software de soporte que se lista a continuación:

- Lenguaje Java
- Servidor de JSP's Tomcat
- Configuración de ODBC
- Instalación de los archivos JAR para ORACLE, POSTGRES e INTERBASE
- Instalación del archivo de la aplicación .WAR
- Configuración del archivo de propiedades settings.properties

Lo anterior debe ser realizado considerando el sistema operativo en el que el programa **SIPU©** será instalado. Por ello, se incluirán instrucciones para los sistemas operativos siguientes:

- Windows 98 Segunda edición
- Windows 2000
- Windows XP
- LINUX Red Hat 7.5

B.2 Lenguaje JAVA

El archivo de instalación del lenguaje JAVA puede ser obtenido del sitio de SUN (http://java.sun.com/) y se puede ser identificado a través del siguiente nombre: **j2sdk-1_4_1_01-windows-i586.exe** en sus versiones para Windows y **j2sdk-1_4_1_01-linux-i586.bin** en sus versiones para LINUX.

B.2.1 Instalación en Windows 98, Windows 2000 y Windows XP

En cualquier plataforma Windows, sólo es necesario ejecutar el archivo de instalación y el proceso se realizará automáticamente. Es recomendable que durante la instalación se pida que el lenguaje quede instalado en una ubicación sencilla de acceder o recordar. Para este ejemplo, se sugiere: C:\java14.

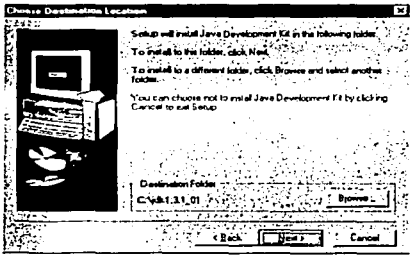


Figura B.1

Finalmente, modifique el **path** para que incluya el subdirectorio "bin" de Java, esto es modificar las variables de ambiente para cada sistema (Ya sea Windows98, Windows2000 o Windows XP), lo cual se explica a continuación.

Win9x (Windows 98)

Edite el archivo C:\autoexec.bat, actualizando el path y el CLASSAPTH de la siguiente manera:

path=%path%;"C:\java14\bin

después, reinicie el equipo.

Windows 2000, NT y XP

Haga clic con el botón derecho sobre el icono del escritorio llamado "Mi PC" y haga clic en el tab de propiedades. Posteriormente haga clic en el tab de "Opciones Avanzadas" y seleccione "variables de entorno". modifique las variables.

LINUX

Edite el archivo /etc/profile añadiendo al final de éste la línea:

set PATH=$PATH:/opt/java14/bin

Lo anterior suponiendo que java ha sido instalado en la ubicación /opt/java14

B.2.2 Instalación del servidor de JSP's

Posterior a la instalación del JSDK se requiere la instalación del servidor de JSP's. En este proyecto se empleó el servidor TomCat versión 4.0.

Tomcat es un Contenedor (entre mucho otros que existen en el mercado) de servlets que sirve como marco general de soporte a aplicaciones WEB. Es gratuito y es distribuido bajo la licencia GNU a través del sitio http://jakarta.apache.org. El archivo de instalación puede ser identificado a través de su nombre:

jakarta-tomcat-4.1.12.exe (para Windows)
jakarta-tomcat-4.1.12-src.tar.gz (para LINUX)

Nota: Si se desea instalar Tomcat, esto debe ser hecho posteriormente a la Instalación de Java. De hecho, en los sistemas Windows, el Instalador de Tomcat detecta su ubicación automáticamente. A continuación se presenta la Pantalla de instalación del servidor TomCat 4.0:

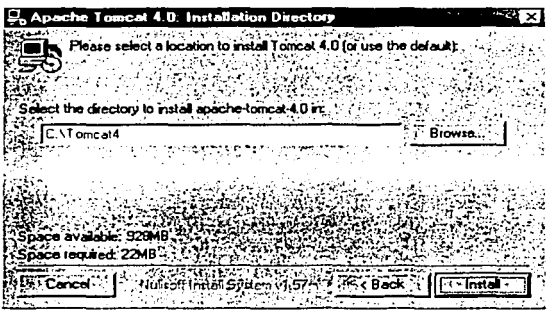


Figura B.2

En su versión para Windows, el instalador de Tomcat realiza todo el trabajo necesario, por lo que al finalizar la instalación ya no hay que realizar mas configuraciones para el ambiente de Tomcat. Sin embargo, para sistemas LINUX el procedimiento es el siguiente:

