UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Apoyos didácticos para el curso de Ingeniería de Software

Reporte de actividad docente

Que para obtener el título de:

Licenciado en ciencias de la computación

P r e s e n t a :

Miguel Ehécatl Morales Trujillo

Tutora:
M. en C. María Guadalupe Elena Ibargüengoitia González

2007

1. Datos del alumno

Morales
Trujillo
Miguel Ehécatl
56 32 31 58
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
300109133

2. Datos del tutor

M en C
María Guadalupe Elena
Ibargüengoitia
González

3. Datos del sinodal 1

Dra
Hanna
Oktaba

4. Datos del sinodal 2

Mat
Sylviane
Levy
Amselle

5. Datos del sinodal 3

Dra
Amparo
López
Gaona

6. Datos del sinodal 4

L en C C
René Alejandro
Villeda
Ruz

7. Datos del trabajo escrito

Apoyos didácticos para el curso de Ingeniería de Software
115 p
2007

Agradecimientos

Agradezco sinceramente a la M. en C. Guadalupe Ibargüengoitia González por todo el apoyo y tiempo que me dedicó para la realización de este trabajo, siempre dispuesta a atender mis inquietudes y resolver mis dudas.

Al jurado integrado por las Dras. Hanna Oktaba y Amparo López Gaona, la Mat. Sylviane Levy Amselle y al L.C.C René Alejandro Villeda Ruz por sus valiosos comentarios y sugerencias que no hicieron más que mejorar este trabajo.

Dedicatoria

A mis padres.
A mis hermanas.
A mi esposa y a mi hijo.

Introducción.

El objetivo de este trabajo es proveer de ejemplos al libro de texto y la página del curso de “Ingeniería del Software” que sirven de apoyo a los alumnos que cursan esta materia en la Facultad de Ciencias.

Aprovechando la gran cantidad de documentación generada por alumnos que cursaron las materias de “Ingeniería de Software” de la Licenciatura en Ciencias de la Computación y “Tecnología Orientada a Objetos” del Posgrado en Ciencia e Ingeniería de la Computación se inició la búsqueda y selección de documentos que conforman el proceso de desarrollo de software.

El primer paso fue revisar la documentación de cada proceso, es decir, lo hecho por cada equipo de trabajo durante el curso, localizando aquellos documentos que cumplían cabalmente con lo estipulado para el curso.

Posteriormente, se seleccionaron los documentos que resultaban más didácticos, es decir, aquéllos que ejemplificaban la estructura principal de cada documento de manera clara y sencilla, que podían responderle al alumno la mayoría de sus inquietudes al comenzar a generar un documento. Ya con los ejemplos seleccionados se procedió a la verificación y corrección de dichos documentos, resultando en un compendio con los productos generados propuestos por el libro de texto del curso que auxiliarán al alumno a lo largo de cada fase y durante cada ciclo de su proceso.

Finalmente se han seleccionado artículos de divulgación en revistas científicas y especializadas en computación, principalmente en Ingeniería de Software, como lo son IEEE Computer e IEEE Software, Communication ACM y Software Guru, principalmente, tanto en sus ediciones impresas como digitales. Todo esto con el fin de que el alumno que cursa Ingeniería de Software tenga un panorama global de los alcances y aplicaciones de la industria del software más allá de las aulas y en empresas competitivas del mundo real.

En la primera parte de este trabajo se dará un panorama actual de la Ingeniería de Software, tanto en el salón de clases como en la vida real, con ello se busca que el alumno identifique de una manera teórica y práctica el vasto campo de esta disciplina que es relativamente joven.

En los capítulos posteriores, del segundo al noveno, se describirán cada una de las fases del proceso de desarrollo de software, así como cada uno de los productos generados a través de éstas. Cada definición de los cuarenta y dos productos propuestos a lo largo del desarrollo busca ser didáctica para que el alumno genere por sí mismo un producto de calidad sin que repita los defectos que se encontraron en productos hechos en ciclos anteriores. Inmediatamente después de las descripciones de los productos se incluye un ejemplo de cada producto que observa las características descritas en el libro de texto de Ingeniería de Software.

El trabajo realizado en este escrito se incorporará al sitio web del curso de Ingeniería de Software, particularmente en la sección de “Apoyos al libro de Ingeniería de Software Pragmática”. Dicha sección consta de enlaces hacia cada uno de los capítulos del libro. Para acceder a ésta se debe presionar “Apoyos al libro de Ingeniería de Software Pragmática” del menú principal, con lo que se mostrará la página que contiene la totalidad de los capítulos del libro. Para ver un capítulo determinado, se presiona el enlace correspondiente. Dentro de cada capítulo se encuentran los diagramas de carriles donde se presentan las actividades y productos a realizar en cada fase. Las definiciones y los productos que se colocarán en cada sección se tomarán y ejemplificarán de los que se realicen a lo largo de este escrito.

En cuanto a los artículos de apoyo de cada fase, se anexarán en la página principal de cada capítulo del libro, incluyendo su descripción y una liga a la fuente original de donde se tomó dicho artículo.

CAPÍTULO 1 Qué es la Ingeniería de Software.

La Ingeniería de Software es una disciplina que amalgama aspectos tecnológicos con aspectos administrativos, es decir, dirige al proceso de creación de objetos tecnológicos de alta calidad enmarcados en una organización y planeación cuidadosamente documentada, que al final resulta en un amplio compendio de datos y productos que fundamentan cada una de las decisiones tomadas a lo largo del ciclo que resultó con la obtención de un sistema computacional útil para el usuario final.

La planeación bien efectuada, y por supuesto documentada, implica que el producto sea entregado en el tiempo especificado desde un principio con el cliente, además el equipo de trabajo comenzará a acostumbrarse a medir los tiempos y detectar los posibles conflictos tanto técnicos como humanos que pudieran presentarse a lo largo del ciclo logrando una minimización de los problemas, lo que impacta directamente en un mayor tiempo dedicado a la realización del proyecto.

La Ingeniería de Software como asignatura en la Facultad de Ciencias está organizada de tal manera que el alumno experimente y aprenda lo qué significa trabajar en equipo, desempeñar un rol específico y enfocarse en una meta común. En particular la meta común es la elaboración de un sistema de alta calidad siguiendo los métodos propuestos por el libro “Ingeniería de Software Pragmática”.

A lo largo del proceso, que es de dos ciclos, los alumnos se familiarizarán con estos principios que en un futuro muy cercano regirán en su ámbito laboral. Descubrirán por sí mismos la finalidad de cada documento o producto que vayan generando, y lo más importante, se darán cuenta que hacer un sistema de software no sólo es implementarlo, sino que ésta última es sólo una de las fases del desarrollo.

Como se mencionó, en esta materia los alumnos formarán equipos, cada equipo constará de cinco integrantes que llamaremos *Ingenieros de Desarrollo* (ID), paralelamente cada integrante desarrollará a lo largo de cada ciclo uno de los siguientes roles: *Líder del Equipo* (LE), *Administrador de Apoyo* (AA), *Administrador de Calidad* (AC), *Administrador de Desarrollo* (AD) y *Administrador de Planeación* (AP). Cada rol tiene bien definidas sus responsabilidades y su perfil.

En cuanto al *Líder del Equipo*, entre las habilidades requeridas para desarrollar este rol están: destreza en el liderazgo de grupos, saber conciliar los intereses del grupo y saber motivarlo. En cuanto a las funciones del LE están construir un equipo efectivo y motivado, que cumpla con los compromisos establecidos, bien comunicado, resolverá los conflictos internos y además mantendrá informado al instructor sobre el progreso del equipo.

El *Administrador de Desarrollo* deberá tener marcadas habilidades y experiencia para la programación, conocimiento de varias herramientas auxiliares, lenguajes y ambientes de programación, entre las funciones del AD se pueden considerar el guiar al equipo a través del desarrollo de cada una de las fases, es responsable de producir un producto de calidad y explotar las habilidades y conocimientos de cada uno de los

miembros del equipo.

Ser organizado, saber planear las actividades y las tareas de todo el proceso, ser reconocido por su organización y su intolerancia a las faltas de compromiso, son algunas características del AP, entre sus funciones están el producir un plan preciso y adecuado para todo el equipo y para cada miembro del equipo, vigilar el estatus del plan frecuentemente y registrar los riesgos y eventos ocurridos durante el proceso.

Las habilidades requeridas para el AC son: saber identificar posibles fuentes de defectos, ser meticuloso en la revisión de los productos y sobre todo estar interesado en llevar un proceso de calidad, en cuanto a sus funciones se pueden identificar el establecer estándares y vigilar que se cumplan, revisar cada uno de los productos generados antes de que éstos sean entregados al instructor, dirigir las revisiones y registrar los defectos encontrados.

El AA deberá ser hábil en la búsqueda de herramientas alternativas y de apoyo para el equipo, tener la facilidad de poder explicar nuevos conocimientos de manera clara y sencilla y ser disciplinado en el manejo de las versiones de los productos, en cuanto a las funciones del AA se tienen el proporcionar al equipo las herramientas solicitadas, controlar el cambio de versiones de los productos y del sistema, así como mantener la correspondencia entre los requerimientos, el diseño, la construcción y las pruebas.

Cada uno de los roles participará activamente como *Ingeniero de Desarrollo* sin dejar de cumplir con las obligaciones del rol particular que desempeña.

Cronológicamente la estructura del curso que se imparte es la siguiente: al inicio del curso se presentará una introducción a la Ingeniería de Software y al trabajo en equipo. Posteriormente se dividirá el tiempo restante en dos ciclos, englobando el primero la totalidad del proceso descrito en el libro de la materia y con una duración aproximada de 10 semanas para las ocho fases. A continuación, el segundo ciclo tendrá una duración de cinco semanas, dejando en manos de cada equipo la decisión de cuáles productos generará. Al final del segundo ciclo se deberá entregar el sistema de software funcionando correctamente y con un alto grado de calidad. Inculcando finalmente en el alumno los principios de la Ingeniería de Software así como la experiencia de haber trabajado en equipo de manera exitosa al entregar un producto de calidad.

El proceso de desarrollo del sistema está basado, como ya se mencionó, en lo propuesto por el libro “Ingeniería de Software Pragmática”, que a su vez se guió en el proceso TSPi propuesto por Humphrey en el libro *“Introduction to the Team Software Process”*. Dicho proceso se llevará a cabo por ciclos, cada ciclo está compuesto por ocho fases, el número de ciclos es definido por las necesidades detectadas por el equipo de trabajo, en este caso serán dos fases. La carga de trabajo en cada fase será establecida por cada equipo y se recomienda que en la primera fase se concentre la mayor parte del trabajo debido a que la duración de ésta es el doble del segundo ciclo.

Las ocho fases son: Lanzamiento, Estrategia, Planeación, Especificación de los requerimientos, Diseño, Construcción, Pruebas y Cierre, las cuales describiremos brevemente a continuación.

Durante la fase de Lanzamiento se conformará el equipo de trabajo y se asignarán los roles a cada miembro del recién formado grupo, se conocerán los objetivos particulares de cada rol así como el del equipo que se esperaría alcanzar al final del ciclo.

En la fase de Estrategia el equipo de trabajo analizará desde varias perspectivas el problema para así definir un procedimiento para solventarlo a lo largo de las siguientes fases. En el caso del curso de Ingeniería de Software se busca una partición del proyecto en dos segmentos, dichos segmentos se cubrirán siguiendo la estrategia definida en esta fase.

La fase de Planeación, es cuando el equipo generará un plan en él que el trabajo sea repartido equitativamente entre los miembros del equipo durante cada fase y que se acople cronológicamente a las fechas de verificación del avance del proceso.

La Especificación de los requerimientos es la fase en la que el equipo tendrá como principal objetivo entender las necesidades del cliente, y con ello identificar con mayor facilidad las exigencias que el sistema cubrirá.

Durante la fase de Diseño se establecerán las relaciones existentes entre los componentes que se identificaron durante la fase anterior. Se definirá la base de la estructura funcional del sistema, aunque no por esto será la definitiva, ya que puede modificarse a lo largo del ciclo. Además se especificarán el ambiente de implementación y el estándar de diseño. Se generará una gran variedad de diagramas entre los que destacan los de distribución, de clases, de paquetes, de secuencia y de navegación.

Para la fase de Construcción, se considera sólo a la programación del sistema, es decir generar el software. Se detallarán los diagramas de clases previamente generados para su inmediata implementación, así como la generación del plan de pruebas con el cual se medirá el rendimiento y correcto funcionamiento del sistema.

A lo largo de la fase de Integración y Prueba del Sistema se pretende reportar y corregir los errores encontrados en el sistema, los errores serán reportados mediante el informe correspondiente a cada plan de pruebas. Se generarán los manuales de usuario, de instalación y el de mantenimiento y desarrollo.

Por último, la importancia de la fase de Cierre recae en la entrega del producto terminado, la evaluación a conciencia del producto y del desempeño de cada uno de los miembros del equipo. En esta fase quedarán reportadas aquellas experiencias vividas por los miembros del equipo y que consideren de utilidad para un ciclo posterior o un nuevo proyecto.

Cada una de estas fases se describirá detalladamente en los capítulos siguientes, así como los productos a generar y las recomendaciones de lecturas de artículos que se juzgan convenientes para profundizar en cada fase.

CAPITULO 2 Lanzamiento

2.1 Fase de Lanzamiento.

Durante esta fase se conformará el equipo de trabajo y se asignarán los roles a cada miembro. Es necesario que cada miembro de la nueva sociedad note, que una buena organización justo en el comienzo del proyecto, ahorrará tiempo y esfuerzo en las siguientes fases, además de que conocerán los objetivos particulares de cada rol, así como el del equipo que esperarían alcanzar al final del ciclo. Se propondrá un mecanismo para que el equipo se mantenga bien comunicado, es recomendable que dicho mecanismo de comunicación sea el establecimiento de una reunión semanal del equipo, en donde se analizarán todos los aspectos relacionados con el proyecto y estarán dirigidas por el *Líder del equipo*. Dichas reuniones son encaminadas por la *Agenda* y lo que haya acontecido en ella será reportado en las *Minutas*.

2.2 Productos generados

Los productos a generar en esta fase son cinco, de los cuales la *Agenda* y la *Minuta* corresponden a productos que se crearán cada semana ya que son fruto de las reuniones semanales que como su nombre lo indica, hay al menos una cada semana. La *Forma de Registro de Riesgos* también se generará todas las semanas ya que los riesgos pueden ser detectados o impactar al equipo en cualquier momento. Por otra parte el *Estándar de la documentación* buscará unificar la presentación de los productos y los *Objetivos del equipo, del producto, personales y de rol* se plasman para enfocar al equipo a un mismo destino y mostrar que el llevarlos a cabo tendrá un efecto positivo en el avance del proyecto.

2.2.1 Agenda

La *agenda* de las reuniones del equipo se generará cada semana. En ésta se plasmarán los puntos a tratar durante la reunión del equipo. Dichos puntos deben ser puestos en conocimiento de todo el equipo de trabajo para que se discutan y se determine una solución. La *agenda* contiene el lugar, la fecha y la hora de la reunión del equipo.

El objetivo de la *agenda* es difundir entre el equipo lo que se necesita tratar o hacer del conocimiento de los miembros, evitando con esto perder tiempo y aumentar la productividad de la reunión. El responsable de este producto es el *Líder del equipo*.

2.2.2 Minuta

La *minuta* del equipo le corresponde al *Líder del equipo*, en ella se desglosarán los temas tratados durante la reunión, se incluirá la fecha, hora y lugar de reunión, así como los asistentes y las incidencias de la misma. Una *minuta* tiene como principal finalidad registrar

los eventos acontecidos en la reunión, así cada miembro del equipo puede acceder a ella y solventar cualquier duda o en su defecto enterarse qué sucedió en dicha junta si es que no asistió.

En la *minuta* se anotarán las decisiones tomadas por lo miembros del equipo respecto a los asuntos propuestos en la *agenda*.

2.2.3 Objetivos del equipo, del producto, personales y de rol

Los o*bjetivos del equipo* son aquéllos que se desean alcanzar en conjunto, aquellas metas que de antemano se sabe que de no trabajar en equipo difícilmente lograrían. Estos objetivos proporcionan una base de lanzamiento para el proyecto, que de ser cumplidos satisfactoriamente darán como resultado un equipo de trabajo y un cliente satisfechos con lo realizado a lo largo del proyecto.

En cuanto a los *objetivos del producto*, se consideran todas aquellas metas que debe cumplir el producto terminado. Van más encaminados a cuestiones de funcionalidad del sistema desde el punto de vista del usuario final, cliente y por supuesto de los realizadores.

Los *objetivos personales* van de acuerdo a las preferencias y prioridades de cada miembro, se presupone que están en común acuerdo con el resto del equipo, así que dichas metas personales servirán como una estructura de cooperación entre los miembros para alcanzar los objetivos más generales.

Los objetivos de cada rol deben ser asignados previamente a la asignación de roles, que en este caso éstos ya están definidos. Cada miembro del equipo debe conocer no sólo las metas del rol que le fue asignado, sino las de los demás roles para garantizar el correcto funcionamiento del equipo a lo largo de cada ciclo ya que al desempeñar un rol se tienden a olvidar las metas del equipo. Es claro que cada miembro del equipo generará sus propios objetivos, así que el encargado de recopilar los documentos será el *Líder del equipo*.

2.2.4 Estándar de la documentación

Este documento es la base común de todos los documentos que se generarán a lo largo del proceso de desarrollo de software, debe ser respetado por cada uno de los miembros del equipo. Ayuda a mantener una buena organización, en él se deben especificar qué editor de texto se usará, el tamaño y tipo de letra y el formato de la página. Seguir un estándar implica muchas ventajas, el hecho de que los documentos estén unificados facilita su lectura y entendimiento, tómese en cuenta que los documentos muy probablemente serán utilizados como referencia en futuras ocasiones por personas que no necesariamente desarrollaron el sistema pero que requieren de alguna información para darle mantenimiento o escalarlo. Repercutiendo directamente en una manera de fácil acceso a la información.

Siempre será útil contar con un encabezado único que se utilizará en cada documento, dicho encabezado contendrá el nombre del equipo, la fase en que se encuentra el proceso, el miembro del equipo que realizó el documento y su rol.

Cabe señalar que un *estándar de documentación* sencillo resulta más fácil de respetar y facilita la interpretación al lector. El encargado de generar este documento es el *Administrador de Calidad*.

2.2.5 Forma de Registro de Riesgos

La *Forma de Registro de Riesgos*, es en la que se indicarán los posibles inconvenientes a los que el equipo se pudiera enfrentar a lo largo del ciclo.

Los riesgos identificados deben ser registrados de manera clara, una breve oración en infinitivo bastaría para ejemplificar dicho riesgo. Además se debe asignar una probabilidad de ocurrencia, *alta*, *media* o *baja*, a cada riesgo que se haya identificado y lo más importante, establecer el plan a seguir en caso de ocurrencia de alguno de estos riesgos. La realización de dicho plan de acción debe ser viable en cualquier momento, ya que un riesgo se puede prevenir pero no evitar.

Esta forma en particular resulta muy consultada a lo largo del proceso, ya que de haber sido realizada de manera responsable se podrá recurrir a ella y obtener una solución al contratiempo que haya surgido de manera rápida y eficaz sin necesidad de realizar una reunión de equipo para plantear el problema, ya que se presupone que la solución propuesta ya había sido aprobada por el resto del equipo.

El encargado de esta forma es el *Administrador de Planeación* y deberá actualizarla semana a semana.

2.3 Ejemplos de productos generados en esta fase

A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Agenda

Human Soft convoca a la reunión semanal del equipo.

Fecha: Jueves 31 de Agosto del 2006
Lugar: Taller de Ingeniería de Software
Hora: 15:00

1. Iniciar la reunión y pasar lista.
2. Leer la Agenda propuesta.
3. Plantear y discutir los objetivos del equipo.
4. Solicitar los objetivos personales y de rol.
5. Definir responsabilidades de cada rol.
6. Establecer el estándar de documentación a utilizar.
7. Informar sobre las tareas propuestas para cada rol.
8. Asignar las tareas para la siguiente semana en la forma Semana Personal de la siguiente semana.
9. Identificar los riesgos posibles del equipo y registrar en la forma Registro de Riesgos, proponer posibles soluciones.
10. Finalizar la reunión

Minuta

Human Soft informa de lo acontecido en la reunión semanal del equipo.