- Descompactar en cualquier ubicación (se sugiere una ruta corta y fácil de recordar) a través del siguiente comando: **tar xvzf jakarta-tomcat-4.1.12-src.tar.gz**

- Crear 2 variables de ambiente (JAVA_HOME & CATALINA_HOME) el archivo /etc/profile:
  - ✓ **export JAVA_HOME=\opt\java14**
  - ✓ **export CATALINA_HOME=\opt\tomcat 4**

- Ir a **\opt\tomcat4\bin** y dar permisos de ejecución a los **.sh** siguientes:
  - ✓ **startup.sh** y
  - ✓ **shutdown.sh**

  con el comando **chmod.** La sintaxis es: chmod 777 startup.sh y chmod 777 shutdown.sh

B.3 ODBC

Windows XP

1. Abra el Panel de Control de doble clic sobre Origen de datos ODBC.
2. De clic en Agregar en la ventana de Administrador de origen de datos ODBC.
3. Seleccione el tipo **driver** que se va a dar de alta en este caso su nombre es Microsoft Access Driver (*.mdb) y de clic en Finalizar.
4. Coloque un nombre a su driver, y de clic en Seleccionar para apuntar a la dirección de su base de datos.
5. Por último, de clic en Aceptar.

Windows 2000

1. Abra el Panel de Control de doble clic sobre ODBC Data Sources (32bit).
2. De clic en Add en la ventana de ODBC Data Source Administrator.
3. Seleccione el tipo **driver** que se va a dar de alta en este caso su nombre es MS Access DataBase y de clic en Add.
4. Coloque un nombre a su driver, y de clic en Seleccionar para apuntar a la dirección de su base de datos.
5. Por último, de clic en Aceptar.

Windows 98

1. Abra el Panel de Control de doble clic sobre Origen de datos ODBC.
2. De clic en Agregar en la ventana de Administrador de orígenes de datos ODBC.

3. Seleccione el tipo **driver** que se va a dar de alta en este caso su nombre es Microsoft Access Driver (*.mdb) y de clic en Finalizar.
4. Coloque un nombre a su driver, y de clic en Seleccionar para apuntar a la dirección de su base de datos.

Por último, de clic en Aceptar y la configuración de ODBC se habrá completado.

B.4 Instalación del archivo de instalación WAR

El archivo SUPI.WAR es el archivo que contiene a la aplicación y deberá ser colocado en el subdirectorio CATALINA_HOME/webapps. A continuación, el servidor de JSP's deberá de ser reiniciado. Y la aplicación estará disponible en la dirección http://localhost:8080/SIPU.

Cabe mencionar que en este documento se está mencionando la variable de ambiente CATALINA_HOME representando la ubicación del servidor de JSP's Tomcat, que deberá de haber sido instalado previamente.

B.5 Instalación de los archivos JAR para ORACLE, POSTGRES e INTERBASE.

La aplicación **SIPU©** fue desarrollada para poder soportar los manejadores de bases de datos ORACLE 8i, POSTGRES 7.0, Microsoft Access 2000 e Internase 6.5. Para que en su momento, éstos manejadores puedan ser accedidos, será necesario colocar en el subdirectorio CATALINA_HOME/common/lib los siguientes archivos:

- Oracle.jar
- Internase.jar
- Jdbc7-1-0.1.jar

B.6 Configuración del archivo settings.properties.

Finalmente, el archivo de propiedades "setting.properties" deberá ser configurado de la siguiente manera:

```
01 DBMS=ORACLE
02 URL=jdbc:oracle:thin:@uxmcc1.iimas.unam.mx:1521:mcic
03 DRV=oracle.jdbc.driver.OracleDriver
04 USR=arellano
05 PSW=abc
06 nConn=8
07 getPoolCnnTimeOut=7000
08 logPath=log.wri
09 tracePoolSize=true
10 TiempoEntreRegistrosMismoUsuario=1800
11
12 logPath=C:\\programas\\Tomcat4\\webapps\\IT\\
13 logFilePreName=log.wri
```

- En la línea 01 se deberá substituir la palabra ORACLE por ACCESS o INTERBASE o POSTGRES según sea el caso basados en el manejador que se desea utilizar.
- En la línea 02 se deberá colocar la dirección de la base de datos. La estructura de esta dirección cambia, en función del manejador que se utilice. Para mayor referencia, consultar la documentación de driver JDBC a utilizar. Como ejemplo, para Access ésta debería ser: jdbc:odbc:sipu, tomando en cuenta que existe una entrada ODBC llamada sipu y apuntando a una base de datos existente con las estructuras y datos correspondientes.
- En la línea 03 deberá ir la descripción del driver utilizado:

Para ACCESS es:	sun.jdbc.odbc.JdbcOdbcDriver
Para ORACLE es:	oracle.jdbc.driver.OracleDriver
Para INTERBASE es:	internase.borland.jdbc.Driver
Para POSTGRES es:	org.postgresql.Driver

- En las líneas 04 y 05 se deberá sustituir 'arellano' y 'abc' por el usuario y el clave de una cuenta existente en el manejador de bases de datos.

- Finalmente, en la línea 12 para el rubro logPath, bastará incluir una trayectoria existente en el server y con derechos de escritura para el manejador de JSP's. Para los sistemas Windows, cada subdirectorio deberá ir separado por diagonales dobles (\\). Para sistemas UNIX-like cada subdirectorio deberá ir separado por diagonales invertidas simples (/). Por ejemplo: /tmp/sipuLog/

Nota: Los números de línea son auxiliares para esta explicación. No existen en el archivo original y no deberán ser incorporados en ningún momento.

En este punto, conviene mencionar cuál es la función de los demás parámetros del archivo de configuración. Esta configuración no es necesaria pero puede ser de utilidad en caso de que el sistema requiera de la utilización de mas recursos de los que se determinaron inicialmente:

- En la línea 06 se establece el número de conexiones que será establecido al momento de arrancar el servidor de JSP's.
- En la línea 07 se establece el número máximo de milisegundos que el programa esperará antes de informar que existió un time-out, por falta de conexiones a la base de datos.
- En la línea 08 se define el nombre del archivo que contendrá la lista de transacciones, ingresos al sistema y errores ocurridos durante su operación.
- La línea 09 permite desplegar los mensajes del log de transacciones en la ventana de terminal de Tomcat si está en 'true'.
- La línea 10 provoca que un usuario no pueda registrarse en el sistema a menos que haya transcurrido cierto tiempo medido en segundos a partir de su último registro.

Con esto último concluye el apéndice B dedicado a la instalación y puesta en marcha del sistema de información para programas universitarios **SIPU©**.

C

Convenciones de codificación en Java

1. Nombres de archivos.

Los nombres de archivos tendrán la primer letra con mayúscula en caso de ser nombres compuestos las primeras letras de cada palabra deberán estar en mayúsculas. No se deberá dejar ni espacios en blanco ni guiones entre palabras.

Ejemplo.

PagoNomina.java
PagoNomina.class

2. Organización de archivos.

Las secciones deberán estar separadas por renglones en blanco y en caso necesario deberán llevar un comentario descriptivo, los archivos con mas de 2000 líneas de código se deberán evitar para reducir la complejidad de su lectura y revisión en caso necesario.

El archivo fuente Java deberá tener el siguiente orden:

- Comentarios al inicio del archivo
- Paquetes y declaraciones de importación
- Clase y declaración de interfaz

2.1 Comentarios al inicio del archivo.

Todo archivo fuente empezará con un comentario con el estilo que se muestra en el siguiente ejemplo, con el nombre de la clase, versión de la información, fecha y el aviso de los derechos de propiedad:

```
/*
 * Nombre de la clase
 *
 * Versión de la información
 *
 * Fecha
 *
 * El aviso de derechos de propiedad
 */
```

2.2 Paquetes y declaraciones de importación

La primera línea no comentada del código fuente Java debe ser una declaración de paquetes. Después de eso, se deberán declarar las importaciones que siguen.

Por ejemplo:

```
package java.awt.;

import java.awt.peei.CanvasPeer;
```

2.3 Clase y declaración de interfaz.

La siguiente tabla describe las partes de una clase o declaración de interfaz, en el orden que ellos deban aparecer.