Fecha: Jueves 31 de Agosto del 2006.
Inicio de la reunión: 15:05

Lista de Asistentes (Al pase de lista):

Martha	AP
Hugo	AC
Roberto	AA
Miguel	LE
Rafael	AD

Acuerdos

- Se enviarán por correo electrónico los objetivos personales y de rol para que el líder se encargue de la entrega al instructor.
- Los días de reunión se entregará la Forma Semana Personal de la semana y se traerá la forma Semana Personal de la siguiente semana.
- AD pidió los conocimientos básicos de herramientas de desarrollo a cada integrante para determinar qué tan preparados estamos y tomar las determinaciones correctas.
- AC se comprometió a enviar un ejemplo del estándar establecido en la reunión para determinar si faltan formatos que considerar.
- Se trató el problema del lugar de reunión, se propuso una alternativa al lugar previamente acordado.

Fin de la reunión: 16:32

Objetivos del Equipo

Todo equipo debe de compartir una serie de objetivos generales para la realización exitosa de su proyecto, éstos proporcionan la base para desarrollar un producto satisfactorio tanto para el usuario o cliente como para el propio equipo; teniendo esto claro se definen los objetivos para el ciclo 01 del equipo en el siguiente listado:

- Conformar un buen equipo de trabajo.
- Aprobar el primer y segundo ciclo del curso.
- Que se establezca una buena comunicación y respeto entre todos los miembros del equipo.
- Que se asuman cabalmente las responsabilidades de todos como equipo.

Objetivos del Producto

Dadas las especificaciones del producto, se proponen los siguientes objetivos del mismo:

- Generar un producto que funcione adecuadamente.
- Que el producto cumpla con las especificaciones requeridas al concluir el primer y segundo ciclo.
- Que sea un producto de calidad.
- Que sea un producto escalable, entendible y fácil de usar.

Objetivos Personales

Dados los objetivos del equipo, es necesario que cada uno de los miembros del equipo tenga sus propios objetivos y metas. A continuación se describen los objetivos de cada miembro del equipo:

Martha (AP)

- Aprender a trabajar en equipo.
- Entregar a tiempo mi trabajo asignado.
- Colaborar con todos mis compañeros del equipo.
- Tener respeto y tolerancia hacia mis compañeros.

Hugo (AC)

- Ser responsable con el rol que voy a desempeñar.
- Tratar de terminar en el tiempo establecido cada uno de los documentos a elaborar.
- Poner todo lo que este de mi parte para conformar un buen equipo de trabajo.
- Tener una excelente comunicación con los demás miembros del equipo para establecer una buena sinergia.

Roberto (AA)

- Aprender a realizar trabajo en equipo.
- Adquirir los conocimientos para realizar un buen producto de software.
- Conocer más herramientas de software.
- Alcanzar las metas propuestas.

Miguel (LE)

- Adquirir y aplicar buenas prácticas de programación.
- Obtener herramientas eficientes para el desarrollo del producto.
- Supervisar que el desarrollo del producto sea de calidad.
- Entregar en tiempo y forma las tareas que me correspondan.

Rafael (AD)

- Cumplir puntualmente con las actividades encomendadas, tanto por parte del profesor y los ayudantes como con mi equipo de trabajo.
- Lograr una buena comunicación y una buena relación de trabajo con mis compañeros de equipo.
- Que las técnicas y nuevos conocimientos en general adquiridos en este ciclo, los pueda dominar para que me sean de provecho para futuras actividades escolares y profesionales.
- Dedicarle tiempo suficiente para obtener una alta calificación en el curso.

Objetivos de los Roles

En conjunto con los objetivos de los miembros del equipo y los del propio equipo también son necesarios los objetivos de los roles una vez que éstos han sido asignados a los miembros del equipo, a continuación se enumeran dichas metas:

Líder del Equipo

- Construir y mantener un equipo efectivo.
- Motivar a todos los miembros a participar activamente en el proyecto.
- Resolver los conflictos que pudieran presentarse en el equipo.
- Mantener al instructor informado del progreso del equipo.
- Convocar y dirigir las reuniones del equipo de forma eficaz.

Administrador de Desarrollo

- Entender el proceso de desarrollo.
- Dirigir al equipo en las actividades de desarrollo en cada una de las fases.

Administrador de Planeación

- Producir un plan para el equipo y para cada miembro del mismo que sea completo, preciso y adecuado.
- Dar seguimiento al plan cada semana.
- Dar seguimiento y detectar los riesgos que se presenten en el proyecto.

Administrador de Calidad

- Hacer que todos los miembros del equipo cumplan con un nivel de calidad en sus productos.
- Conseguir que todos los miembros del equipo generen productos de calidad.
- Que se lleven a cabo las reuniones de revisión entre colegas y se registren los defectos en las formas REGD.
- Hacer que se corrijan los defectos registrados.

Administrador de Apoyo

- Conseguir las herramientas necesarias para apoyar al equipo en su trabajo.
- No permitir que se hagan cambios no autorizados a productos ya aprobados.
- Facilitar la reutilización de lo desarrollado en el primer ciclo para agilizar el segundo ciclo.

Estándar de la Documentación

El objetivo de realizar el estándar de documentación es que toda la información generada a lo largo de proyecto respete un patrón, para que posteriormente se pueda integrar fácilmente en un documento final.

Estructura de los documentos

Los documentos deberán de tener una portada que incluya el nombre del equipo y de la fase, el ciclo correspondiente, un índice para presentar la información que contienen, y se apegarán de tener la siguiente estructura:

- Encabezado de los documentos del proyecto.

El encabezado estará compuesto por el nombre del documento con su versión correspondiente del lado izquierdo con negrita y cursiva letra Times New Roman de 12 puntos; Así como el logotipo del equipo del lado izquierdo.

- Nombre de los documentos en el proyecto.

Solo se pondrá en la primera página con letra Arial de 18 puntos, centrado, con mayúsculas y negrita. Además de contener los datos del documento como son el identificador, versión, descripción, fecha de creación y autor los cuales serán con letra Arial de 12 puntos y las etiquetas serán con negrita y justificadas a la derecha. Con una línea de 4 _ puntos de color gris como separador.

- Pie de página para los documentos del proyecto.

Del lado izquierdo tendrá el nombre de la fase a la que pertenece el documento y del lado derecho se colocará el rol responsable del documento, ambas anotaciones deberán de estar escritas con letra Times New Roman de 12 puntos

- Títulos.

Los títulos serán escritos con letra Arial de 16 puntos con negrita y alineados al centro.

- Subtítulos.

Los subtítulos serán escritos con letra Arial de 12 puntos con negrita y alineados a la izquierda.

- Párrafos.

Los párrafos serán justificados, con letra Arial de 12 puntos, iniciando con una sangría de un tabulador con un separador de líneas sencillo, dejando pasar un renglón entre el subtitulo y el párrafo.

- Viñetas y numeraciones.

Las viñetas utilizadas para listados serán puntos negros, en el caso que se requiera de numeración se utilizarán números arábigos seguidos de un punto y seguido.

Márgenes del documento

Los márgenes que a los que deberán ajustarse los documentos son los siguientes:

Superior:	2.5 cm
Inferior:	2.5 cm
Encabezado:	1.25 cm
Pie de Pagina:	1.25 cm
Izquierda:	2.0 cm
Derecha:	2.0 cm

Forma de Registro de Riesgos

El desarrollo del plan de riegos del proyecto se presenta en la siguiente tabla, consideramos riesgos directos e indirectos, la probabilidad de ocurrencia, el indicador de riesgo y la estrategia para manejar el riesgo que consiste en plantear un plan para solucionar un riesgo.

Riesgo	*Probabilidad de Ocurrencia*	*Gravedad*	*Estrategia para manejar el riesgo*
Falta de papelería.	Baja	Alta	Asignar como encargado de abastecimiento al AA.
Falta de impresora.	Baja	Alta	Ubicar un centro de impresión cercano y disponible, turnado al AA.
Enfermedad de algún miembro del equipo.	Baja	Media	Repartición del trabajo faltante con el consentimiento del afectado.
Exceso de trabajo de otras materias.	Alta	Alta	Este es un aspecto que se solucionará individualmente y a conciencia de cada elemento de este equipo.
Falta de conjunción del equipo.	Baja	Alta	Unidad y cooperación por parte de todos. Al final esto es una actividad humana y es latente ese riesgo.
Falta de equipo de cómputo.	Media	Alta	Obtener las tarjetas de acceso al laboratorio de Ing. Software y minimizar el impacto de esta carencia.
Falta de paquetería en las máquinas.	Media	Alta	Asignar como encargado de obtención de paquetería al AA.
Sucesión de días inhábiles.	Alta	Baja	Repartición del trabajo en los días que si laboraremos.
Malentendidos en la uniformidad de los documentos de entrega.	Baja	Baja	Para ello establecimos un estándar en la documentación, así como en la papelería de uso interno del equipo.

2.4 Artículos de apoyo para la fase

Para esta fase se recomienda la lectura del artículo "La cultura de medir" donde se esboza la necesidad de establecer métricas a lo largo de un proceso de desarrollo de software. Mediante las métricas podemos graduar nuestros procesos, ya sea espacial o temporalmente, no sólo para mantener la planeación establecida en orden, sino también para tomar buenas decisiones en base a la información que arroje el análisis final de las métricas realizadas. Para ello es necesario que la toma de métricas sea real y registrada lo más concretamente posible.

En el proceso que se seguirá se tienen formas auxiliares donde se registran el número de defectos, tamaño de los productos, tiempo invertido en cada semana y finalmente toda esa información se concentra en un informe semanal del equipo o en el Informe de Mediciones que el equipo deberá analizar para aislar la información que considere mejorará el desempeño en ciclos o procesos posteriores.

Cuellar, Luis.
"*La cultura de medir*", en: Software Guru. 02. 05 México D.F. 2006. p.10

CAPITULO 3 Estrategia

3.1 Fase de Estrategia.

Durante esta fase el equipo de trabajo analiza desde varias perspectivas el problema para así definir una táctica de ataque. En el caso del curso de Ingeniería de Software se busca partir el proyecto en dos ciclos, dichos segmentos se cubrirán siguiendo la estrategia definida justo en esta fase.

3.2 Productos generados

Al finalizar esta fase se habrán generado cinco documentos, entre ellos la *Forma Estrategia* que es en la que se establece la táctica que se eligió para atacar y solventar el problema. Se definirá un *Depósito de productos del equipo* en él que se almacenarán cada uno de los productos generados en el proceso, por otro lado el *Plan de la Configuración* es el listado de todos los productos que se planean generar a lo largo del ciclo y el responsable asignado a éstos, este documento es la base de la *Forma del informe semanal del estado de la configuración* en el que se registra el estado del producto y su ubicación en el depósito del equipo. Por último, la *Forma del informe semanal del estado de los cambios* tiene como objetivo mantener al tanto al equipo de nuevos cambios o de la situación de los anteriormente propuestos.

3.2.1 Forma Estrategia

Antes de realizar esta forma es necesario que el equipo analice las necesidades del sistema y defina en qué ciclo serán abastecidas. Para esto se puede auxiliar mediante “gráficas de dependencias” con las que se marcarán las tendencias de cómo atacar al problema. El equipo discutirá los pros y contras de cada una de las propuestas. Aquélla que resulte satisfacer mejor las necesidades del equipo será presentada en la *Forma Estrategia*.

Esta forma está compuesta por un listado de las necesidades o funcionalidades identificadas y el ciclo en que se decidió satisfacer cada una, es muy recomendable anexar el diagrama de necesidades que se generó en el análisis previo.

3.2.2 Depósito de productos del equipo

Crear un depósito con los productos generados desde el comienzo, facilitará a los miembros del equipo y al instructor el acceso a cualquier producto que sea requerido. El equipo tendrá dos depósitos de productos, uno digital y el otro en papel, ambos depósitos serán gestionados por el *Administrador de Apoyo*.

Además se deberán especificar la organización y estructura del depósito digital, como puede ser el diseño del esqueleto del árbol de directorios del depósito. El objetivo de que el depósito contenga todo esto es para que los miembros del equipo tengan acceso para apoyarse, facilitar y mejorar sus tareas y actividades dentro del proyecto.

Para el caso del depósito en papel, se debe diseñar una portada que contendrá los datos del equipo (nombre del equipo, nombre de los integrantes y roles, fase, ciclo) estos documentos se pueden agrupar en una carpeta que se irá engrosando conforme avance el proyecto y que estará bajo cuidado del *Administrador de Apoyo*.

Es aconsejable que la carpeta con los documentos generados esté dividida en dos segmentos, el primero contendrá a cada uno de los ciclos con sus respectivas fases, y la segunda servirá para almacenar los documentos que son generados semana con semana que sirven para llevar un control interno del equipo, como lo son las *agendas*, las *minutas*, las *formas semana personal* y *del equipo.*

3.2.3 Plan de configuración

En este documento se enmarcan los productos que se generarán fase por fase, así como el responsable de cada uno de éstos. Se incluye el estándar de nombrado, éste debe incluir principalmente nombre del documento, ciclo, fase y versión, se podrán agregar datos que el equipo considere significativos para el nombrado. El estándar de nombrado es necesario para obtener una interpretación ágil y concreta.

Conforme el proyecto avance, surgirá la necesidad de hacer cambios a productos que previamente se habían aprobado, por ello se recomienda definir lo antes posible un comité que controle la ejecución de estos cambios. Dicho comité estará conformado por dos miembros del equipo y estará encargado de aprobar o rechazar las solicitudes de cambio recibidas, las cuales pueden ser emitidas por cualquier miembro del equipo.

Se busca que el comité tenga pocos miembros para que pueda ser convocado de manera rápida y eficiente. En caso de que la solicitud sea aceptada, el encargado del depósito, *el Administrador de Apoyo,* deberá proporcionar una copia del documento que se pretende modificar a aquél que haya solicitado el cambio. Ya que el cambio haya sido realizado, la nueva versión del documento será revisada por el *Administrador de Calidad*, posteriormente será corregido si es necesario y finalmente se agregará al depósito del equipo, como cualquier otro documento.

La petición de cambio a un producto arriba mencionada se realizará utilizando la *Forma de solicitud de cambios*, dicha forma debe contener la información necesaria para ubicar al producto en cuestión dentro del depósito, esto es, su nombre y dirección. Se deberán argumentar las razones del cambio describiendo qué defecto se encontró, la descripción del cambio propuesto para corregir el defecto, analizar y detallar las perturbaciones posibles al realizar dicho cambio en los demás productos, el dictamen del comité (A*probado, Rechazado o en Revisión)*, en caso de ser aprobado se indicará quién es el responsable del producto y la fecha en que se aprobó y en caso de ser rechazado deberán documentarse las razones de la negativa a la propuesta del cambio.

3.2.4 Forma del informe del estado de la configuración

En esta forma se incluyen el listado de los elementos de la configuración, la versión más reciente y el estado en que se encuentra el elemento. Este documento se genera cada que concluye un ciclo y está a cargo del *Administrador de Apoyo*.

Se recomienda organizarlo por columnas, siendo la primera la que contenga el elemento de la configuración así como su ubicación en el depósito, la siguiente columna contendrá a la última versión y en la tercer columna se indicará si está *Completo* o *En Proceso*, es decir el estado del elemento.

La función principal de este documento es ayudarnos a llevar el control, al termino de cada ciclo, de la situación en que se encuentran los productos hasta ese momento generados.

3.2.5 Forma del informe semanal del estado de los cambios

El *Informe semanal del estado de los cambios* debe indicar si algún cambio aprobado por el comité de control de cambios y descrito en la *Forma de solicitud de cambios* ha sido *terminado* satisfactoriamente o sigue *en proceso.* Esta forma es necesaria para mantener informado al equipo, ya que es muy probable que un cambio no sólo afecte al producto en sí, recordemos que todo esto fue considerado en la misma *Forma de solicitud de cambios* en la sección de *impactos del cambio*. Estas formas deben ser almacenadas en el depósito del equipo.

3.3 Ejemplos de productos generados en esta fase

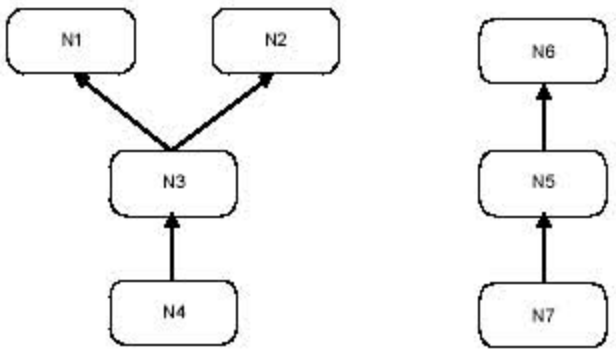
A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Forma Estrategia

La estrategia que se llevará a cabo en el primer ciclo se eligió contemplando la mayoría de las funcionalidades básicas del proyecto. La implementación de las necesidades faltantes, que se consideran más sencillas se realizaran en el segundo ciclo además en el segundo ciclo se pretende hacer cambios o modificaciones para el mejoramiento del producto. El equipo prefirió cubrir la mayor parte de las funcionalidades en el primer ciclo.

Tabla de Necesidades

Necesidades	Ciclo	
	1	2
N1. El sistema permitirá al usuario registrar datos del paciente (alumno).	X	
N2. El sistema permitirá al usuario, registrar datos del médico.	X	
N3. El usuario podrá registrar las citas de los pacientes.	X	
N4. El sistema permitirá realizar consultas.	X	
N5. El sistema deberá, contar con una interfaz gráfica.		X
N6. El sistema deberá estar codificado en Java cumpliendo con el estándar de codificación específico y se usarán tablas para guardar los datos que sean persistentes.		X
N7. El sistema deberá operar en el ambiente de Linux y Windows.		X
Necesidades totales por ciclo.	4	3



Gráfica de dependencias

Depósito de Productos del Equipo

El depósito digital del equipo tendrá la siguiente organización.

HumanSoft ¬
- Proyecto ¬
  - Ciclo1¬
    - LAN1 ¬
      - Documentos ¬
      - …
      - Formas ¬
    - EST1 ¬
    - …
    - CIERRE1 ¬
  - Ciclo2¬
    - LAN2 ¬
    - …
    - CIERRE2 ¬
- Libro ¬
  - Capitulo00
  - …
  - Capitulo15
  - Formas
- Prácticas ¬
  - Práctica00
  - …
  - Práctica22
- Directorio

El objetivo de que el depósito contenga todo esto es para que las faltas de documentación sean mínimas, deseablemente nulas, y que los miembros del equipo tengamos acceso para apoyarnos, facilitar y mejorar nuestras tareas y actividades dentro de Human Soft. Además un directorio con los datos de los miembros para que puedan ser localizados en caso de alguna contingencia con el proyecto,

Por otro lado el estándar de nombrado para la interpretación ágil y concreta es el siguiente.

fase ciclo componente autor versión

{LAN,EST,PLAN,…}{1,2,…}{Agenda, Minuta,…}{LE, AC, AP,…}{1.0,….}

Para el caso en el que los documentos sean de carácter individual y se necesite uno por cada miembro del equipo esto será agregado después del nombre del componte.

Plan de la Configuración

El plan de configuración se llevara a cabo basado en los siguientes elementos:

1. Plan del manejo de configuración de software

- Los documentos y los productos tendrán un nombre y una ubicación única dado que se tendrán carpetas virtuales para los ciclos y dentro de ellas se encontrarán las etapas, teniendo así una clasificación de los documentos que formarán parte de la línea base.
- Para tener el control de los productos de línea base se deberá de llenar la forma solicitud de cambios, para cualquier cambio que se necesite realizar, y siendo esta forma aprobada por el comité de control de cambios.
- Si se aprueba algún cambio a los productos de línea base se deberá proporcionar el producto correspondiente por parte del Administrador de Apoyo (AA) al responsable de dicho producto para realizar los cambios pertinentes.

2. Sistema para la línea base

- Los productos de línea base son aquellos productos revisados por el Administrador de Calidad (AC) y aprobados por el instructor; estos estarán bajo el plan de configuración descrito actualmente.
- Se tendrá un respaldo en la computadora del Administrador de Apoyo, así como un respaldo en CD de los productos de línea base que se han desarrollado, además de tener una copia en el servidor en donde los miembros del equipo podrán consultar los productos sin poder realizar modificaciones.
- La administración de los archivos se realizará tanto manual como automáticamente, efectuando la labor manual el Administrador de Apoyo por medio de la organización y verificación de los productos el mismo, y la labor automática será efectuada con CVS (Concurrent Version System) supervisada por el Administrador de Apoyo.

3. Tabla de componentes que se generarán a través del ciclo.

Fase	Componente	Responsable
Lanzamiento	Agenda	**LE**
	Minuta	**LE**
	Objetivos del equipo, del producto, personales y de rol	**LE**
	Estándar de documentación	**AC**
	Forma de registro de riesgos	**AA**
Estrategia	Forma estrategia	**AD**
	Depósito de productos del equipo	**AA**
	Plan de configuración	**AA**
	Forma del informe del estado de la configuración	**AA**
	Forma del informe semanal del estado de los cambios	**AA**
Planeación	Plan del equipo	**AP**
	Forma de registro de defectos	**AC**
Especificación de los Requerimientos	Texto con la definición del problema	**AD**
	Glosario de términos	**AD**
	Requerimientos no funcionales	**AD**
	Diagrama general de casos de uso	**AD**
	Detalle de los casos de uso	**ID**
	Prototipo de la interfaz del usuario	**ID**
	Plan de pruebas del sistema	**ID**
Diseño	Especificación del ambiente de implementación	**AD**
	Estándar de diseño	**AD**
	Arquitectura con diagrama de paquetes	**AD**
	Diagrama de distribución	**AD**
	Plan de pruebas de integración	**AD**
	Diagramas de clases	**ID**
	Diagramas de secuencia	**ID**
	Diagrama de estados	**ID**
Implementación	Diagramas de clases refinados	**ID**
	Código para las clases	**ID**
	Plan de pruebas unitarias	**ID**
Integración y Prueba del Sistema	Reporte de las pruebas unitarias	**AA-AC-AD**
	Manual de usuario	**ID**
	Manual de instalación	**ID**
	Manual de desarrollo y mantenimiento	**ID**
Cierre	Lecciones aprendidas y sugerencias de mejoras	**AC-AP**
	Forma de evaluación del equipo y personal	**LE**
	Informe de mediciones	**AC-AP**
	Forma del informe del estado de la configuración	**AA**

Forma del Estado de la Configuración

En base a la tabla de productos elaborada en el Plan de la Configuración, se presenta el último estado de cada uno de los productos hasta ahora generados.

Componente y localización	Versión	Estado
HumanSoft/Proyecto/Ciclo1/LAN1/Agenda	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Minuta	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Objetivos	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Estándar	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Forma de registro de riesgos	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Forma estrategia	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Depósito	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Plan de configuración	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Informe del estado de la configuración	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Informe semanal estado cambios	1.0	Terminado
Plan del equipo	-	-
Texto con la definición del problema	-	-
Glosario de términos	-	-
Requerimientos no funcionales	-	-
Especificación de requerimientos del software	-	-
Diagrama general de casos de uso	-	-
Detalle de los casos de uso	-	-
Prototipo de la interfaz del usuario	-	-
Plan de pruebas del sistema	-	-
Especificación del ambiente de implementación	-	-
Estándar de diseño	-	-
Arquitectura con diagrama de paquetes	-	-
Diagrama de distribución	-	-
Plan de pruebas de integración	-	-
Diagramas de clases	-	-
Diagramas de secuencia	-	-
Diagrama de estado	-	-
Diagramas de clases refinados	-	-
Código para las clases	-	-
Plan de pruebas unitarias	-	-
Reporte de las pruebas unitarias	-	-
Manual de usuario	-	-
Manual de instalación	-	-
Manual de desarrollo y mantenimiento	-	-
Lecciones aprendidas y sugerencias de mejoras	-	-
Forma de evaluación del equipo y personal	-	-
Informe de mediciones	-	-
Forma del informe del estado de la configuración	-	-

Forma del Informe Semanal del Estado de los Cambios

El equipo requiere de un mecanismo para controlar los cambios que resulten necesarios a lo largo del proceso, para ello se ha decidido nombrar un comité que evalúe los cambios propuestos y que actué de manera eficiente, rápida y responsable, además de definir una forma para la solicitud de cambios.

Comité de control de Cambios.

El comité de control de cambios estará integrado por el Administrador de Desarrollo y el Administrador de Apoyo.

Cabe señalar que ante la ausencia del Administrador de Desarrollo su lugar sería ocupado por el Líder del Equipo, además consideramos que en caso de ser un cambio que tenga un impacto mayoritario en el proyecto es necesaria la presencia del miembro que solicitó el cambio para que exprese los motivos concretos de su solicitud, así como el análisis del impacto.

El procedimiento para la solicitud de un cambio se define de la siguiente manera: La solicitud se entregará directamente al líder del equipo, quien a su vez evaluará la urgencia de dicho cambio. En caso de ser una necesidad de primer orden se convocará al Comité de control de cambios lo más pronto posible, de no ser así, se presentará dicha solicitud durante la junta semanal del equipo, que tiene lugar en el Laboratorio de Ingeniería de Software y bases de datos los días miércoles a las 15:00.

Forma de Solicitud de cambios.
Solicitud de cambios.

Equipo Human Soft	**Semana** 11	**Ciclo** 2

Información del componente

Nombre Guión del Ciclo 2 Responsable Human Soft
Dirección del respaldo HumanSoft/Proyecto/Ciclo2/LAN2/Guion del Ciclo02

Razones para hacer el cambio

Creemos que el reacomodo temporal de algunos documentos beneficiará la calidad del sistema.

Impactos del cambio

Mejoras en la planeación y realización de los documentos.

Descripción del cambio

Los Planes de Prueba de Integración de Sistema y Unitarias serán elaborados en la Fase IMPL2.

La Aplicación de las Prueba de Integración de Sistema y Unitarias será efectuada en la Fase PRUE2 .

La Documentación de Usuario será elaborada en la Fase de PRUE2.

Estado del cambio

■ Aprobado ● Rechazado ● En proceso

Aprobado por

Responsable del componente 23 de Noviembre de 2005 fecha
Comité de control de cambios 23 de Noviembre de 2005 fecha

3.4 Artículos de apoyo para la fase

Para esta fase se recomienda la lectura de un artículo como complemento, es el titulado “Administración de Cambios del Software”, en este artículo se describe los que es un cambio al software y el proceso formal para solicitar un cambio mediante un diagrama de estados que representa el ciclo de vida de una solicitud de cambio.

Además presenta de manera introductoria herramientas para automatizar la administración de los cambios tanto a documentos como al código, dos ejemplos de estas herramientas son TFS e IBM Rational ClearQuest. Todo esto con el objetivo de mantener una estabilidad en los productos generados controlando los cambios que se realizan a lo largo del proceso.

Bautista, Y., Calleja, R.
“*Administración de cambios del software*”, en: Software Guru. 03. 02
México D.F. 2007. pp.30-32

CAPITULO 4 Planeación

4.1 Fase de Planeación.

En esta fase el trabajo fuerte recae sobre el *Administrador de Planeación,* pero es necesaria la colaboración de todo el equipo para generar un buen plan en el que el trabajo sea repartido equitativamente entre los miembros del equipo durante cada fase.

4.2 Productos generados

En esta fase se generarán dos productos, el primero de ellos y cuya importancia es prominente, es el *Plan del Equipo,* en el que se indican cronológicamente las actividades que el equipo realizará a lo largo de lo que dure el ciclo, posteriormente se generará un plan individual para cada miembro del equipo, en él se especificarán las actividades que le corresponden según el criterio del equipo. El segundo documento es la *Forma de Registro de Defectos,* en la que el *Administrador de Calidad* indicará qué defectos se encontraron en los productos de esa semana, esta forma define qué calidad va teniendo el proceso.

4.2.1 Plan del equipo

Este documento es coordinado por el *Administrador de Planeación,* en él se programan las actividades del equipo en el periodo de tiempo que durará el proyecto. Es conveniente utilizar un *diagrama de Gantt* en el que colocaremos en la columna de tareas aquellos productos que se deben generar y las actividades a realizar durante el ciclo, posteriormente se asignará una porción de tiempo a cada una de las actividades y productos. Al comenzar esta asignación parecerá arbitraria ya que el equipo desconoce cuánto tardará en generar cada documento, por esta razón es muy importante comenzar a tomar métricas lo más pronto posible. Éstas ayudarán a esbozar el tiempo requerido para la creación de cada producto en ciclos posteriores.

Al finalizar con el *Plan del equipo,* el *Administrador de Planeación* deberá dividir éste para conformar el plan individual que asignará a cada uno de los miembros del equipo.

4.2.2 Forma de registro de defectos

Esta forma es el resultado de las anotaciones de los defectos que se encontraron durante la revisión entre colegas. Cabe resaltar que para la revisión entre colegas se escribe una *lista de verificación,* esta lista es generada por el *Administrador de Calidad* y contiene los aspectos a revisar en cada producto.

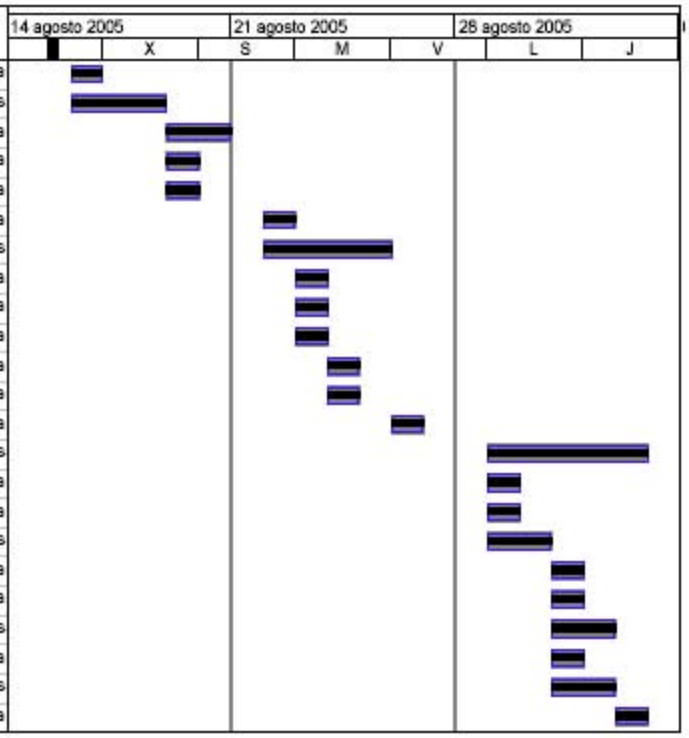
Por cada revisor se genera una forma, en ella se registra la fecha y el producto que se revisa, lo más importante es describir el defecto encontrado así como el estado de éste.

4.3 Ejemplos de productos generados en esta fase

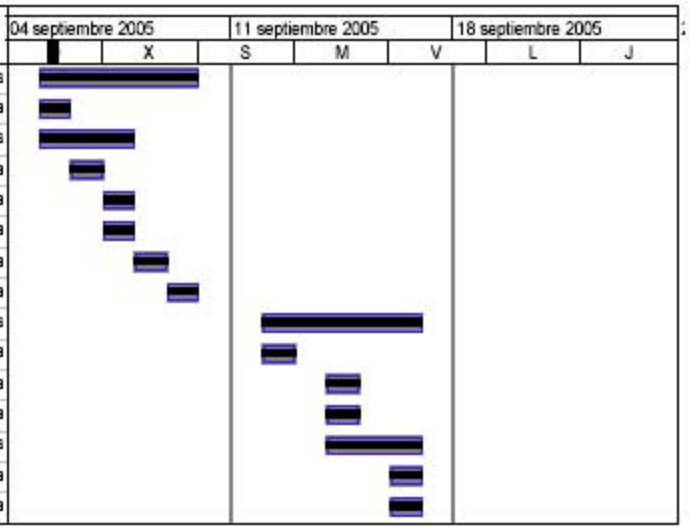
A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos. Los diagramas de Gantt para los planes individuales y del equipo fueron realizados con Microsoft Project.

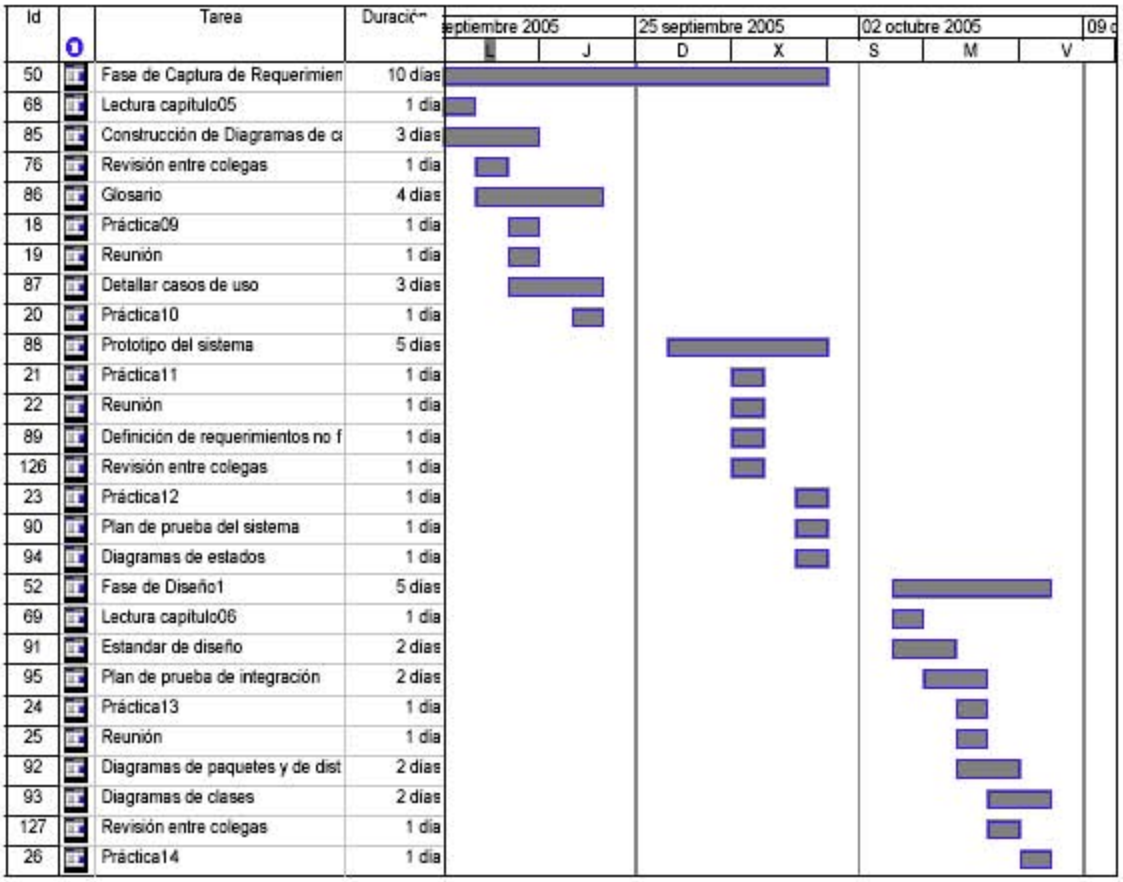
Plan del Equipo

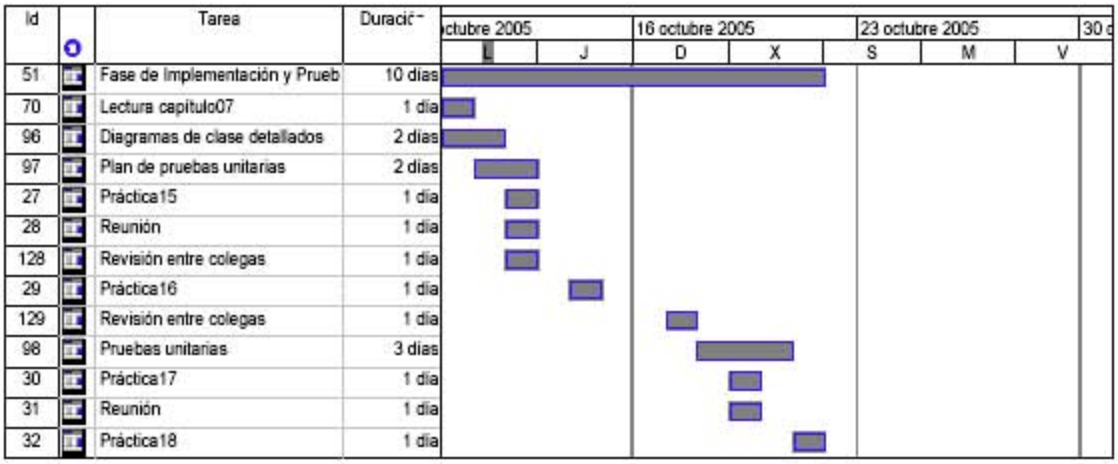
Id	Tarea	Duración
1	Entrega de Temarios	1 día
2	Introducción a la IS	3 días
3	Lectura cápitulo 00	1 día
4	Práctica 00	1 día
5	Lectura artículo	1 día
43	Lectura cápitulo 01	1 día
44	Planteamiento del Problema	4 días
45	Establecer roles	1 día
46	Principios de trabajo en equipo	1 día
84	Mediciones	1 día
6	Práctica01	1 día
7	Reunión	1 día
8	Práctica02	1 día
47	Fase de Lanzamiento1	5 días
65	Lectura capítulo02	1 día
73	Lectura Manual de Rol	1 día
80	Definición de objetivos	2 días
9	Práctica03	1 día
10	Reunión	1 día
81	Definición de Responsabilidades	2 días
82	Estándar de documentación	1 día
83	Riesgos del proyecto	2 días
11	Práctica04	1 día



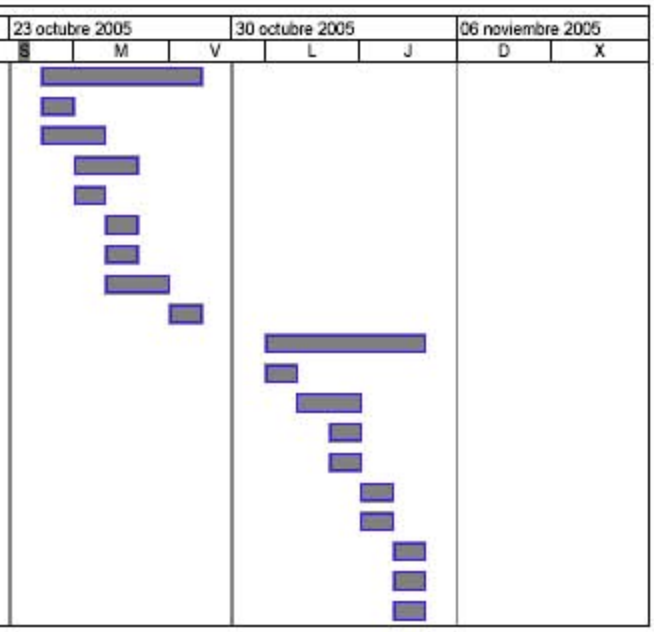
Id	Tarea	Duración
48	Fase de Estrategia1	5 días
66	Lectura capítulo03	1 día
79	Proposición de estrategias	3 días
77	Depósitos de documentos	1 día
12	Práctica05	1 día
13	Reunión	1 día
78	Plan de Configuración	1 día
14	Práctica06	1 día
49	Fase de Planeación1	5 días
67	Lectura capítulo04	1 día
15	Práctica07	1 día
16	Reunión	1 día
74	Producción de plan de equipo	3 días
17	Práctica08	1 día
75	Plan de Calidad	1 día




Id
Tarea
Duración
25 septiembre 2005
02 octubre 2005
L J D X S M V
50 Fase de Captura de Requerimien 10 días
68 Lectura capitulo05 1 día
85 Construcción de Diagramas de cl 3 días
76 Revisión entre colegas 1 día
86 Glosario 4 días
18 Práctica09 1 día
19 Reunión 1 día
87 Detallar casos de uso 3 días
20 Práctica10 1 día
88 Prototipo del sistema 5 días
21 Práctica11 1 día
22 Reunión 1 día
89 Definición de requerimientos no f 1 día
126 Revisión entre colegas 1 día
23 Práctica12 1 día
90 Plan de prueba del sistema 1 día
94 Diagramas de estados 1 día
52 Fase de Diseño1 5 días
69 Lectura capitulo06 1 día
91 Estandar de diseño 2 días
95 Plan de prueba de integración 2 días
24 Práctica13 1 día
25 Reunión 1 día
92 Diagramas de paquetes y de dist 2 días
93 Diagramas de clases 2 días
127 Revisión entre colegas 1 día
26 Práctica14 1 día