	Parte de la Clases/Interfaz	Notas
1	Comentario de documentación de Clase/Interfase (/**..*/)	

2	Sentencia class o interface	Este comentario debería contener cualquier información sobre la clase o la interfase que no fuera apropiada para ponerla en el comentario de documentación
3	Comentario de implementación de Clase/Interfase (/**/), si es necesario	
4	Variables de clase (static)	
5	Variables de instancia	
6	Constructores	
7	Métodos	

Tabla 2.1

3. Identación.

Se emplearán identaciones de cuatro espacios como unidad de identación. Las tabulaciones deberán ponerse exactamente cada ocho espacios. Por lo anterior una tabulación emplea dos unidades de identación.

3.1 Longitud de línea.

Se evitarán líneas de más de 80 caracteres de longitud, en el caso de las líneas de comentarios se deberán usar líneas de no más de 70 caracteres de longitud.

3.2 Expresiones en más de una línea.

En caso de que una expresión no quepa en una sola línea se deberán seguir estos principios generales.

- Separar después de una coma.
- Separar antes de un operador.
- Preferir un nivel más alto de separación a un nivel más bajo de separación.
- Alinear la nueva línea con el comienzo de la expresión al mismo nivel en el nivel previo.

A continuación unos ejemplos de cómo distribuir una expresión en más de una línea:

```
someMethod(longExpression1, longEspression2, longExpression3,
                        longExpression4, longExpression5);

      var = someMethod1(longExpression1,
                                    someMethod2(longExpression2,

longExpression3));
```

Al distribuir una expresión aritmética en más de una línea se deberá preferir la primera de las siguientes opciones:

```
longName1 = longName2 * (longName3 + longName4 - longName5)
                      + 4 * longname6; //PREFERIR ESTA

longName1 = longName2 * (longName3 + longName4 -
                        longName5) + 4 * longname6; //EVITAR
```

Otro ejemplo de identación para evitar hacer el cuerpo difícil de leer o identificar es el siguiente:

```
//EVITAR ESTA IDENTACION
if ((condition1 && condition2)
     || (condition3 && condition4)
     ||!(condition5 && condition6)) { //MALA ENVOLTURA
     doSomethingAboutIt();     //LINEA FACIL DE PERDERSE
}


//USAR ESTA IDENTACION EN LUGAR DE LA ANTERIOR
if ((condition1 && condition2)
            || (condition3 && condition4)
            ||!(condition5 && condition6)) {
     doSomethingAboutIt();
}


//O USAR ESTA
if ((condition1 && condition2) || (condition3 && cond4)
            ||!(condition5 && condition6)) {
     doSomethingAboutIt();
}
```

4. Comentarios.

Se considerarán los siguientes tipos de comentarios, los de implementación (delimitados por /*...*/ y //) y comentarios de documentación (delimitados por /**...*/).

Se deberán usar los comentarios para dar apreciaciones globales de código y proporcionar información adicional del propio código. Los comentarios deberán contener sólo información que es pertinente para leer y entender el programa. Por ejemplo, información sobre cómo el paquete correspondiente es construido o en que directorio que reside no deberá ser incluido como un comentario.

Los comentarios no deberán incluirse en cajas grandes dibujadas con asteriscos u otros caracteres.

Los comentarios no deberán incluir caracteres especiales como un retroceso.

4.1 Comentarios de implementación.

4.1.1 Comentarios de bloques.

Los comentarios de bloque pueden ser usados al principio de cada archivo y antes de cada método. Ellos también pueden usarse en otros lugares, como dentro de los métodos.

Un comentario de bloque deberá ser precedido por un salto de línea para ponerlo aparte del resto del código.

```
/*
*   Aquí es el comentario de bloque
*/
```

Los comentarios de bloque pueden empezar con /*- que se reconoce por identado(1) como el principio de un comentario de bloque que no deberá ser reformateado.

Ejemplo:

```
/*-
* Aquí es el comentario de bloque con algunos formatos
* muy especiales que yo quiero para que sean ignorados.
*       uno
*                dos
```

```
*                          tres
*/
```

4.1.2 Una sola línea de comentarios.

Los comentarios cortos podrán aparecer en una sola línea con identación al nivel del código que sigue. Si un comentario no pudiera ser escrito en una sola línea, se deberá seguir el formato de comentario de bloque. Una sola línea de comentario deberá ser precedido por una línea en blanco.