Id
Tarea
Duración
16 octubre 2005
23 octubre 2005
L J D X S M V
51 Fase de Implementación y Prueb 10 días
70 Lectura capitulo07 1 día
96 Diagramas de clase detallados 2 días
97 Plan de pruebas unitarias 2 días
27 Práctica15 1 día
28 Reunión 1 día
128 Revisión entre colegas 1 día
29 Práctica16 1 día
129 Revisión entre colegas 1 día
98 Pruebas unitarias 3 días
30 Práctica17 1 día
31 Reunión 1 día
32 Práctica18 1 día

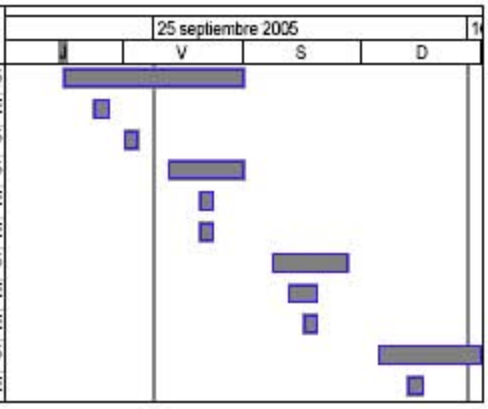
Id	Tarea	Duración
53	Fase de Prueba1	5 días
71	Lectura papítulo08	1 día
99	Aplicación de pruebas de integra	2 días
100	Aplicación de pruebas de sistema	2 días
130	Revisión entre colegas	1 día
33	Práctica19	1 día
34	Reunión	1 día
101	Producción de manuales del siste	2 días
35	Práctica20	1 día
54	Fase de Cierre1	5 días
72	Lectura capítulo09	1 día
104	Informe de configuración	2 días
36	Práctica21	1 día
37	Reunión	1 día
63	Primer Examen Parcial	1 día
103	Identificar lecciones aprendidasy	1 día
38	Práctica22	1 día
102	Evaluación personal y del equipo	1 día
122	Entrega de la primera versión	1 día



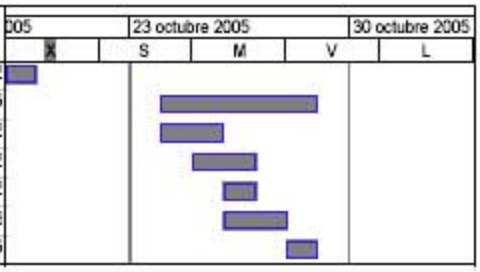
Plan Individual

El siguiente plan individual corresponde a Hugo que desempeña el rol de Administrador de Apoyo.

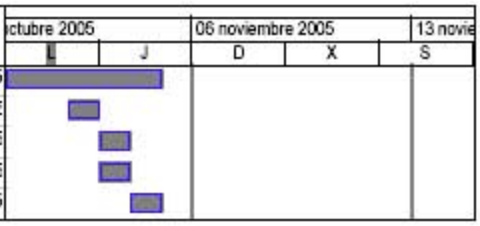
Id	Tarea	Comienzo	Fin
8	Fase de Captura de Requerimientos1	lun 19/09/05	vie 30/09/05
1	Reunión	mié 21/09/05	mié 21/09/05
14	Revisión entre colegas	vie 23/09/05	vie 23/09/05
15	Prototipo del sistema	lun 26/09/05	vie 30/09/05
2	Reunión	mié 28/09/05	mié 28/09/05
16	Definición de requerimientos no funcionales	mié 28/09/05	mié 28/09/05
10	Fase de Diseño1	lun 03/10/05	vie 07/10/05
17	Plan de prueba de integración	mar 04/10/05	mié 05/10/05
3	Reunión	mié 05/10/05	mié 05/10/05
9	Fase de Implementación y Pruebas Unitarias	lun 10/10/05	vie 21/10/05
4	Reunión	mié 12/10/05	mié 12/10/05



Id	Tarea	Comienzo	Fin
5	Reunión	mié 19/10/05	mié 19/10/05
11	Fase de Prueba1	lun 24/10/05	vie 28/10/05
18	Aplicación de pruebas de integración	lun 24/10/05	mar 25/10/05
19	Aplicación de pruebas de sistema	mar 25/10/05	mié 26/10/05
6	Reunión	mié 26/10/05	mié 26/10/05
20	Producción de manuales del sistema	mié 26/10/05	jue 27/10/05
23	Revisión entre colegas	vie 28/10/05	vie 28/10/05



Id	Tarea	Comienzo	Fin
12	Fase de Cierre1	lun 31/10/05	vie 04/11/05
7	Reunión	mié 02/11/05	mié 02/11/05
13	Primer Examen Parcial	jue 03/11/05	jue 03/11/05
22	Identificar lecciones aprendidasy propuestas	jue 03/11/05	jue 03/11/05
21	Evaluación personal y del equipo	vie 04/11/05	vie 04/11/05



Forma de Registro de defectos

Equipo: Human Soft	**Semana:** 5	**Ciclo:** 1

Producto: Glosario de términos	**Fecha:** 25 de Septiembre del 2005
Descripción de los defectos:	
Existe ambigüedad en términos definidos por el cliente.	

Producto: Minuta	**Fecha:** 18 de Septiembre del 2005
Descripción de los defectos:	
Se encontraron faltas de ortografía en la minuta de la reunión semanal.	

Total de productos revisados:	**Total de defectos encontrados:**
2	4

4.4 Artículos de apoyo para la fase

El artículo recomendado para esta fase está titulado “Del negocio al sistema: El diagrama de actividad”, donde se desarrollan las características que ofrece el diagrama de actividad. Éste describe una actividad desde su inicio hasta que ésta concluye, que puede estar inmerso en un diagrama de carriles. Soporta condicionales y flujos paralelos, todo esto enmarcado en carriles que representan cada uno a una entidad distinta participante en el negocio. Cabe remarcar que el diagrama de actividades es una herramienta auxiliar para mejorar el diseño del sistema pero que no está considerado en los productos a generar en el ciclo.

Orozco, Sergio.
“*Del negocio al sistema: El diagrama de actividad*”, en: Software Guru. 02. 04 México D.F. 2006. pp. 46-47

CAPITULO 5 Especificación de los Requerimientos

5.1 Fase de Especificación de los Requerimientos.

En esta fase el equipo tendrá como principal objetivo entender las necesidades del cliente, para esto es necesario introducirse en el ambiente donde éste se desempeña para así identificar con mayor facilidad los requerimientos. El equipo de trabajo señala los requerimientos técnicos del producto que el cliente podría pasar por alto. El éxito de esta fase estará determinado por qué tan buena es la comunicación entre el cliente y el equipo de trabajo y la habilidad del equipo para aislar las necesidades reales del sistema.

5.2 Productos generados

En esta fase se generan varios documentos que tienen que ver con las necesidades del individuo que requiere el sistema, el *Texto con la definición del problema* es un documento en el que el cliente define las características que quiere que el sistema una vez terminado solvente. El *Glosario de términos* se realizará en conjunción con el cliente para unificar el lenguaje que maneja éste con el que maneja el equipo de trabajo.

En cuanto a los *Requerimientos no funcionales* y la *Especificación de Requerimientos del software* tendrán que realizarse de acuerdo a las necesidades y peticiones del cliente, siendo responsabilidad del equipo aislar de manera correcta los aspectos que el sistema necesita para posteriormente diseñarlos.

En cuanto el equipo logre aislar las necesidades reales del sistema podrá plasmarlas en el *Diagrama general de casos de uso* para que más adelante pueda ahondar en cada uno de los casos de uso detectados y pueda generar más adelante el *Detalle de los casos de uso*.

La mejor manera de que el cliente esté convencido del avance en el entendimiento de sus requerimientos por parte del equipo y viceversa, es un ejemplo visual, por ello se propone la creación del *Prototipo de la interfaz del usuario* en esta fase, este prototipo permitirá que el cliente proponga modificaciones o aclare puntos opacos antes de que éstos sean implementados.

Por último se generará el *Plan de pruebas del sistema* en el que se diseñan los casos de prueba a los que será sometido el sistema una vez que se implemente.

5.2.1 Texto con la definición del problema

Este documento es proporcionado por el cliente, en él se especificarán las necesidades que el cliente requiere que cubra el sistema, generalmente un listado facilita al equipo de trabajo identificar qué necesidades deben ser cubiertas en primera instancia, es decir, establecer una plataforma desde la cual se irán alcanzando los demás puntos especificados.

En este caso, la definición del problema será proporcionada por el instructor, que actuará como el cliente. En otro caso, es conveniente que se realicen entrevistas con el cliente, ya que en muchas ocasiones éste no tiene claro lo que realmente necesita.

El texto en donde se defina el problema debe incluir: quiénes serán los actores que interactuarán con el sistema y un listado con las características que se espera cumpla el sistema.

5.2.2 Glosario de términos

La función principal de un glosario de términos es unificar el lenguaje del cliente con el equipo de trabajo. En muchas ocasiones el equipo de trabajo enfrentará la realización de un proyecto que está contenido en un área del conocimiento muy distinta a la computación. Por ello es necesario para el equipo familiarizarse con el lenguaje del cliente para evitar malos entendidos y tener una comunicación constante y bien interpretada. Por otro lado, el cliente se podrá auxiliar del glosario para comprender los términos computacionales que ignore pero que tenga la necesidad de conocer.

Un buen glosario contendrá términos útiles para ambas partes, tanto cliente como equipo desarrollador, de preferencia debe ser breve y lo más conciso posible. Un glosario de cientos de palabras resultará incomodo y poco útil, lo que se busca es que cada parte sepa de qué se está hablando.

5.2.3 Diagrama general de casos de uso

Un caso de uso es una funcionalidad que es requerida en el sistema en la que, a través de una secuencia de pasos se llega a ejecutar en tiempo y forma lo requerido por el usuario. Cada caso de uso es representado por un óvalo con el nombre de la acción en infinitivo.

El diagrama general de casos de uso es un modelo en el cual se deben plasmar todas las posibles acciones que el sistema puede ejecutar a petición de cada actor. Dichas acciones o casos de uso, como ya se mencionó, se expresan en infinitivo, además por medio de flechas se indica qué actor puede realizarlas. Se recomienda no ahondar mucho en los detalles de los casos de uso, sino modelar las funciones del sistema de modo general. También en este módulo se debe indicar el alcance de lo que se implementará en cada ciclo de acuerdo con lo establecido en la *fase de estrategia*.

5.2.4 Detalle de los casos de uso

Este documento debe incluir en primera instancia al actor, quien es un individuo ajeno al sistema que dispara el inicio de un caso de uso. Otro aspecto que se incluye es la descripción del caso de uso, en donde se escribirán los pasos que sigue el actor y el sistema para realizar un caso de uso.

Las precondiciones de un caso de uso, son aquellas acciones que se dan por hecho en el instante inmediato anterior en el que el actor realice la acción, mientras que las poscondiciones son aquellos hechos que se esperan al concluir con la secuencia de pasos del caso de uso.

Es necesario hacer notar que existe la posibilidad de que el flujo del caso de uso no sea normal y que ocurran excepciones o un flujo esperado pero no deseado, a esto se le llamará flujo excepcional y tiene que ver con el manejo de errores externos durante la ejecución del caso de uso.

5.2.5 Prototipo de la interfaz del usuario

El prototipo de la interfaz del usuario es el esqueleto de la interfaz que se presentará a los usuarios. Es conveniente asociar cada pantalla con un caso de uso, esto permitirá ubicar y reconocer excepciones que se podrían presentar y cómo manejarlas, así como proponer una organización de los elementos de cada pantalla que permitan un uso más eficiente al usuario.

Se deberá tener especial cuidado en el diseño del prototipo, ya que obligatoriamente cada interfaz deberá contener los elementos necesarios para cubrir de manera satisfactoria el caso de uso al que representa.

Debiendo tener especial atención en las siguientes coincidencias: los nombres y acciones de las pantallas y botones diseñados en el prototipo con lo descrito en los casos de uso.

El prototipo es una herramienta muy útil para fomentar el diálogo respecto al proyecto con el cliente, ya que permite al cliente entender y poder ver el futuro funcionamiento de su sistema, ofreciéndole la oportunidad de sentirse parte activa en el proceso.

Se recomienda someter a pruebas el diseño de la interfaz, esto se puede lograr presentándola a usuarios distintos a los propios miembros del equipo, personas que no estén familiarizadas con el sistema y que se les solicite realizar alguna operación con éste, analizando qué tan intuitiva, cómoda, entendible, interactiva y sobre todo usable resultó ser la interfaz.

5.2.6 Requerimientos no funcionales

En este documento se establecerá lo que el cliente espera del funcionamiento del sistema, además de las necesidades que los desarrolladores deben poner a punto. Es recomendable basarse en la descripción dada por Moprosoft de los requerimientos no funcionales, entre los que destacan:

I*nterfaz con el usuario,* en este punto el cliente describirá las características para la usabilidad de su sistema, en las que puede incluir el tipo de ambiente gráfico que resulte más amigable, fácil de entender para el usuario final y que resuelva las necesidades del usuario de manera sencilla y eficiente.

Interfaz externa: aquí se describirá las posibles interacciones en las que el sistema pudiera relacionarse con otros sistemas.

Confiabilidad: especifica qué tan robusto y completo se requiere que sea el sistema a desarrollar.

Eficiencia: define la proporción del uso de los recursos, cuál de éstos se aprovechará a costa de los otros.

Mantenimiento: es necesario considerar que es probable que el mantenimiento y las mejoras al sistema no sean realizadas por el mismo equipo de trabajo. Por ello se deberá considerar el crear un sistema de fácil comprensión para que se pueda mantener.

Portabilidad: indica qué tan viable es que el sistema pueda desenvolverse en distintos ambientes.

Restricciones de diseño construcción: considera las restricciones que el cliente posee para el desarrollo de su sistema.

Legales y reglamentarias: estas restricciones son impuestas por agentes externos, como pueden ser políticas de la compañía del cliente, legislaciones vigentes en el lugar donde el sistema será utilizado y algunos otros aspectos sociales.

El tiempo de respuesta del sistema, los recursos de software y hardware de los que se dispone y hasta el nivel de seguridad que debe poseer el sistema.

Para obtener mejores resultados es conveniente que se analice cada punto del documento entre el cliente y los desarrolladores, y que se cuestionen ambas partes para que cada uno comprenda las necesidades reales del sistema.

5.2.7 Plan de pruebas del sistema

El plan de pruebas del sistema está enfocado en asegurar que se cumpla con cada uno de los requerimientos establecidos por el cliente, aquéllos que sean identificados por el equipo en el *texto con la definición del problema* y los que resultan triviales para los desarrolladores, como son entrar a una aplicación, salir del sistema, entre otros.

Cada caso de prueba está íntimamente ligado con un caso de uso, por lo que el formato del documento estará marcado por un caso de prueba ligado con un caso de uso, la descripción de los datos para la prueba y los resultados esperados, que verificaremos al realizar las pruebas del sistema de acuerdo a este plan al acabar de construirlo.

5.3 Ejemplos de productos generados en esta fase

A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Texto con la Definición del Problema.

Definición de necesidades para el sistema de software
Sistema para servicios médicos

El departamento de servicios médicos de la UNAM, requiere de un sistema de software que apoye al personal administrativo (clientes del sistema) en el registro de consultas y citas de pacientes (alumnado de la comunidad universitaria). Se espera que el sistema cumpla con las siguientes características:

1. El servicio es exclusivamente para alumnos y sólo para aquellos que estén inscritos en un semestre vigente.

2. La información de alumnos que se requiere es la siguiente: número de cuenta, nombre, fecha de nacimiento, semestre vigente y carrera, principalmente.

3. Se tendrán dos tipos de médicos: con especialidad y generales.

4. Los datos del médico que se requieren son: nombre, RFC, especialidad, consultorio asignado y turno en que labora.

5. Se asignará una cita para el alumno que la solicite de acuerdo a los siguientes criterios.

    A. Si el alumno la solicita por primera vez, el usuario dará de alta al alumno en el sistema. El usuario debe asignar cita con un médico general y no puede ser asignado directamente con algún médico especialista. El médico asignado será el primero que tenga un periodo disponible teniendo un margen de dos días para programar la cita considerando el turno para el cual es solicitada. Si dos médicos coinciden en espacios disponibles libres para programar la cita, el criterio para seleccionar al médico es el orden alfabético en el nombre del médico.

    B. Si el alumno ya tiene un registro previo en la unidad médica, se le asignará cita, pudiendo ser esta con médico general o especialista.

6. Las citas se registrarán indicando el nombre del médico, nombre del alumno que solicita atención, así como la fecha, hora, consultorio de atención y especialidad sí aplica.

7. Las citas se asignan en periodos de 30 minutos y se tienen dos días para programarla.

8. El sistema indicará los "espacios" disponibles que tiene cada médico y que pueden ser utilizados para asignar una cita.

9. Se tienen dos turnos de atención: matutino y vespertino, por lo que cada consultorio es compartido al menos por dos médicos.

10. Al momento de registrar un paciente, éste deberá comprobar su inscripción al semestre vigente.

11. El usuario podrá dar de alta y baja a un paciente del sistema, así como hacer cambios en los datos del mismo.

12. Se podrán saber cuáles son los pacientes (ordenados alfabéticamente) que serán atendidos a una hora y/o día específico.

13. Se podrá conocer el número de pacientes que serán atendidos por especialidad.

14. Se podrá cancelar una cita y dejar el periodo de tiempo o "espacio" libre para que esté disponible y pueda ser asignado nuevamente.
15. Todas las consultas se podrán imprimir.

16. El sistema deberá contar con una interfaz gráfica.

17. El sistema deberá estar codificado en Java cumpliendo con el estándar de codificación específico.

18. El sistema deberá operar en el ambiente de Linux y Windows.

Glosario de Términos

El glosario de términos es un diccionario en donde se definen los términos utilizados durante el desarrollo del sistema.

A

Administrador.- Persona encargada de realizar la administración del sistema de servicios médicos, lleva el control de las citas, médicos y pacientes.

Alumno.- Individuo que cuenta con un comprobante de inscripción a la institución educativa válido durante el semestre en curso.

C

Cita.- Acuerdo entre dos partes, un médico y un paciente, en el que se establece lugar, horario y motivo para reunirse en un futuro.

Contraseña.- Conjunto de caracteres que identifican a los usuarios del sistema como validos Es requerida para confirmar que la clave pertenece al miembro al que se desea acceder.

E

Especialidad.- Área especifica de la medicina que cuenta con médicos expertos en los elementos que esta abarca.

I

Ingresar al sistema.- Entrar al sistema de servicios médicos.

M

Médico.- Persona que cuenta con un título de Médico que avala su capacidad para desempeñarse en el sector de la salud pública.

P

Paciente.- Persona que externa algún padecimiento y solicita ser tratado por el personal médico de la institución.

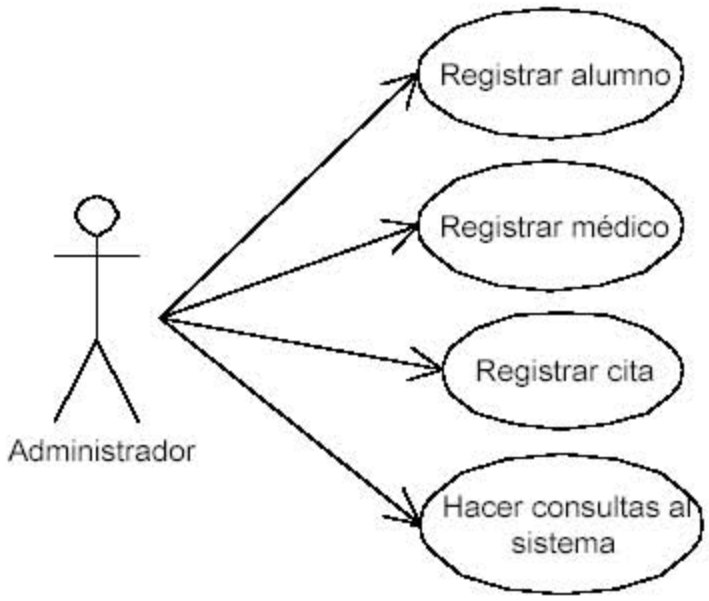
Diagrama General de Casos de Uso

Descripción del Sistema

Se desarrollará un sistema de administración eficiente para llevar un control sobre las citas del departamento de servicio médico en la UNAM.

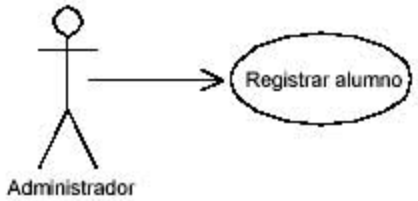
Diagrama

A continuación se presenta el diagrama general de uso del sistema. El sistema será manejado sólo por un usuario.



Detalle de los Casos de Uso

ID: N1
CASO DE USO: Registrar alumno.
ACTOR: Usuario
DESCRIPCION: Introducir, editar o borrar a un alumno del sistema.
PRECONDICION: Los alumnos deberán ser parte de los alumnos activos de la UNAM.



N1: Registrar Alumno

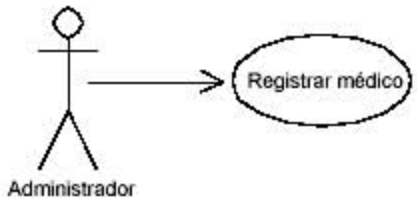
ACTOR		SISTEMA		
PASO	**ACCION**	**PASO**	**ACCION**	**EXCEPCION**
1	El usuario elige la opción de registrar un alumno.	-		-
2	El usuario autentifica al alumno mediante su número de cuenta.	3	El sistema verifica que el alumno este acreditado como alumno de la UNAM.	E1

N2: Excepciones

ID	NOMBRE	ACCION
E1	Alumno no acreditado como parte de la UNAM	El usuario no puede realizar ninguna operación con los datos del alumno.

POSCONDICION: El usuario puede realizar operaciones sobre los datos del alumno.

ID: N2
CASO DE USO: Registrar médico.
ACTOR: Usuario.
DESCRIPCIÓN: Registrar los datos de un médico al sistema.
PRECONDICIÓN: Que sea médico y que forme parte del equipo laboral de la UNAM.



N2: Registrar un médico

ACTOR		SISTEMA		
PASO	**ACCION**	**PASO**	**ACCION**	**EXCEPCION**
1	El usuario elige la opción dar de alta un médico.	2	El sistema muestra una interfaz dónde introducir el RFC, nombre, especialidad, consultorio y turno del médico (paso 2).	-
3	El usuario teclea el RFC, nombre, especialidad, consultorio y turno del médico	4	El sistema hace una búsqueda con los datos del médico (paso 5).	E2.1, E2.2
-	-	5	El sistema agrega los datos del médico.	-
6	El usuario acepta	7	El sistema regresa el menú anterior.	-

N2: Excepciones

ID	NOMBRE	ACCION
E2.1	El usuario dejo algún campo en blanco.	El sistema lanza un aviso de falta de datos.
E2.2	El médico ya esta dado de alta.	El sistema avisa que el médico ya esta dado de alta.

POSCONDICION: Los datos del médico han sido actualizados.

Prototipo de la Interfaz del Usuario

En las ventanas mostradas a continuación las palabras que se encuentran subrayadas representan ligas a otras pantallas.

Figura 1: Opciones del sistema.

Figura 2: Opciones del sistema para alumno.

Figura 3: Opciones del sistema para citas

Figura 4: Diferentes consultas a la base de datos del sistema

Figura 5: Número de pacientes por especialidad

Requerimientos No Funcionales

1. **Interfaz del usuario**
   Para que la interfaz con el usuario sea la mejor posible se pondrán claramente las opciones y lo que se puede realizar desde cualquier parte del sistema. Se espera que sea fácil de entender y cómoda para el usuario.

2. **Confiabilidad**
   Que no sucedan errores al definir las citas, que siempre se asigne una en un horario libre y con el médico correcto.

3. **Eficiencia**
   Se espera que el tiempo de respuesta sea favorable al realizar las consultas necesarias para cumplir con las necesidades del sistema.

4. **Mantenimiento**
   Se diseñará de tal manera que se puedan agregar funcionalidades a largo plazo.

5. **Portabilidad**
   El sistema podrá instalarse en los siguientes sistemas operativos: *Windows XP / Fedora Core 2*. Dado que se desarrollará en el lenguaje *J2EE 1.4 SDK and Sun Java System Application Server Platform Edition 8* se podrá obtener portabilidad gracias a las características del mismo.

Plan de Pruebas del Sistema

Objetivos de las Pruebas del Sistema

Las pruebas en un sistema son importantes ya que nos brindan una forma de poder verificar si nuestros productos cumplen con los requerimientos que el cliente solicita. Tener un plan de pruebas del sistema en etapas tempranas del desarrollo, cuando aun se están definiendo las necesidades del cliente, es importante, debido a que nos permite desarrollar las pruebas que se aplicarán al sistema antes de comenzar a programar el sistema en sí, dando la oportunidad a que se puedan definir de mejor forma las necesidades del cliente y además nos permite el ahorro de tiempo ya que al diseñar las pruebas tempranamente evitamos que en etapas subsecuentes las pruebas se ignoren o no se hagan debidamente.

Estrategia de las pruebas

Se definirán casos de prueba para cada caso de uso identificado, un caso de prueba con datos correctos y completos con los que se espera una salida satisfactoria, y otros casos de prueba que prueben cada uno de los diferentes tipos de excepciones registradas en el detalle de los casos de uso.

Casos de Prueba

Caso de uso	Entradas	Resultados Esperados
N1.1	No. cuenta: 300109133 Nombre: Miguel Ehécatl Morales Trujillo Fecha de nacimiento: 04-04-1984 Semestre Vigente: 2004-2 Carrera: Ciencias de la Computación	El sistema agrega los datos del alumno.
N1.1	No. cuenta: Nombre: Miguel Ehécatl Morales Trujillo Fecha de nacimiento: 04-04-1984 Semestre Vigente: 2004-2 Carrera:	El sistema lanza un aviso de falta de datos.
N1.1	No. cuenta: 300109133 (Número de cuenta de alguien ya registrado)	El sistema avisa que el alumno ya está dado de alta.
N1.2 (paso 3}	No. cuenta: 300109133 (Número de cuenta de alguien NO registrado)	El sistema avisa que el alumno no se encuentra.
N1.2 (paso 3)	No. cuenta: 300109133 (Número de cuenta de alguien registrado)	El sistema despliega una forma con campos editables.
N1.2 (paso 6)	No. cuenta: 300109133 Nombre: Fecha de nacimiento: Semestre Vigente: Carrera: Ciencias de la Computación	El sistema lanza un aviso de falta de datos.
N1.2 (paso 6)	No. cuenta: 300109133 Nombre: Miguel Ehécatl Morales Trujillo Fecha de nacimiento: 04-04-1984 Semestre Vigente: 2008-1 Carrera: Ciencias de la Computación	El sistema avisa que se realizó la actualización.
N1.3 (paso 3)	No. cuenta: 300109133 (Número de cuenta de alguien NO registrado)	El sistema avisa que el alumno no está registrado.
N1.3 (paso 3)	No. cuenta:	El sistema lanza un aviso de falta de datos.
N1.3 (paso 3)	No. cuenta: 300109133 (Número de cuenta de alguien registrado)	El sistema elimina el registro del alumno.

Caso de uso	Entradas	Resultados Esperados
N2	rfc: BAAAP 12351-65 Nombre: Benito Antonio Alcaraz Pérez Especialidad: Médico general Consultorio: 15 Turno: Matutino	El sistema agrega los datos del médico
N2.1	rfc: Nombre: Especialidad: Médico general Consultorio: 6 Turno: Matutino	El sistema lanza un aviso de falta de datos
N2.2	rfc: BAAAP 12351-65 (rfc de alguien ya registrado)	El sistema avisa que el médico ya está dado de alta

5.4 Artículos de apoyo para la fase

En esta fase se recomienda la lectura de varios artículos, entre los cuales destaca "Patrones de casos de uso" ya que será de gran ayuda para la elaboración del diagrama general de casos de uso y para el detalle de los casos de uso. En el artículo se presentan las ventajas de generar los casos de uso así como la descripción y elementos de éstos. Remarca la necesidad de modelar el sistema identificando los requerimientos y cubriéndolos con casos de uso para su posterior implementación. Incluye un ejemplo bien desarrollado de un caso de uso.

Como apoyo para la especificación de los requerimientos es conveniente la lectura del artículo "El ABC de un taller de requerimientos", en el que se propone convocar a una sesión con el cliente, los usuarios principales del sistema y los miembros del equipo. Para llevar a cabo dicha reunión y sacarle el mayor provecho es necesario que el equipo genere una agenda de dicha reunión, en la cual se marcará un límite de tiempo y de participantes. Se presenta un listado de material de apoyo recomendado, por último registrar los resultados obtenidos a lo largo de la sesión.

Cuesta R., Saúl.
"*Patrones de casos de uso*", en: Software Guru. 01. 06
México D.F. 2005. pp.40-42

Alegría V., Mónica.
"*El ABC de un taller de requerimientos*", en: Software Guru. 02. 03
México D.F. 2006. pp.34-35

CAPITULO 6 Diseño

6.1 Fase de Diseño

Durante la fase de diseño se establecerán las relaciones existentes entre los componentes que se identificaron durante la *fase de especificación de los requerimientos*. Se definirá la arquitectura del sistema a desarrollar. La base de la estructura funcional del sistema quedará establecida en esta fase, aunque se irá modificando a lo largo del ciclo es aquí donde quedan formalmente establecidos los cimientos del sistema.

6.2 Productos generados

En esta fase se especificará el *ambiente de implementación* y un *estándar de diseño* para unificar el trabajo del equipo referente a la construcción del sistema. Se generarán *diagramas de distribución* de los componentes, *de clases* y *de paquetes* donde se agruparán los componentes de acuerdo a su función en la arquitectura del sistema.

Se generará un *plan de integración* que como su nombre lo indica, marcará el orden en que los componentes se irán amalgamando para formar uno solo.

Los *diagramas de secuencia* que representarán en el tiempo la descripción de cada caso de uso, así como el *diagrama de estado* que indicará el estado en el que puede encontrarse un usuario mientras utiliza el sistema al realizar alguna funcionalidad.

6.2.1 Especificación del ambiente de implementación

El documento especifica el ambiente de implementación y las herramientas de las que se auxiliará el equipo para la programación, en los que se debe incluir: el compilador a utilizar y su versión, las herramientas de diagramación para UML y su versión, de ser necesario el sistema manejador de base de datos, el servidor de aplicaciones y sus versiones, las herramientas para automatizar pruebas unitarias y las herramientas para realizar las conexiones a la base de datos. Nótese que gran parte de esta información se pudo obtener durante la especificación de los requerimientos.

Este documento será de mucha utilidad para el *Administrador de Apoyo* ya que una vez que el equipo haya acordado los recursos necesarios para la implementación, el *Administrador de Apoyo* será el encargado de conseguir el software necesario, instalarlo en las máquinas donde el equipo planea trabajar y además resolver dudas sobre el uso de cada una de las herramientas a los integrantes del equipo.

6.2.2 Estándar de diseño

El estándar del diseño es un documento para unificar, entre los miembros del equipo, varios aspectos entre los que se incluyen:

- La convención para el nombrado de los componentes: entre los componentes se consideran: clases, variables, métodos, interfaces entre otros. Existen ya

convenciones utilizadas para nombrar componentes que resultaría muy conveniente seguir para incluir al sistema dentro de un estándar global. La principal ventaja de seguir un estándar es que a los miembros del equipo les resultará fácil de identificar cada uno de los componentes que se vayan generando a lo largo del proceso.

- Los formatos para las interfaces de los componentes, incluyen el formato de los parámetros y los códigos de error.
- Los mensajes de error que mandará el sistema: es necesario establecer cómo se manejarán los errores que ocurran durante la ejecución, establecer el formato de los mensajes de error así como la manera en que serán presentados al usuario. Debe tomarse en cuenta que el usuario no está obligado a ser un experto en computación por lo que el mensaje que se desplegará, de ser el caso, deberá ser claro para que cualquiera pueda tomar una decisión de cómo solucionar el problema.

6.2.3 Arquitectura con diagrama de paquetes

Los paquetes serán utilizados para agrupar en subconjuntos los diversos elementos del sistema. Cada componente que se identifique será agrupado en alguno de los siguientes subconjuntos:

- Paquete de interfaz de usuario: aquí se incluyen archivos cuya función es presentarse al usuario una pantalla, la interfaz de comunicación hombre-máquina, pueden tener como extensión html o jsp.
- Paquete de lógica de la aplicación: en este paquete se incluirán las clases que están encargadas de dar funcionalidad al sistema, las clases de control. Estas clases estarán en constante comunicación con las de los otros dos paquetes.
- Paquete de almacenamiento: en este subconjunto se agrupan las clases que están encargadas de la generación, manejo y mantenimiento de la base de datos del sistema.

Cada paquete dentro de este documento es representado por un ícono con la forma de un ‘folder’ en cuya pestaña indicaremos el nombre del paquete. Es posible que un paquete contenga subpaquetes, las relaciones entre los paquetes estarán representadas por flechas punteadas que indican la dependencia. Para un mejor entendimiento del documento se recomienda establecer una jerarquía entre los paquetes.

6.2.4 Diagrama de distribución

En este diagrama se definirá qué se instalará en qué máquina y cómo estarán intercomunicadas, en caso de que el sistema sea distribuido, es decir, que los componentes no se encuentren en sólo una máquina.

Cada máquina se representa con un cubo y una etiqueta que indicará los componentes a incluir en esa máquina, mientras que las conexiones en el diagrama representarán una conexión física real entre las máquinas. Conviene indicar el mecanismo de comunicación que utilizarán dichos componentes interconectados, es decir su protocolo de comunicación.

6.2.5 Diagramas de clases

Este documento contendrá los diagramas que describen el comportamiento estático del sistema y los tipos de relaciones existentes entre las clases. Se generará un diagrama de clases por cada paquete que contenga el sistema, en este caso se generarán tres: uno para la capa interfaz de usuario, la capa de almacenamiento y la capa de control.

Las clases serán representadas por un rectángulo con tres partes, la primera parte contendrá el nombre de la clase, la siguiente los atributos y la tercera para los métodos contenidos en la clase.

Las relaciones son de tres tipos: de asociación, de agregación y de generalización.

Las relaciones del tipo asociación son aquéllas que ligan a objetos de clases que recomunican entre sí. Este tipo de relación es representado por una línea que conecta a las clases asociadas.

Las relaciones de agregación son un tipo de asociación que denota que los objetos de una clase forman parte de otra clase más general. Se denota por una línea con un rombo del lado de la clase compuesta.

Las relaciones de generalización relacionan a una clase general que comparte sus atributos y operaciones con clases más particulares, es decir, clases que se encargan de operaciones en particular. Se denota por línea con un triángulo del lado de la general.

Para generar el documento, es conveniente como primer paso identificar qué clases será necesario crear para que el sistema cumpla con los requerimientos establecidos, esto se realizará revisando cada caso de uso identificando qué necesidades cubre cada uno de éstos y con ello decidir qué clases cumplirán con ese trabajo. A lo largo de esta revisión el equipo identificará acciones específicas que tienen que ser realizadas por una clase de tipo interfaz, control o de datos.

6.2.6 Diagramas de secuencia

Cada caso de uso deberá ser representado con un diagrama de secuencia, éstos muestran cómo se comportan los objetos entre ellos a través del tiempo. Representarán el comportamiento dinámico de los casos de uso.

Los objetos son representados como un rectángulo con el nombre del objeto en la parte superior, del que se desprende una línea vertical que representa a dicho objeto a través del tiempo, avanzando de arriba hacia abajo. A dichos objetos se les puede solicitar realizar un método en algún instante. El actor es el encargado de invocar los métodos, dichas invocaciones son representadas con líneas sólidas desde el objeto origen hasta señalar al objeto destino, sobre dicha flecha se indicará el nombre del método con o sin los parámetros requeridos.

Deberá tomarse en cuenta que por cada caso de uso deben generarse diagramas de secuencia ejemplificando los flujos normales y aquellos que manejan excepciones, flujos excepcionales, además se deberá identificar cada una de las clases participantes en el caso de uso para que se representen en el diagrama de secuencia.

6.2.7 Diagrama de estado

Este diagrama resulta ser una gráfica dirigida con un estado inicial y varios posibles estados finales, con las respectivas transiciones entre ellos, cada transición va de un estado origen a sólo un estado destino. El diagrama de estados puede ser visto como la forma gráfica de un autómata finito determinístico.

El diagrama de estados, indicará cómo navega el usuario a través de las interfaces del sistema. Cada diagrama tendrá un estado inicial que se identificará con una circunferencia negra que rodea un punto negro, los estados serán representados por un óvalo con el nombre de la interfaz en la que se encuentra el usuario al realizar alguna acción, las acciones estarán determinadas en el documento por flechas con un estado origen, una acción y el estado destino.

Es necesario indicar aquellos eventos que generarían un error y manejarlos de acuerdo a sus necesidades, por ejemplo enviando al usuario a un estado donde pueda recuperarse para continuar sin necesidad de reiniciar todo el proceso.

6.2.8 Plan de pruebas de integración

Este documento se puede considerar como el instructivo de ensamblado para el sistema, en él se establece el orden de los componentes que se irán integrando. La recomendación es tomar en cuenta la dependencia estática, es decir, identificar qué paquetes dependen de otros para funcionar correctamente.

Aquellos paquetes que no dependan de otros serán los primeros en ser integrados, posteriormente se integraran aquéllos que dependen totalmente de los paquetes que se integraron en el nivel anterior. Para lograr una buena integración es posible auxiliarse de un diagrama en el que se marquen las dependencias de los paquetes mediante flechas, colocando en el nivel más alto aquéllos que no necesitan de otros para su correcto funcionamiento e ir descendiendo de acuerdo a la dependencia de cada uno con respecto a los del nivel superior.

Posteriormente ya construido el diagrama de dependencias, se colocará la información en una tabla donde se indicará el componente a integrar, los componentes de los que depende y los parámetros a probar. Nótese que al menos el primer componente a integrar no depende de ningún otro y que los siguientes componentes, si dependen de algún otro componente sólo podrá ser de aquéllos que ya están integrados, esto se asegura por el hecho de que nuestro diagrama de dependencias no contiene ciclos.

Finalmente el listado obtenido en esta tabla es el orden de integración de los componentes.

6.3 Ejemplos de productos generados en esta fase

A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Especificación del Ambiente de Implementación

Compilador:

- Java de Sun v1.4.1

Herramientas de diagramación para UML:

- *Argo UML*
- *Dia v0.24* (o cualquier versión)

Manejador de base de datos:

- *PostgreSQL v8.1.4-1*

Servidor de aplicación:

- *TomCat v5.20*

Herramientas para automatizar pruebas unitarias:

- *JUnit v1.1*

Herramienta para la conexión con la base de datos:

- *JDBC*

Estándar de Diseño

Convención para el nombrado de componentes

Nuestro estándar se basará en gran parte del estándar de SUN para java, considerando como más importante lo siguiente:

Clases:

Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas. Debemos intentar mantener los nombres de clases simples y descriptivos. Debemos usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como URL o HTML).

Variables:

Las variables, tanto de ejemplar, de clase, como las constantes de clase se escriben en mayúsculas y minúsculas y con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas. Los nombres de variables no deben empezar con los caracteres subrayado "_" o dollar "$", incluso aunque estén permitidos.

Los nombres de variables deberán ser cortos y llenos de significado. La elección de una variable debería ser mnemónica-es decir, diseñada para indicar al observador casual su utilización. Se deben evitar los nombres de variable de un sólo caracter, excepto para variables temporales.

Los nombres de variables constantes de clases y las constantes ANSI deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("_"). (Se deberían evitar las constantes ANSI para facilitar la depuración.)