Ejemplo de una línea simple de comentarios en el código Java:

```
if (condition) {

        /* Manejo de la condición. */
        . . .
}
```

4.1.3 Comentarios en misma línea de código.

Los comentarios muy cortos pueden aparecer en la misma línea que el código al que describe, pero deberán ser colocados lo bastante lejos para separarlos de las declaraciones. Si más de un comentario corto aparece en un bloque de código, todos ellos deberán estar con identación en la misma columna.

Ejemplo de comentarios en la misma línea en el código Java:

```
if (a = = 2) {
     return TRUE;              /* caso especial */
} else {
     return isPrime (a);       /* solo trabaja para un impar */
}
```

4.1.4 Comentarios de fin de línea.

El delimitador de comentarios // puede comentar fuera de una línea completa o solo una línea parcial. No deberá ser usado en múltiples líneas consecutivas para comentarios de texto; sin embargo, podrá usarse en múltiples líneas consecutivas para comentar fuera de secciones del código.

Ejemplos de todos los tres estilos siguientes:

```
if (foo > 1) {
```

```
                    // Comentario fuera de una linea completa
                    ...
                }
                 else{
                    return false;       // Comentario de línea
parcial
                }


                // if (bar > 1) {
                //
                //        // Para comentarios fuera de secciones de
código
                //          ...
                //}
                //else{
                //        return false;
                //}
```

4.2 Comentarios de documentación.

Cada comentario doc es un conjunto fijo dentro de los delimitadores de comentarios /** . . . */, con un comentario por clase, interfaz o miembro. Este comentario justamente deberá aparecer antes de la declaración:

```
/**
 * La clase del ejemplo proporciona . . .
 */
public class Example { . . .
```

La primer línea del comentario doc (/**) para las clases e interfaces no será identada; las líneas del comentario doc subsiguiente tienen cada una 1 espacio de identación (para alinear verticalmente los asteriscos).

Si se necesita dar información sobre una clase, interfaz, variable o método que no son apropiados para la documentación, se usara una implementación de comentario de bloque o una sola línea de comentarios inmediatamente *después* de la declaración.

Los comentarios doc no deberán posicionarse dentro de un método o bloque de definición de constructor, porque Java asocia los comentarios de documentación con la primera declaración *después* del comentario.

5. Declaraciones.

5.1 Número de declaraciones por línea.

Si se piensa comentar a las declaraciones se recomienda una declaración por línea, por ejemplo:

```
int level;  // nivel de identación
int size;   // tamaño de la tabla
```

es preferible sobre

```
int level, size;
```

Se deberá evitar la declaración de diferentes tipos en la misma línea. Ejemplo:

```
int foo, fooarray[]; //evitar!
```

5.2 Inicialización.

En cuanto a la inicialización, se deberá procurar inicializar variables locales donde se declaran. La única razón para no inicializar un variable dónde se declara es si el valor inicial depende de algún proceso que ocurre primero.

5.3 Colocación.

Se procurará poner las declaraciones al principio de los bloques, se deberá evitar declarar las variables hasta su primer uso ya que puede causar confusiones y obstaculizar la portabilidad del código.

Ejemplo :

```
void myMethod ( ) {
      int  int1 = 0;   // empezando el bloque del método
      if  (condition) {
            int int2 = 0; // empezando el bloque de "si"
            . . .
      }
}
```

La única excepción a la regla anterior se da en los índices para los ciclos, los cuales en Java pueden declararse como sigue :

```
for (int i = 0; i < maxLoops; i++) { . . . }
```

Se evitarán las declaraciones locales que se confundan con las declaraciones de los niveles superiores. Por ejemplo, no se deberá declarar con el mismo nombre a una variable en un bloque interno:

```
int count;
. . .
myMethod ( ) {
          if (condition) {
               int count;      //evitar!
               . . .
          }
           . . .
}
```

5.4 Declaración de clases e interfaces.

Cuando se codifiquen clases Java en interfaces, se deberán seguir las siguientes reglas estructuradas:

- No deberá haber espacios entre un nombre de método y los paréntesis "(" de su lista de parámetros.
- La llave de apertura "{" aparecerá al final de la misma línea de la declaración
- La llave de cierre "}" deberá aparecer al principio de una línea identada solo para emparejar su correspondiente apertura de declaración, excepto cuando es una declaración nula, la llave "}" deberá aparecer inmediatamente después de la llave "}".

Ejemplo:

```
class Sample extendí Object {
      int ivar1;
      int ivar2;
      Sample (int i, intj) {
               ivar1 = 1;
               ivar2 = j;
      }
```

```
        int emptyMethod ( ) ()
        . . .
)
```

- Los métodos deberán ser separados por una línea en blanco.

5.5 Declaraciones simples.

Cada línea deberá contener una declaración a lo más, por ejemplo:

```
argv++;                 // Correcto
argc++;                 // Correcto
argv++;  argc--;        // Incorrecto!
```

5.6 Declaraciones compuestas.

Las declaraciones compuestas son declaraciones que contienen listas de declaraciones incluidas en llaves "{ declaraciones }". En las siguientes secciones se muestran los ejemplos.

- Las declaraciones adjuntas deberán identarse en más de un nivel que la declaración compuesta.
- La llave de apertura deberá estar al final de la línea en que empieza la declaración compuesta; la llave de cerradura deberá empezar en una línea y estar identada al principio de la declaración compuesta.
- Las llaves son usadas alrededor de todas las declaraciones, incluso declaraciones únicas, cuando son la parte de una estructura de control, tal como la declaración if-else o for. Esto facilita agregar declaraciones sin producir bugs accidentalmente debido al olvido de agregar las llaves.

5.7 Retorno de declaraciones.

Una declaración de retorno (return) con un valor no deberá usar los paréntesis a menos que ellos hagan el valor de regreso más obvio de alguna manera.

Ejemplo:

```
return;

return myDisk.size( );

return (size ? size : defaultSize);
```

5.8 Declaraciones if, if - else, if else - if else.

Las declaraciones if – else, deberán tener la siguiente forma:

```
if (condition) {
     statements;
}

if (condition) {


     statements;
} else {

     statements;
}

if (condition) {
     statements;
} else if (condition) {
     statements;
} else {
     statements;
}
```

La declaración if siempre usa llaves {}. Se deberá evitar la siguiente forma que esta propensa a errores:

```
if (condition)   // evitar que se omitan las llaves , {}!
    statement;
```

5.9 Declaraciones for.

La declaración for deberá tener la siguiente forma:

```
for (initialization; condition; update) {
        statements;
}
```

Una declaración for vacía (en la cual todo el trabajo es hecho en la inicialización, condición, y cláusulas de actualización) deberá tener la siguiente forma:

```
for (initialization; condition; update);
```

Cuando se use el operador punto y coma ";" en la inicialización o cláusula de actualización de una declaración for, se evitará la complejidad de usar más de tres variables. Si es necesario, se emplearán las declaraciones separadas antes del loop for (para la cláusula de inicialización) o al final del loop (para la cláusula de actualización).

5.10 Declaraciones while.

La declaración while deberá tener la siguiente forma:

```
while (condition) {
        statement;
}
```

Una declaración while vacía deberá tener la siguiente forma:

```
while (condition);
```

5.11 Declaraciones do – while.

La declaración while deberá tener la siguiente forma:

```
do {
        statement;
} while (condition);
```

5.12 Declaraciones switch.

Una declaración switch deberá tener la siguiente forma:

```
switch ( condicion) {
    case ABC:
          statements;
          /* falls through */
    case DEF:
          statements;
          break;
    case XYZ:
          statements;
          break;
    default:
          statements;
          break;
}
```

Cada vez que un caso falla (no se deberá incluir una declaración break), se deberá agregar un comentario donde la declaración break normalmente estaría. Esto se muestra en el ejemplo anterior con el comentario /* falls through */.

Cada declaración switch deberá incluir un caso default. El break en el caso default es redundante, pero previene una caída a través del error si después otro se agrega.

5.13 Declaraciones try-catch.

Una declaración try-catch deberá tener el formato siguiente:

```
try {
      statements;
} catch (ExceptionClass e) {
      statements;
}
```

Una declaración try-catch también puede seguirse por finally, qué se ejecuta sin tener en cuenta si el bloque de try ha completado con éxito o no.

```
try {
      statements;
} catch (ExceptionClass e) {
      statements;
```

```
} finally {
      statements;
}
```

6. Áreas en blanco.