Métodos:

Los métodos deberán ser verbos, en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas.

Interfaces:

Los nombres de interfaces se tratan igual que los nombres de clases.

Declaraciones de Clases e Interfaces

Cuando codificamos clases e interfaces Java, se deben seguir las siguientes reglas de formateo:

- No debe haber ningún espacio entre el nombre y el paréntesis "(" que inicia la lista de parámetros.
- El corchete abierto "{" aparece en la misma línea que la declaración.
- El corchete cerrado "}" empieza una línea por sí mismo, indentado a su correspondiente sentencia de apertura, excepto cuando es una sentencia null en la que el "}" debería aparecer inmediatamente después del "{":

```
class Sample extends Object {
        int ivar1;
        int ivar2;

        Sample(int i, int j) {
                ivar1 = i;
                ivar2 = j;
        }

        int emptyMethod() {}
        ...
}
```

- Los métodos están separados por una línea en blanco.

Formatos para las interfaces de los componentes

Los parámetros de las variables serán de la siguiente manera:

- • Variables del tipo entero: Campos del formulario que sean caracteres numéricos, es decir, por ejemplo teléfono, número de cuenta, fecha, contraseña, etc.
- • Variables del tipo string: Campos del formulario que sean caracteres no numéricos, es decir por ejemplo nombre, dirección, etc.
- • Variables del tipo booleano: Las variables de este tipo solo se usan para métodos que determinan un predicado.

Los errores serán tratados con excepciones comunes de JAVA.

Si algún campo no está especificado o no es sencillo determinar el tipo al que pertenece de las anteriores, se determinará entre todos los miembros del equipo.

Mensajes de error que mandará el sistema

Los mensajes de error del sistema se manejaran de tipo string, siendo una frase o palabra descriptiva, que indique al usuario que tipo de error se produjo, cómo solucionarlo y cómo deberá reaccionar el sistema.

En ninguna circunstancia la frase puede ser en sentido ofensivo, ya sea hacia el usuario, el sistema o el desarrollador. Tampoco se admiten frases muy técnicas, es decir, que necesiten un amplio conocimiento del sistema. Estas frases deben ser lo mas básicas posibles para que cualquier persona(usuario, desarrollador, técnico) pueda determinar que hacer en caso de este error.

Si existe un error al cual no se pueda determinar una frase con las características anteriores, se determinará entre todos los integrantes del equipo una frase apropiada.

Arquitectura con diagrama de paquetes

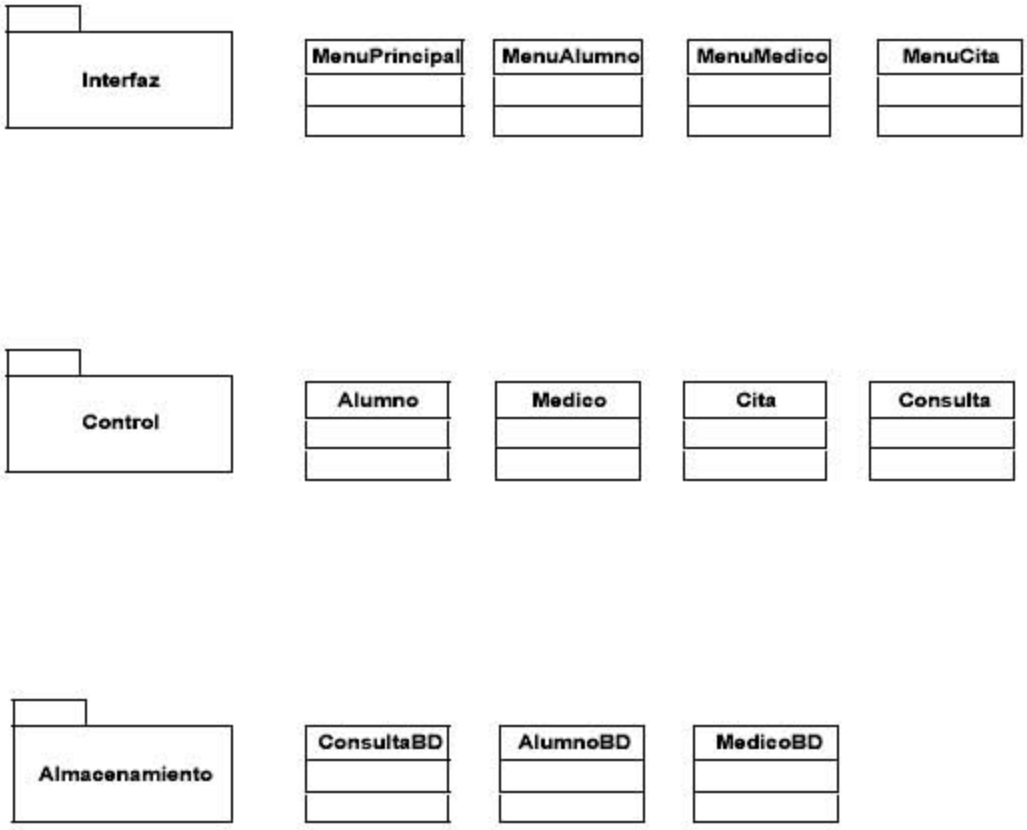


Diagrama de Distribución

A continuación se muestra la distribución de los nodos (servidores y clientes), así como los componentes de cada nodo del sistema de servicios médicos.

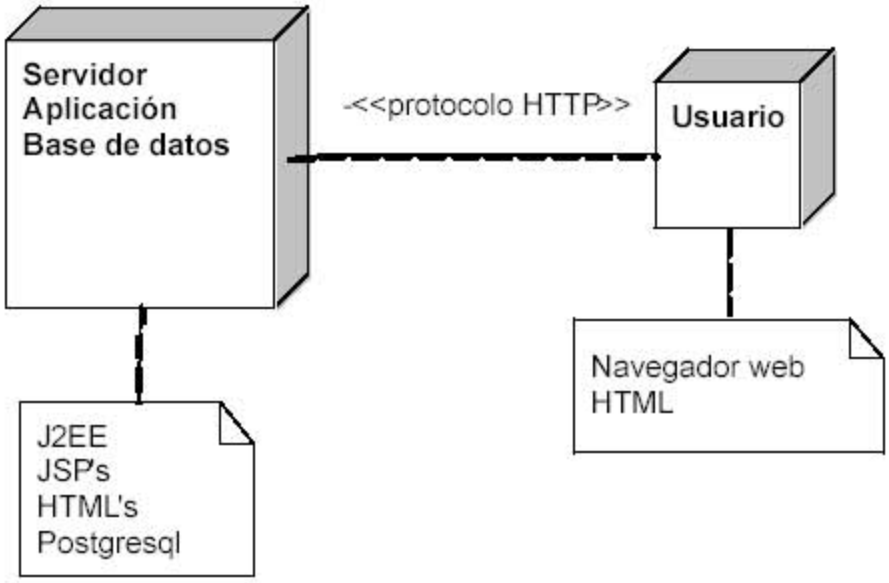
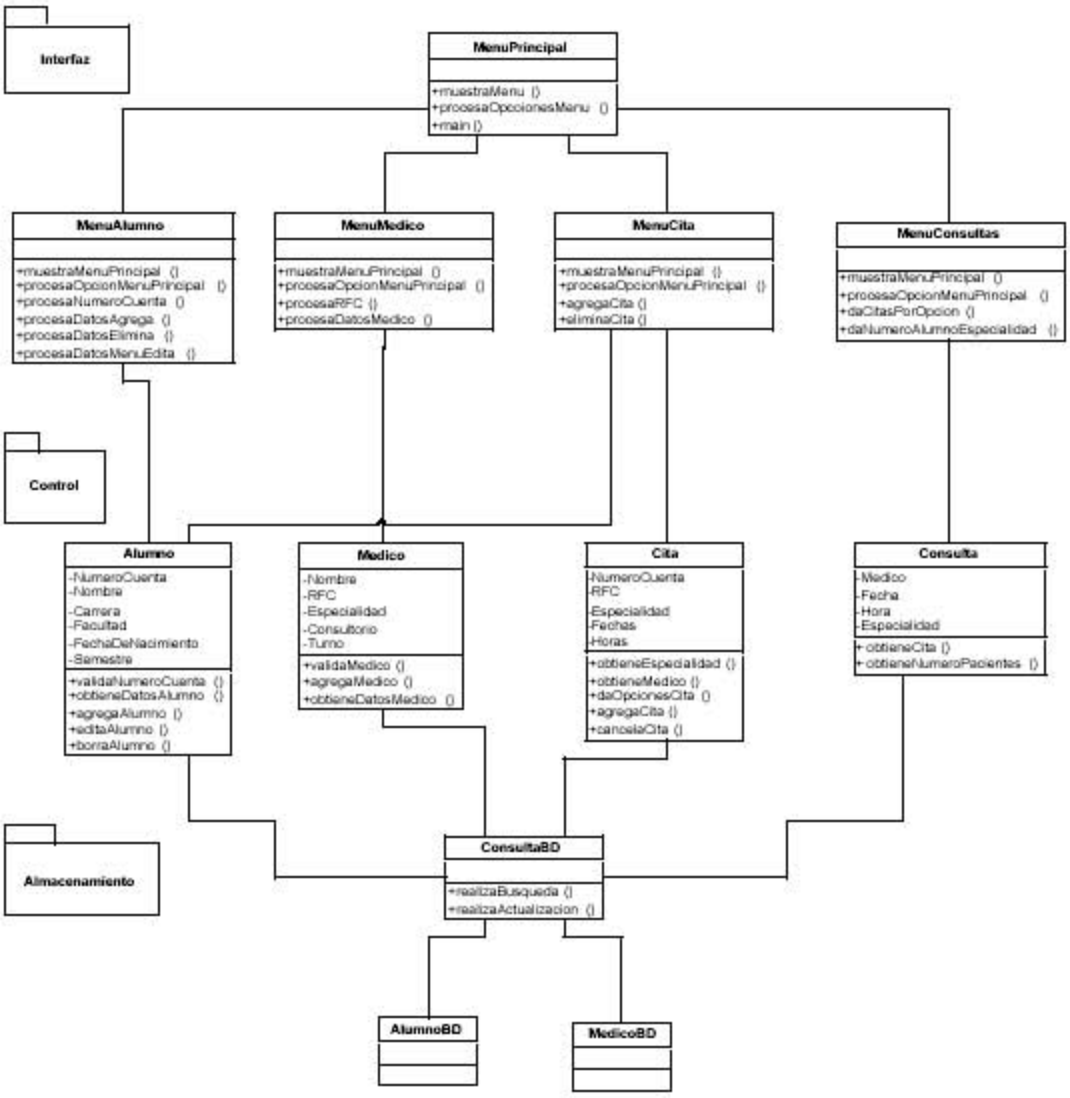


Diagrama de Clases



Diagramas de Secuencia

N2: Registrar un Médico

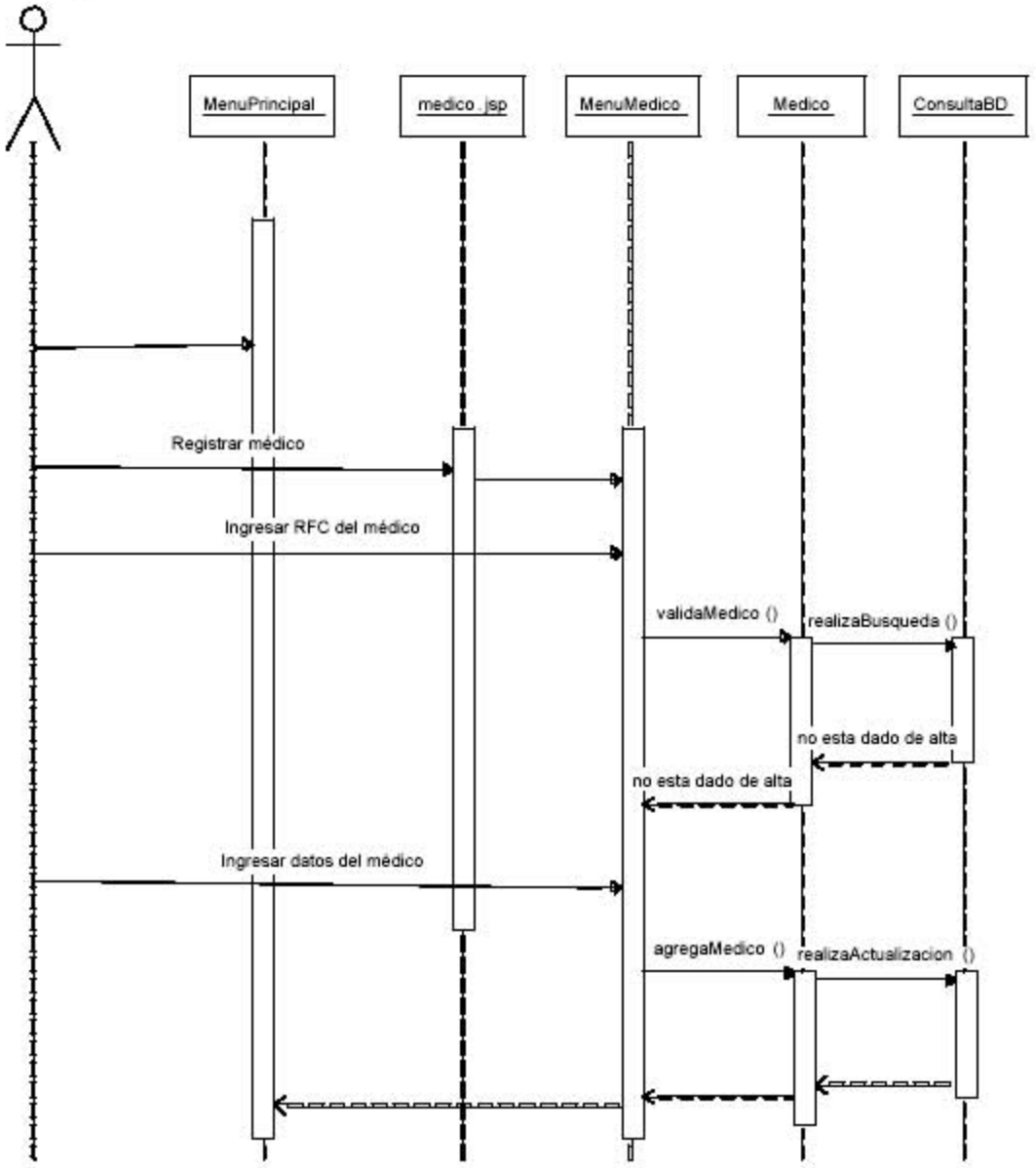
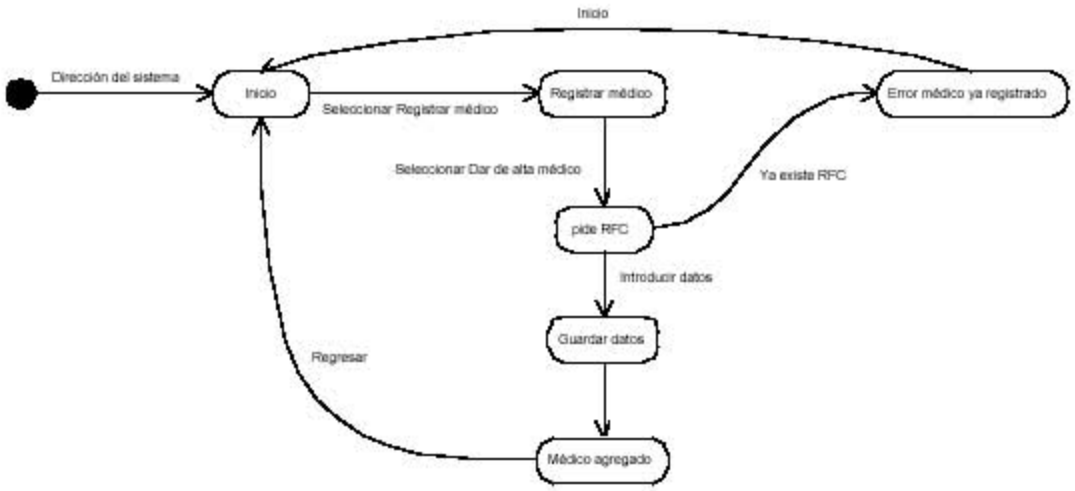


Diagrama de Estados

Diagrama de estados correspondiente al caso de uso Registrar médico



Plan de Pruebas de Integración

La estrategia a seguir para el plan de pruebas de integración será la de Dependencia de paquetes, la cual estará basada en la dependencia definida según en el diagrama de paquetes.

El orden de las clases que se implementarán es:

1. Alumno.java, Medico.java, Cita.java y Consulta.java, pudiendo implementar estas cuatro clases en paralelo.

2. MenuAlumno.java, MenuMedico.java, MenuCita.java y MenuConsulta.java, pudiendo implementar estas cuatro clases en paralelo.

3. MenuPrincipal.java

6.4 Artículos de apoyo para la fase

Para esta fase se recomienda la lectura de varios artículos, la mayoría sobre diagramas UML, el primero de ellos titulado “Modelo conceptual” en el que el autor propone una manera de representar el problema utilizando diagramas de clases, dividiendo el proceso en dos secciones, primeramente el análisis del problema y luego el diseño de lo que se abstrajo. Define las características de un diagrama de clases, en las que incluye la entidad o clase, los atributos y acciones de éstas y los distintos tipos de relaciones existentes entre clases con sus respectivas multiplicidades. Resulta una buena introducción al modelado del problema por medio de diagramas de clases.

Por otro lado como introducción para la elaboración de los diagramas de estado, se puede recurrir al artículo “La vida de un objeto” donde se describen las características y componentes de dichos diagramas, comenzando con aquellas unidades que pueden ser consideradas objetos. Los distintos estados en los que se puede encontrar un objeto así como las transiciones entre ellos y las acciones que les dan lugar. Tómese en cuenta que los diagramas de estado resultantes modelarán al estado del actor en el sistema, ya que el objeto es el actor.

Como apoyo para generar el diagrama de paquetes el artículo “Divide y vencerás” presenta las ventajas de poder agrupar los elementos del sistema en paquetes, ya sean de casos de uso o de clases, para mantener una organización efectiva y poder facilitar la representación lógica del sistema, además de enunciar distintos criterios para definir los paquetes en los que se agrupará. En el caso del diagrama de paquetes del curso la agrupación en paquetes está establecida de acuerdo a la capa a la que pertenece cada clase.

El artículo “Usabilidad” describe concretamente este atributo de un sistema, como es que desde la fase de diseño del sistema se debe ponderar qué tan útil será el sistema terminado para el usuario final. Con el término usabilidad se refiere a qué tan fácil de usar es, qué tan eficiente y sobre todo que el usuario quede satisfecho con el sistema que usa. Seguir los pasos propuestos en el artículo acercará más al sistema a ser útil y sobre todo el equipo de trabajo se dará cuenta en una fase temprana del proceso qué tan usable es su sistema, para así poder mejorar cualquier componente.

Orozco, Sergio.
“*Modelo conceptual*”, en: Software Guru. 01. 06
México D.F. 2005. pp. 44-45

Orozco, Sergio.
“*La vida de un objeto*”, en: Software Guru. 02. 03
México D.F. 2006. pp. 44-45

Orozco, Sergio.
“*Divide y vencerás*”, en: Software Guru. 02. 05
México D.F. 2006. pp. 42-43

Esparza, Alfonso.
“*Usabilidad*”, en: Software Guru. 02. 05
México D.F. 2006. p.46

CAPITULO 7 Construcción

7.1 Fase de Construcción.

El objetivo de esta fase es el de implementar el software, la implementación abarca pasos de programación y la realización de pruebas al software. Se profundizará en la especificación de la implementación del sistema, detallando los diagramas de clases, previamente generados, para su implementación, así como la generación del plan de pruebas.

7.2 Productos generados

En esta fase se describen a mayor profundidad las capas del sistema y se declaran los requerimientos necesarios para que el sistema funcione correctamente. Se realizará el detallado a los diagramas de clases, el *Código para las clases* y el *Plan de pruebas unitarias* que detalla las pruebas que se realizarán al código para verificar su correcto funcionamiento.