6.1 Líneas en blanco.

Siempre deberán usarse dos líneas en blanco en las siguientes circunstancias:

- Entre las secciones de un archivo fuente
- Entre la definición de clase e interfaces

Siempre deberá usarse una línea en blanco en las siguientes circunstancias:

- Entre los métodos
- Entre las variables locales en un método y su primera declaración
- Antes de un bloque de comentarios o una sola línea de comentario
- Entre las secciones lógicas dentro de un método para mejorar la legibilidad

6.2 Espacios en blanco.

Los espacios en blanco deberán usarse en las siguientes circunstancias:

- Una palabra clave seguida por un paréntesis deberá separarse por un espacio.

Ejemplo:

```
while (true) {
. . .
}
```

El espacio en blanco no deberá usarse entre el nombre de un método y su paréntesis de apertura. Esto ayudará a distinguir las palabras clave de las llamadas de los métodos.

- Un espacio en blanco deberá aparecer antes de las comas de la lista de argumentos

- Todo operador binario excepto "." (punto) deberá separarse de sus operandos por espacios. El espacio en blanco nunca deberá separarse de su operador unario como el unario menos, incremento ("++"), y decremento ("—") de sus operandos.

Ejemplo:

```
a  +=  c + d;
a  =  (a + b) / (c * d);

while (d++ = s++) {
       n++;
}
prints("size is " + foo +"\n");
```

- Las expresiones en una declaración for deberán separase por espacios en blanco.

Ejemplo:

```
for (expr1; expr2; expr3)
```

- Los cast deberán ser seguidos por un espacio en blanco.

Ejemplo:

```
myMethod ((byte) aNum, (Object) x);
myMethod ((int) (cp + 5), ((int) (i + 3))+ 1);
```

7. Convenciones de nombramiento.

Las convenciones de nombramiento hacen programas más entendibles haciéndoles más fácil de leer. Ellos también pueden dar información sobre la función del identificador – por ejemplo, si es una constante, paquete o clase – lo cual puede ser útil en el entendimiento del código.

Tipos de Identificador	Ejemplos	Reglas de nombramiento
Paquetes	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese	El prefijo de un único nombre de paquete es siempre escrito en todo con letras minúsculas ASCII y deberá ser uno de

		los nombres de dominio de nivel-superior, concurrentemente com, edu, gov, mil, net,org, o una de las dos letras del Ingles de los códigos de identificación de pa ses como se especifica en la norma estÆndar ISO 3166, 1981.
Clases	class Raster; class ImageSprite;	Los nombres de clases deberán ser sustantivos, en el caso mixto con la primera letra de cada palabra interior en mayúsculas. Intente guardar su clase con un nombre simple y descriptivo. Se usarán palabras enteras – se deberá evitar acrónimos y abreviaciones (a menos que la abreviación sea mucho más ampliamente usada que la forma larga, tal como URL o HTML).
Interfaces	interface RasterDelegate interface Storing;	Los nombres de las interfaces deberán ser escritos como los nombres de las clases. La primera letra de cada palabra en caso de ser una palabra mixta, deberá estar en mayúsculas.
Métodos	run ( ); runFast ( ); getBackground ( );	Los métodos deberán ser verbos, en el caso mixto con la primer letra minúscula, con la primer letra de cada palabra interna en mayúsculas como el método runFast()

Tabla 7.1

Convenciones de nombramiento de variables.

Tipos de identificador	Ejemplos	Reglas de Nombramiento
Variables	int i; char c; float myWidth;	Excepto por las variables, toda instancia, clase, y clase constante caen en el caso mixto con la primera letra minúscula y las palabras internas estarán en letras mayúsculas. Los nombres de las variables no deberán empezar con los caracteres de subrayado _ o el signo de dólar $,

		incluso aunque ambos se permiten. Los nombres de las variables deberán ser cortos, pero a la vez significativos. La opción de un nombre de variable deberá ser en código mnemotécnico – esto es, diseñado para indicar al observador casual el intento de su uso. Un carácter como nombre de variable deberá ser evitable excepto por temporalidad "desechable" de variables. Los nombres comunes para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres.

Constantes	static final int MIN_WIDTH = 4;
	static final int MAX_WIDTH = 999;
	static final int GET_THE_CPU = 1;
		L os nombres de variables declaradas, clases constantes y constantes de ANSI deberán ser todas mayúsculas con palabras separadas por el carácter de subrayado ("_"). (Las constantes ANSI deberán ser evitadas, para el fácil debugueo.)