7.2.1 Diagramas de clases detallados

El detallado o refinamiento de los diagramas de clases se deja para esta fase ya que es cuando se comenzará a implementar el código para las clases y resulta necesario establecer aspectos como son los tipos de datos de entrada y salida de cada uno de los métodos de las clases en la capa de control o el tipo de cada uno de los atributos de cada entidad de la capa de almacenamiento.

El nivel de detalle de cada diagrama debe ser establecido por el equipo como un estándar.

7.2.2 Código para las clases

El grueso de la producción de esta fase es el código para las clases, dependiendo del lenguaje de programación señalado en la especificación de la implementación el equipo establecerá un estándar para los archivos que contengan el código, pudiendo ser éste el propuesto por en el estándar del lenguaje o alguno ideado por los miembros del equipo.

Dicho formato deberá facilitar la identificación de las clases mediante la forma de nombrarlas y la distribución del código dentro del archivo.

7.2.3 Plan de pruebas unitarias

Como se ha mencionado en esta fase se busca probar el buen desempeño del sistema. En el plan de pruebas unitarias existen dos tipos de evaluación, las pruebas de caja blanca que corresponden a aquéllas que obligan la ejecución de cada una de las instrucciones que fueron codificadas en cada una de las clases. Para determinar cuántos casos de prueba son necesarios es conveniente señalar todas las posibles trayectorias que un actor determinado tenga opción de seguir durante la ejecución del programa, tomando en cuenta los operadores condicionales que se encuentren en el código.

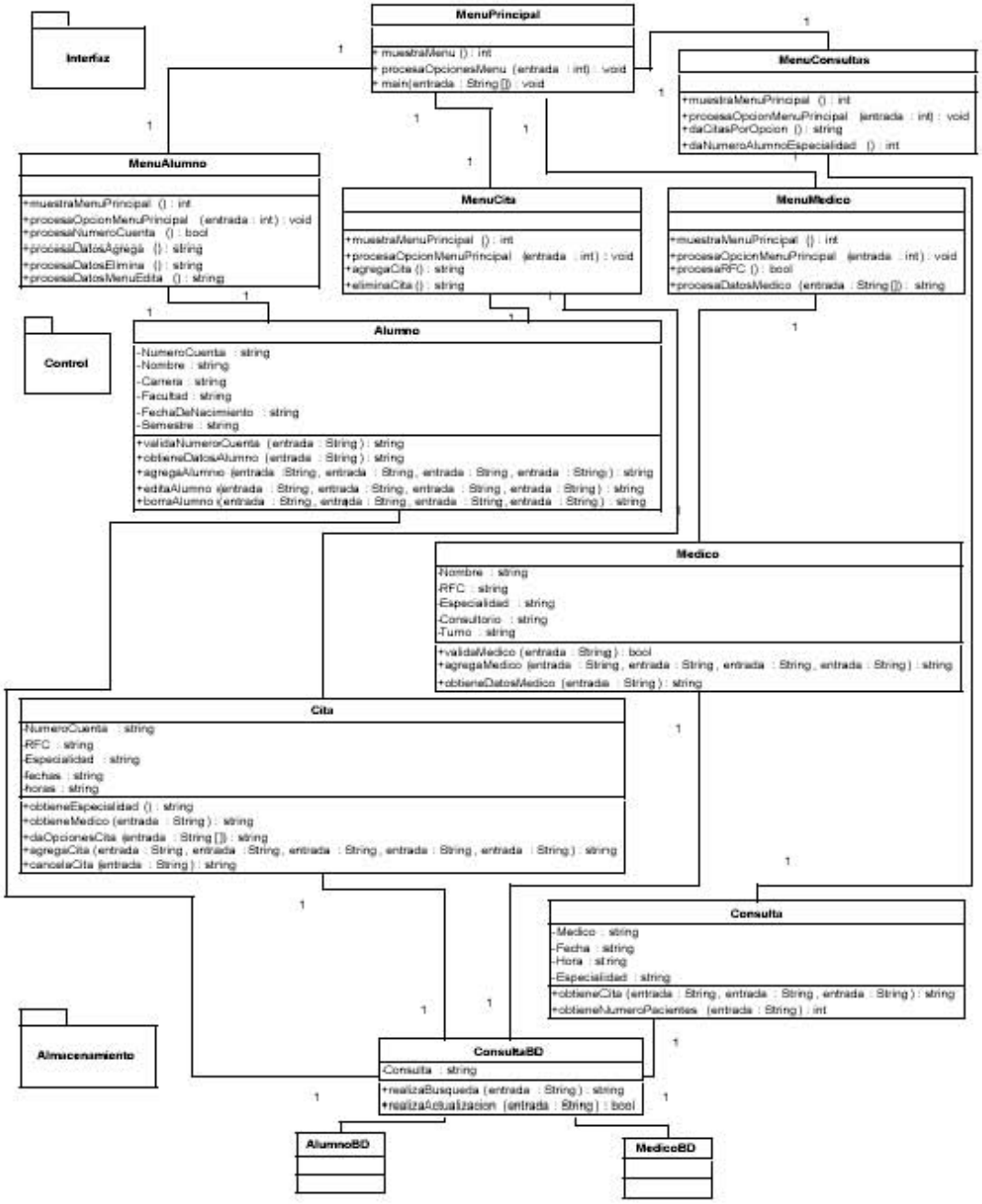
Por otro lado, las pruebas de caja negra, tomarán como punto de partida los posibles valores de entrada asignados por el usuario y los valores esperados de salida del sistema.

El responsable de dirigir este plan de pruebas es el *Ingeniero de Desarrollo*, para el código que genere realizará las pruebas y reportará los defectos encontrados para corregirlos.

7.3 Ejemplos de productos generados en esta fase

A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Diagramas de Clases Detallados



Código para las Clases

```
/*
 * Alumno.java
 * Clase que tiene el control del Alumno.
 * @version 1.1   30 de octubre de 2006.
 * @autor  Javonez
 * Copyright (c) 2004 Javonez, Inc. Todos los derechos reservados.
 */

package javonario.control;

import javonario.baseDatos.*;
import javonario.interfaz.*;

import java.sql.*;
import java.io.*;
import java.util.Vector;

public class Alumno {

   /**
    * Metodo para verificar si un alumno se encuentra dentro de la base de datos del sistema.
    * @param no_cuenta el numero de cuenta de algun alumno
    * @return  verdadero si el alumno cuyo numero de cuenta es no_cuenta,
    * se encuentra en el sistema, falso en otro caso
    */
   public static boolean validaNumeroCuenta(String no_cuenta) {
            if (ConsultaBD.realizaConsulta
               ("SELECT no_cuenta  FROM alumno WHERE no_cuenta = '"+no_cuenta+"';").size() == 0 )
               return false;
            return true;
   }

   /**
    * Metodo que regresa el atributo del alumno con un numero
    * de cuenta dado
    * @param atributo a buscar
    * @param no_cuenta numero de cuenta de un alumno
    * @return atributo encontrado
    */
   public static String obtieneAtributo (String atributo, String no_cuenta) {
            return  (String)ConsultaBD.realizaConsulta
               (" SELECT " + atributo +
                " FROM   alumno "+
                " WHERE  no_cuenta = '"+no_cuenta+"';").get(0);
   }

   /**
   * Metodo que elimina del sistema al alumno cuyo numero de cuenta es no_cuenta
   * @param no_cuenta
   * @return cadena vacia como signo de que la actualizacion fue hecha
   */
   public static String borraAlumno (String no_cuenta) {
            try {
               ConsultaBD.realizaActualizacion(" DELETE FROM cita "+
                                                   " WHERE no_cuenta = '"+no_cuenta+"'; ");
               ConsultaBD.realizaActualizacion(" DELETE FROM alumno "+
                                                   " WHERE no_cuenta = '"+no_cuenta+"'; ");
               return "";
            }catch (SQLException e) {
               return null;
            }
   }
```

```
/**
 * Metodo que agrega un alumno al sistema a partir de todos sus datos
 * @param no_cuenta
 * @param nombre_alumno
 * @param carrera
 * @param nacimiento
 * @param semestre
 * @param numeroCita
 * @return los datos del alumno que fue agregado al sistema
 */
public static String agregaAlumno (String no_cuenta, String nombre_alumno, String carrera,
                                   String nacimiento, String semestre, String numeroCita) {
        try {
           ConsultaBD.realizaActualizacion("INSERT INTO alumno VALUES ("+
                                           ""+no_cuenta   +","+
                                           ""+nombre_alumno+","+
                                           ""+carrera     +","+
                                           ""+nacimiento  +","+
                                           ""+semestre    +","+
                                           ""+numeroCita  +");");
           return "";
        }catch (SQLException e) {
           return null;
        }
}

/**
 * Metodo que regresa todos los atributos de una alumno
 * en formato html (tabla html)
 * @param no_cuenta numero del cuenta de un alumno
 * @return cadena en formato html
 */
public static String obtieneDatosAlumno (String no_cuenta) {
        Vector consulta = ConsultaBD.realizaConsulta // vector donde guardamos al alumno
           (" SELECT no_cuenta, nombre_alumno, carrera, nacimiento, semestre, numerocita "+
            " FROM   alumno "+
            " WHERE  no_cuenta = '"+no_cuenta+"';");

        if ( consulta.size() == 0 || consulta == null )
           return  null;
        else {
           String tabla = "<table width=\"100%\"  border=\"1\" "+
                  "cellspacing=\"0\" cellpadding=\"0\">\n";

           consulta.add( 0,  MenuCita.tag("b","class=\"titulo\"","semestre" ));
           consulta.add( 0,  MenuCita.tag("b","class=\"titulo\"","fecha de nacimiento" ));
           consulta.add( 0,  MenuCita.tag("b","class=\"titulo\"","carrera" ));
           consulta.add( 0,  MenuCita.tag("b","class=\"titulo\"","nombre de alumno" ));
           consulta.add( 0,  MenuCita.tag("b","class=\"titulo\"","numero de cuenta" ));

           int i = 0;
           for (int k = 0 ; k < consulta.size()-1; k++){
                  if ( i == 0 )
                     tabla = tabla+"<tr";
                  tabla = tabla+"<td><center>"+
                     consulta.get(k)+"</center></td>";
                  if ( i == 4) {
                     tabla = tabla+"</tr>";
                     i = -1;
                  }
                  i++;
           }
           tabla = tabla+"</table>";
           return tabla;
        }
}
```

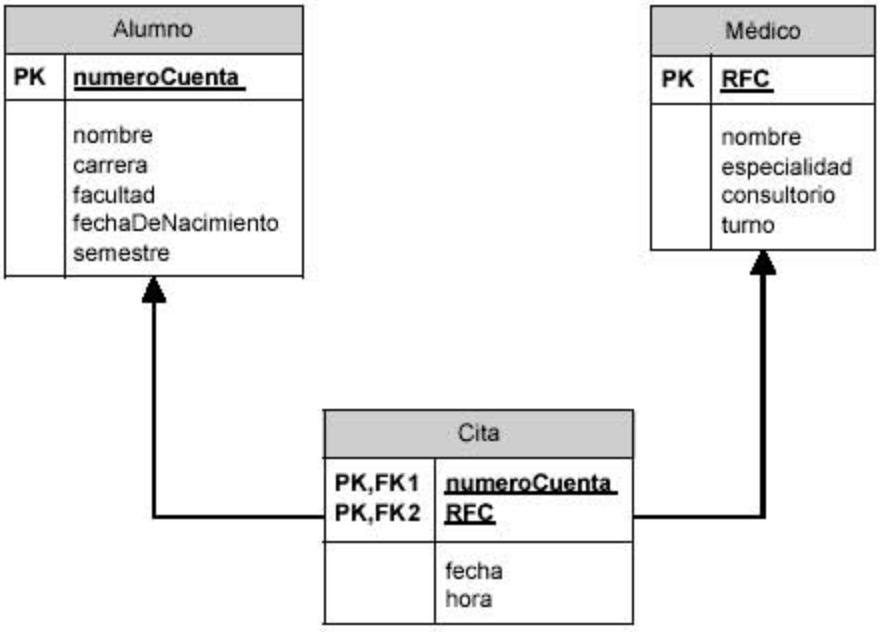
```
 /**
   * Metodo que edita los datos del alumno cuyo numero de cuenta es no_cuenta
   * los datos deben de estar en formato SQL, ej: " nombre_alumno = 'Javoncita', "
   * @param no_cuenta
   * @param nombre_alumno
   * @param carrera
   * @param nacimiento
   * @param semestre
   * @param numeroCita
   * @return cadena vacia como signo de que la actualizacion fue hecha
   */
  public static String editaAlumno (String no_cuenta, String nombre_alumno, String carrera,
                                    String nacimiento, String semestre, String numeroCita) {
        String actualizaciones =  nombre_alumno + carrera + nacimiento + semestre + numeroCita;
        if (actualizaciones.equals(""))
           return ""; //no hay nada que editar
        actualizaciones = actualizaciones.substring(0,actualizaciones.length()-2);

        try {
           ConsultaBD.realizaActualizacion(" UPDATE alumno "+
                                           " SET "          +
                                           actualizaciones  +
                                           " WHERE   no_cuenta = "+no_cuenta    +"'; ");
           return "";
        }catch (SQLException e) {
           return null;
        }
  }

}

/* Alumno.java termina aqui. */
```

Diseño de la Base de Datos



Plan de Pruebas Unitarias

CAJA NEGRA

Paquete de Control

Clase: Alumno

Método	Recibe	Regresa	Excepción
obtieneDatosAlumno	String: Número de cuenta	String: Datos del alumno	Los datos del alumno no existen o el número de cuenta es invalido
agregaAlumno	String: Nombre, String: Carrera, String: Facultad, String: Fecha de nacimiento, String: Semestre	String: Los datos completos del alumnos que se agregaron	Falta de datos o datos ya existentes
editaAlumno	String: Nombre, String: Carrera, String: Facultad, String: Fecha de nacimiento, String: Semestre	String: Los datos completos del alumno que se editaron	Falta de datos o datos inexistentes
borraAlumno	String: Nombre, String: Carrera, String: Facultad, String: Fecha de nacimiento, String: Semestre	String: Los datos completos del alumno que se eliminaron	Falta de datos o datos inexistentes

Descripción: Esta clase se encarga de realizar cadenas legibles para sql, estos métodos se encargan del manejo de los datos de los alumnos.

Paquete BD

Clase: Conexión

Método	Recibe	Regresa	Excepción
abreConexion	void	void	No se pudo abrir la conexión a la base de datos
cierraConexion	void	void	No se pudo cerrar la conexión a la base de datos

Descripción: Esta clase se encarga de mantener la comunicación con la base de datos.

Clase: ConsultaBD

Método	Recibe	Regresa	Excepción
realizaBusqueda	String: Consulta	String[]: Un arreglo de renglones de la base de datos	La consulta es inválida.
realizaActualizacion	String: Consulta	boolean: Si fue o no posible realizar la consulta	-

Descripción: Esta clase se encarga de hacer consultas o actualizaciones a la base de datos.

Paquete de Interfaz

Clase: MenuPrincial

Método	Recibe	Regresa	Excepción
muestraMenu	void	int: Opción del menú elegida por el usuario	La opción elegida es inválida
procesaOpcionesMenu	int: Opción elegida por el usuario	void	La opción elegida es inválida
main	String[]: Arreglo de parámetros	void	-

Descripción: Esta clase muestra la interfaz principal de usuario, que permite al usuario elegir entre los distintos casos de uso.

CAJA BLANCA

```
/*
 * Medico.java
 * Clase que tiene el control del Medico.
 */

public class Medico {

    /* Metodo para verificar si un medico se encuentra dentro de la base de datos
del sistema.
     * @param rfc el RFC de algun medico
     * @return true si el medico cuyo RFC es rfc, false en otro caso
     */
    public static boolean validaMedico(String rfc) {
      if (ConsultaBD.realizaConsulta
          ("SELECT rfc  FROM medico WHERE rfc = '"+rfc+"';").size() == 0 )
          return false;
      return true;
    }


/**
 * Metodo que agrega un medico al sistema a partir de todos sus datos
 * @param rfc
 * @param nombre_medico
 * @param especialidad
 * @param consultorio
 * @return los datos del medico que fue agregado al sistema
 */
    public static String agregaMedico
       (String rfc, String nombre_medico, String especialidad, String consultorio,
String turno ) {
      try {
          if (ConsultaBD.verificaConsulta
                ("SELECT consultorio, turno FROM medico WHERE consultorio = '"+
                 consultorio+"' AND turno = '"+turno+"'; ")  )
            return null;

          ConsultaBD.realizaActualizacion
             ("INSERT INTO medico VALUES ('"+rfc+
              "', '"+nombre_medico+
              "', '"+especialidad+
              "', '"+consultorio+
              "', '"+turno+"');");
          return "Ha sido agregado el medico "+nombre_medico+
                "\ncon rfc "+rfc+
                "\nespecialidad "+especialidad+
                "\nconsultorio "+consultorio+
                "\nturno "+turno;
      }catch (SQLException e) {

          return null;
      }
    }
}

/* Medico.java termina aqui. */
```

Clase Medico	Caso de prueba	Resultado esperado	Resultado obtenido
validaMedico(rfc)	Condición para un rfc existente rfc: **string = "ABCD000000"**	true	true
validaMedico(rfc)	Condición para un rfc inexistente rfc: **string = "ABCD000000"**	false	false

Clase Medico	Caso de prueba	Resultado esperado	Resultado obtenido
agregaMedico(rfc, nombre_medico, especialidad, consultorio, turno)	Condición para una agregación exitosa rfc: **string = "ABCD000000"** nombre_medico: **string = "Arturo Díaz"** especialidad: **string = "Ginecología"** consultorio: **string = "22"** turno: **string = "matutino"**	Mensaje Ha sido agregado el médico nombre_medico. Despliega los datos agregados	Mensaje Ha sido agregado el médico nombre_medico. Despliega los datos agregados
agregaMedico(rfc, nombre_medico, especialidad, consultorio, turno)	Condición para una agregación fallida rfc: **string = null** nombre_medico: **string = "Arturo Díaz"** especialidad: **string = "Ginecología"** consultorio: **string = "22"** turno: **string = null**	null	null

7.4 Artículos de apoyo para la fase

En esta fase se recomienda leer el artículo “Componiendo lo descompuesto” en el que se presenta el diagrama de estructura compuesta, que va más allá de lo requerido para el refinado de los diagramas de clases, que tiene la capacidad de englobar las partes que componen una clase sin impedir que se puedan reconocer clara y posteriormente. Este diagrama, como su nombre lo refiere, utiliza como base las relaciones entre sus clases a la composición.

Se exhorta a leer el artículo “Desarrollo guiado por pruebas”, en él se describe el proceso de implementación del sistema a partir de las pruebas, es decir, primero se genera un caso de prueba y a partir de éste se comienza a producir el código que tiene como finalidad dar un resultado exitoso al caso de prueba. Presenta el ciclo de vida de esta práctica así como sus ventajas y desventajas, entre las que destacan que al finalizar el proceso no sólo se habrá implementado el sistema sino que también se habrá probado el sistema, en contra se tiene que considerar que el proceso depende totalmente del diseño de los casos de prueba.

Por último la lectura de “Metodologías ágiles” es un recuento de éstas a través de la presentación del manifiesto ágil, así como de sus principales características, las cuales pueden ser confrontadas con las hasta ahora desarrolladas.

Orozco, Sergio.
“Componiendo lo descompuesto”, en: Software Guru. 03. 02
México D.F. 2007. pp. 44-45

Súcari, Ariel.
“*Desarrollo guiado por pruebas*”, en: Software Guru. 02. 03
México D.F. 2006. pp. 14-15

Gómez, John.
“*Metodologías ágiles*”, en: Software Guru. 01. 06
México D.F. 2005. pp. 38-39

CAPITULO 8 Integración Prueba del Sistema

8.1 Fase de Integración y Prueba del Sistema

A lo largo de esta fase se integrará el sistema para posteriormente afrontar la realización de las pruebas. Se generarán los manuales de usuario, de instalación y de mantenimiento y desarrollo.

8.2 Productos generados

En esta fase se generarán los tres distintos manuales, de usuario, de instalación y de mantenimiento, que buscan completar el objetivo de documentar la totalidad del proceso y además de actuar como consultores inmediatos a los que recurre el usuario al sucederle dudas o fallas. Por otro lado, se entregará el *Reporte de la integración* en donde se reportará lo acontecido durante la integración de los módulos del sistema. El *Informe de las pruebas del sistema* es en el que se documentan los resultados obtenidos al aplicar las pruebas.