Tabla 7.2

8. Practicas de programación.

8.1 Acceso proporcionado a variables de instancia y variables de clases.

No se deberá instanciar o declarar una variable de clase pública sin una buena razón. A menudo, las variables de instancia no necesitan ser puestas explícitamente o enviadas - a menudo pasa como un efecto secundario de llamadas de métodos.

Un ejemplo de una apropiada variable de instancia pública es el caso donde la clase es esencialmente una estructura de datos, es decir, si usted hubiera usado un *struct* en lugar de una clase (si Java soportara el *struct*), entonces es apropiado utilizar las clases en lugar de variables públicas.

8.2 Referencias a variables de clases y métodos.

Se evitará usar un objeto para acceder a una variable clase (estática) o método. Se usará un nombre de la clase en cambio. Por ejemplo:

```
classMethod ( );          //Correcto
AClass.classMethod ( );   //Correcto
anObject.classMethod ( ); //Incorrecto
```

8.3 Constantes.

Las constantes numéricas no deberán codificarse directamente, excepto por -1, 0 , y 1, los cuales podrán aparecer en un loop for como valores de conteo.

8.4 Variables de asignación.

Se deberá evitar asignar varias variables al mismo valor en una sola declaración. Esto es difícil de leer. Ejemplo:

```
fooBar.fChar  =  barFoo.lchar  =  'c';   // Incorrecto
```

No se deberá usar el operador de asignación en un lugar dónde pueda confundirse fácilmente con el operador de igualdad. Ejemplo:

```
if (c++ = d++) {   // Incorrecto  (Java no lo permite)
    . . .
}
```

deberá escribirse como

```
if ((c++ = d++) != 0) {
    . . .
}
```

No se deberá usar las asignaciones de inclusión en un esfuerzo por mejorar el tiempo de ejecución. Este es el trabajo del compilador. Ejemplo:

```
d = (a = b + c) + r;     // Incorrecto
```

deberá escribirse como:

```
a = b + c;
d = a + r;
```

8.5 Prácticas misceláneas.

8.5.1 Paréntesis.

Generalmente es una buena idea usar los paréntesis en expresiones que involúcran operadores mixtos para evitar problemas con operadores de precedencia. Aun si el operador de precedencia parece claro, podría no ser claro para otros programadores, no se deberá asumir que es clara la interpretación de precedencia.

```
if  ( a = = b  && c = = d)   // No asumir que esto es claro
if  ((a = = b) && (c = = d)) // preferir este código
```

8.5.2 Valores devueltos.

Cuando se intente hacer la estructura de un programa encontrando los resultados.
Ejemplo:

```
if ( booleanExpression) {
      return true;
} else {
      return false;
}
```

deberá escribirse en cambio como:

```
return booleanExpression;
```

Similarmente,

```
if (condition) {
    return x;
}
return y;
```


deberá escribirse como

```
return (condition ? x : y);
```

8.5.3 Expresiones antes de "?"en el operador condicional.

Si una expresión que contiene a un operador binario aparece antes el '?' en el operador ternario ':', deberán usarse paréntesis. Ejemplo:

```
(x >= 0) ? x : -x;
```

Bibliografía

[1] Introduction to the Team Software Process. Humphrey, Watts. Editorial Addison Wesley, 1999.

[2] Rational Unified Process, Rational Software.

[3] eXtreme programming. Beck, Kent. Editorial Addison Wesley, 2000.

[4] El lenguaje unificado de modelado. Booch, Grady. Editorial Addison Wesley, 2000.

[5] ISO/IEC 9126-1

[6] Sitio Web de apoyo a TSP. www.kasia.fciencias.unam.mx/TSPi

[7] UML Distilled. Fowler, Martin. Editorial Pearson, 1999.

[8] Carpeta de desarrollo proyecto SIPU.

[9] Software Engineering. Shari Lawrence Pfleeger. Editorial Prentice Hall, 2001

[10] UML y patrones. Larman, Craig. Editorial Prentice Hall, 1999.

[11] Introduction to the Personal Software Process. Humphrey, Watts. Editorial Addison Wesley, 1997.

[12] Professional Java Server Programming. Various authors. Editorial Wrox, serie Programmer to programmer. 2000.

[13] Professional JSP. Various authors. Editorial Wrox, serie Programmer to programmer. 2001.

[14] El proceso de desarrollo de software.

[15] William S. Davis "Tools and techniques for structured systems analysis and design" [15].