8.2.1 Reporte de las pruebas unitarias

Tomando como base el *Plan de pruebas unitarias* se podrá generar el *Reporte de las pruebas unitarias*, creando una columna nombrada *Regresa*, que se añadirá a las tres ya existentes, *Método*, *Recibe*, *Esperado*, en la columna *Regresa* se anotarán los valores que se recibieron como respuesta a los valores de entrada colocados en *Recibe*. Idealmente los valores en la casilla de *Regresa* y *Esperado* tienen que ser los mismos, lo que implica que el diseño del sistema va de acuerdo a lo planeado.

Obtener valores distintos en estas casillas obligará a una revisión del código y un replanteamiento del caso de prueba. Las pruebas deben de cubrir a cada uno de los métodos de cada una de las clases.

8.2.2 Manual de usuario

La importancia de este documento para el usuario es prioritaria, será a lo primero que se dirija al ocurrir alguna duda mientras utiliza el sistema. Deberá estar escrito en un lenguaje que pueda entender el usuario, evitar que sea ambiguo y que deberá funcionar como una guía a través de cualquier tarea realizada en el sistema.

En cuanto a la organización del manual, deberá contener un índice del contenido en el que se distingan de manera clara los temas tratados y aún más importante es que el usuario pueda identificar en éste el espacio donde se encuentra comprendido su duda o problema.

Es aconsejable que el manual indique paso a paso el cómo realizar las tareas, para identificar cada una de éstas. El título de cada sección indicará de manera precisa qué

funcionalidad llevará a cabo, y comenzará la descripción del proceso desde la primera interfaz con que el usuario tenga contacto, describiendo cada objeto que el usuario observa en la pantalla e indicando su utilidad y cómo llevarla a cabo. Se podrá incluir en el manual la imagen de cada interfaz para un mejor entendimiento para el cliente. Cabe mencionar la opción de generar no sólo un manual físico, sino también un manual digital al que el usuario pueda acceder de manera rápida y mientras usa el sistema, este manual puede considerar situaciones comunes que se le presentan al usuario y cuya solución no requiere de una extensa explicación.

8.2.3 Manual de instalación

El manual de instalación deberá describir los pasos necesarios para instalar el sistema, los requerimientos mínimos de hardware y software existentes en las computadoras donde residirá el sistema.

Se incluirán los datos de contacto de los desarrolladores del software en caso de que el usuario final tenga problemas con la instalación del sistema. El proceso de instalación se puede incluir en el *Manual de Usuario* de ser considerado necesario. La realización de los manuales estará a cargo de los Administradores de Desarrollo.

8.2.4 Manual de desarrollo y mantenimiento

Este manual incluye la documentación generada en las fases de Especificación de Requerimientos, Diseño, Construcción y Pruebas separados de acuerdo a los ciclos a que pertenecen.

Este manual claramente estará compuesto por un lenguaje técnico y los estándares que el equipo haya impuesto al inicio del proceso de desarrollo, y es fundamentalmente para uso exclusivo de los Ingenieros de Desarrollo del proyecto o en su defecto para Ingenieros de Desarrollo ajenos al proyecto inicial que en un futuro estarán encargados de darle mantenimiento o escalar al sistema.

8.3 Ejemplos de productos generados en esta fase

A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Reporte de las Pruebas Unitarias

Las pruebas unitarias se llevaron a cabo en primera instancia por el autor de la clase, verificando que éstas compilaran, después se probaron los métodos uno a uno y finalmente todos los miembros del equipo revisaron las demás clases para aportar sugerencias ya fuera sobre la implementación o el comportamiento de alguna clase.

Paquete de Control

Clase: Alumno

Método	Recibe	Esperado	Regresó
obtieneDatosAlumno	300109133 *Número de cuenta válido	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
obtieneDatosAlumno	000000000 *Número de cuenta inválido	"#cuenta inválido"	"#cuenta inválido"
agregaAlumno	300109133, miguel, ciencias, 1984-04-04, 2008-1, 8 *Datos nuevos	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
agregaAlumno	300109133, miguel, ciencias, 1984-04-04, 2008-1, 8 *Datos ya existentes	"Datos existentes"	"Datos existentes"
editaAlumno	300109133, miguel, ciencias, 1984-04-04, 2008-1, 8	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
editaAlumno	300109133, miguel, ciencias, 2008-1, 8	"Falta de datos"	"Falta de datos"
borraAlumno	300109133 *Número de cuenta válido	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
borraAlumno	000000000 *Número de cuenta inválido	"Datos inexistentes"	"Datos inexistentes"
validaNumeroCuenta	300109133 *Número de cuenta válido	true	true
validaNumeroCuenta	000000000 *Número de cuenta inválido	false	false

Clase: Consulta

Método	Recibe	Esperado	Regresó
obtieneCita	Martha, 01/05/2004, 12:00	| 34 | 98002022 | martha | 12:00 | 01/05/2004 |	| 34 | 98002022 | martha | 12:00 | 01/05/2004 |
obtieneNumeroPacientes	Nutricion	11	11

Paquete BD

Clase: Conexión

Método	Recibe	Esperado	Regresó
abreConexion	void	void	void
cierraConexion	void	void	void

Clase: ConsultaBD

Método	Recibe	Esperado	Regresó
realizaBusqueda	SELECT * FROM medico WHERE rfc='MOTT870702' ;	Vector(MOTT870702, tania, general, 12, matutino).	Vector(MOTT870702, tania, general, 12, matutino).
realizaBusqueda	SELECT * FROM medico WHERE rfc='ABCD000000' ; El rfc ABC000000 no existe en el sistema	Vector() .	Vector() .
realizaActualizacion	INSERT INTO cita VALUES ('5', '222', 'MOTT870702', '10:00', '12-13-2004') ;	true	true
realizaActualizacion	INSERT INTO cita VALUES ('5', '222', 'MOTT870702', '10:00', '12-13-2004') ; *Tratamos de insertar la misma información por segunda vez	false	false

Paquete de Interfaz

Clase: MenuPrincial

Método	Recibe	Esperado	Regresó
muestraMenu	void	3 *Opción elegida por el usuario	3
procesaOpcionesMenu	4	void	void
main	[]	void	void

Clase: MenuAlumno

Método	Recibe	Esperado	Regresó
muestraMenuPrincial	void	3 *Opción elegida por el usuario	3
procesaOpcionMenuPricipal	3 *Opción elegida por el usuario	void	void
procesaNumeroCuenta	void:	true *Si el un número de cuenta es válido	true
procesaNumeroCuenta	void:	false *Si el un número de cuenta es inválido	false
procesaDatosAgrega	void	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
procesaDatosAgrega	void:	"Los datos no se pueden agregar"	"Los datos no se pueden agregar"
procesaDatosElimina	void	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
procesaDatosElimina	void:	"Los datos no se pueden eliminar"	"Los datos no se pueden eliminar"
procesaDatosEdita	void	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"	"| 300109133 | miguel | ciencias | 1984-04-04 | 2008-1 | 8 |"
procesaDatosEdita	void:	"Los datos no se pueden editar"	"Los datos no se pueden editar"

Manual de Usuario

ÍNDICE

Comenzando…

Al utilizar por primera vez al sistema de la Dirección General de Servicios Médicos encontrará un ambiente amigable y de fácil uso que seguramente usted disfrutará mientras desarrolla sus actividades de una manera ágil y efectiva.

El propósito de este Manuel es llevarlo por un camino de agradable tránsito hacia su objetivo, que usted pueda realizar todas las actividades necesarias para coordinar a los alumnos, médicos y las citas de esta institución.

¿Qué quiero hacer?

El sistema le permitirá realizar dos actividades principalmente, las cuales le describimos enseguida:

REGISTROS

En la pantalla principal podrá elegir entre tres opciones de registro, cada una corresponde a distintas acciones, ya que puede registrar alumnos, médicos o citas.

Si usted eligió la opción **'Registrar alumno'** se desplegará en la pantalla en la que podrá ingresar los datos en el orden en el que los requiere el formulario. Si desea limpiar la forma porque cometió algún error deberá presionar el botón **'Limpiar'**, en caso contrario, asegúrese de que los datos estén correctos y al terminar presione el botón **'Registrar'.** Entre los campos que requiere el formulario están número de cuenta, nombre, facultad y semestre, no olvide que para poder completar satisfactoriamente el registro de un alumno este deberá estar debidamente inscrito en el semestre lectivo.

Por otra parte, el sistema le permite registrar médicos y citas, presionando **'Registrar médico'** o **'Registrar citas'** respectivamente. En cualquier caso el proceso es el mismo que para el caso del alumno, se desplegará una pantalla con un formulario para ingresar los datos necesarios.

CONSULTAS

En este caso particular en la pantalla inicial deberá elegir la opción **'Consultar Sistema'** , acción que inmediatamente lo llevará a la pantalla de Consultas, en ella podrá elegir dentro de la lista **'Tipo de Consulta'** aquella consulta que quiera realizar, cuando la haya seleccionado deberá presionar el botón **'Consultar'**.

Podrá realizar principalmente dos tipos de consultas:

Consultar Citas: en esta opción el sistema le entregará un número que corresponderá a la asignación de alumnos a médicos en determinada fecha o especialidad.

Consultar Alumnos o Médicos: dentro de una lista podrá elegir la categoría por la cual quiere que se clasifiquen los alumnos o médicos registrados en el sistema.

¿Problemas?

* En efecto, soy el administrador pero el sistema no me permite hacer consultas.

Esto puede suceder si la categoría elegida para hacer la consulta es inválida, por ejemplo, si se pide ordenar a los alumnos por RFC.

* El alumno no puede ser registrado.

Esto puede deberse a que el alumno que quiere ser registrado no proporciono datos válidos o no se encuentra inscrito en el semestre lectivo.

Soporte Técnico

¿Cómo encontrarnos?

En México Human Soft pone a su disposición la siguiente dirección de correo electrónico para que nos haga saber cualquier duda, comentario, queja o problema que le acontezca.

soporte@humansoft.com

Manual de Instalación

Índice

Requerimientos del Sistema.

- *J2EE 1.4 SDK Sun Java System Application Server Platform Ed. 8*
- 5MB de memoria virtual
- 128Mb de memoria RAM

Nota: Si no se cuenta con Java, una versión se encuentra disponible en la siguiente dirección
http://java.sun.com/

Modo de Instalación.

Linux:

Deberá copiar el contenido del directorio Linux del disquette de instalación al directorio home del usuario. Enseguida se describen cada uno de los pasos.

1. Lo primero que tiene que hacer es montar el disco de la siguiente manera (suponemos que el usuario utiliza bash como su shell):

   bash$mount /dev/fd0

2. Una vez montado el disco, cámbiese al directorio sis dentro del directorio Linux del

   bash$cd /mnt/floopy/Linux/sis

3. Muevase ahora el directorio sis a su directorio home. Para esto escriba:

   bash$cp * $HOME

   Esto copiara todo el contenido del directorio sis al directorio que especifique la variable HOME.

4. Desmonte, si lo desea, el escribiendo las siguientes líneas de comando:

   bash$cd $HOME/sis <ENTER>
   bash$umount /dev/fd0 <ENTER>

5. Lo siguiente que tiene que hacer es compilar el Sistema para esto escriba:

   bash$make all

6. Por último hay que ejecutar el Sistema. Se tiene que cambiar al directorio bin

   bash$cd bin

   y ejecutar desde ahí el Sistema, para esto escriba:

   bash$java inge.inter.MainFrame <ENTER>

7. Lo primero que verá es una ventana de Bienvenida. A partir de aquí las especificaciones del modo de empleo del Sistema se encuentran en el Manual de Usuario.

Windows:

Deberá copiar el contenido del directorio SIS que está dentro del directorio Windows del disquette al directorio raíz de la computadora. Enseguida se describen cada uno de los pasos.

1. Abra una terminal de MSDOS (símbolo del sistema) en su máquina y ejecute la siguiente secuencia de comandos:

```
C:>copy A:Windows\SIS\* C:\ <ENTER>
C:>sis.bat <ENTER>
```

La última línea de comandos puede llevarse a cabo dando doble clic a MiPC, luego doble click en C: que mostrará todos los archivos del directorio raíz y dar doble click sobre el archivo sis.bat.

8.4 Artículos de apoyo para la fase

Muy aconsejable es la lectura del artículo “El proceso de la prueba del software” ya que presenta las ventajas y desventajas del modelo de prueba utilizado durante el curso, que es el “Modelo-V”. Entre las principales ventajas está que la primera mitad del proceso de pruebas se realiza paralelamente a las fases de Especificación de de Requerimientos, la de Diseño y la de Construcción, pero por otro lado describe cómo la segunda mitad presenta la dificultad de mantener actualizada la documentación generada previamente al surgir algún cambio en el sistema. Para concluir presenta una manera alternativa que permite combatir el problema de mantener consistente a la documentación con el sistema.

En el artículo “Qué es usabilidad?” se describe ampliamente el concepto de usabilidad de un sistema de software además de proponer una serie de pasos para conseguirla. En esta fase del proceso ya se habrá prácticamente concluido con el sistema, pero no por ello dejará de ser útil reconsiderar qué tan usable es el sistema, después de leer el artículo seguramente se podrán proponer mejoras para considerar modificar ciertas partes del diseño del software para el segundo ciclo. Con esto no sólo se busca incrementar la calidad del producto sino también la satisfacción del usuario final.

León, Luis.
“*El proceso de la prueba del software*”, en: Software Guru. 01. 06
México D.F. 2005. pp. 46-47

Maurer, Donna.
“*¿Qué es usabilidad?*”, en: Software Guru. 02. 05
México D.F. 2006. pp. 25-27

CAPITULO 9 Cierre

9.1 Fase de Cierre

La importancia de la *fase de Cierre* recae en la entrega del producto terminado y la evaluación del desempeño de cada uno de los miembros del equipo.

En esta fase quedarán reportadas aquellas experiencias vividas por los miembros del equipo y que consideren de utilidad para un ciclo posterior o un nuevo proyecto.

Se entregará un reporte final indicando el estado de todos y cada uno de los elementos del sistema así como un informe final de mediciones de lo hecho a lo largo del ciclo.

9.2 Productos generados

Los productos generados en esta fase son meramente para retroalimentación del equipo, por un lado tenemos la *Forma de evaluación del equipo y personal* para ubicar las debilidades del equipo y poder sanarlas o extirparlas de ser necesario. Las *Lecciones aprendidas y sugerencias de mejoras* que serán de gran ayuda para incrementar la eficiencia y la calidad de futuros procesos. Por otro lado el *Informe de mediciones* actúa como indicador del tiempo invertido y de la calidad del producto, mientras que el *Informe del estado de la configuración* es el concentrado de la ubicación, estado y versión finales de cada uno de los productos que conforman al sistema.

9.2.1 Forma de evaluación del equipo y personal

Este documento busca plasmar una crítica del desempeño personal de los individuos que participaron en el desarrollo del sistema, del equipo en sí, y se deberán identificar aquellos aspectos exitosos así como los que no resultaron como se esperaba a lo largo del ciclo.

Escribir comentarios o críticas constructivas de uno mismo como de cada uno de los demás miembros del equipo con el fin de minimizar obstáculos en los ciclos o procesos posteriores. Esta última sección del documento, es donde se evalúa el desempeño de cada rol.

9.2.2 Lecciones aprendidas y sugerencias de mejoras

En el documento de Lecciones Aprendidas, estarán englobadas las lecciones experimentadas por cada miembro del equipo tanto individual como colectivamente. Se recomienda analizar aquellas prácticas que el equipo considera que beneficiaron el desempeño o que resultaron en una experiencia exitosa, identificar los problemas más recurrentes y a éstos proponerles soluciones o sugerencias para minimizarlos o evitarlos.

Se identifican las mejores prácticas en los rubros del equipo y el proceso. Los problemas recurrentes tanto en el funcionamiento del equipo como en el proceso. Las

experiencias exitosas también se registrarán. En cuanto a las sugerencias de mejora, se reflexiona sobre lo que habrá que mejorar tanto al proceso como al equipo.

9.2.3 Informe de mediciones

El informe de mediciones incluye el cálculo de los minutos invertidos en cada fase así como el tiempo total del ciclo, el tamaño de los productos generados en cada fase así como el total de productos elaborados. El conteo estará a cargo del Administrador de Planeación, ya que este rol tuvo como obligación a lo largo de todo el ciclo el control de los tiempos invertidos por el equipo en la *Forma semana del equipo.*

Por otra parte el Administrador de calidad, que fue el que llevó el control y registro de los defectos encontrados, elaborará una tabla listando todos los productos generados a lo largo del ciclo y divididos por fase en la que indique con exactitud cuántos defectos se encontraron en cada uno de los documentos o productos mencionados. Posteriormente encontrará el índice de errores por cada producto, es decir tomará en cuenta el número de defectos encontrados en un producto entre el tamaño de éste. Un número pequeño indicará que hubo pocos defectos y que el producto fue de buena calidad. Por el contrario, un número grande señala que el producto no es de buena calidad y deberá ser elaborado nuevamente.

Este documento consta de dos partes, la primera resulta muy útil para la planeación de ciclos posteriores ya que se ha creado una idea de los tiempos empleados, y la segunda es un control que indica la calidad del producto resultante.

9.2.4 Informe del estado de la configuración

Este documento contiene la información final de los elementos que conforman al sistema. Para este producto existe una forma del mismo nombre que está compuesta de la siguiente manera: una columna en la que se listan los nombres de los productos generados a lo largo del ciclo, en la segunda columna se lista la última versión registrada del elemento de la configuración, en la última columna se indicará la dirección donde se encuentra localizado cada uno de los documentos dentro del depósito electrónico del equipo.

9.3 Ejemplos de productos generados en esta fase

A continuación se incluye un ejemplo de cada uno de los productos anteriormente descritos.

Forma de Evaluación del Equipo y Personal

Equipo Human Soft	**Ciclo** UNO

Evaluación del equipo:

Evaluación del equipo sobre los siguientes criterios. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Espíritu de equipo	1	2	3	4	5
Efectividad del equipo	1	2	3	4	5
Experiencia adquirida	1	2	3	4	5
Productividad del equipo	1	2	3	4	5
Calidad del proceso	1	2	3	4	5
Calidad del producto	1	2	3	4	5

Evaluación de cada rol:

Para cada rol, evalúe el porcentaje de trabajo requerido y la dificultad durante este ciclo		
Rol	Trabajo requerido	Dificultad en el rol
Líder de Equipo	25%	Alta
Administrador de Desarrollo	30%	Alta
Administrador de Planeación	15%	Media
Administrador de Calidad y Proceso	15%	Media
Administrador de Apoyo	15%	Media
Contribución total (100%)	100%	

Evalúe la contribución de cada rol. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Líder de Equipo	1	2	3	4	5
Administrador de Desarrollo	1	2	3	4	5
Administrador de Planeación	1	2	3	4	5
Administrador de Calidad y Proceso	1	2	3	4	5
Administrador de Apoyo	1	2	3	4	5

Evalúe cada rol de acuerdo al apoyo y ayuda proporcionada. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Líder de Equipo	1	2	3	4	5
Administrador de Desarrollo	1	2	3	4	5
Administrador de Planeación	1	2	3	4	5
Administrador de Calidad y Proceso	1	2	3	4	5
Administrador de Apoyo	1	2	3	4	5

Evalúe cada rol de acuerdo a su desempeño. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Líder de Equipo	1	2	3	4	5
Administrador de Desarrollo	1	2	3	4	5
Administrador de Planeación	1	2	3	4	5
Administrador de Calidad y Proceso	1	2	3	4	5
Administrador de Apoyo	1	2	3	4	5

Lecciones Aprendidas y Sugerencias de mejoras

Lecciones aprendidas

Mejores prácticas

Del equipo	Del proceso
Buena comunicación en el equipo.	Implementar el sistema de versiones.
Se mejoró la calidad del producto conforme se avanzó el proyecto.	

Problemas recurrentes

Del equipo	Del proceso
Retraso en la entrega de documentación.	No llevar a cabo las revisiones de calidad.
Inconsistencia de algunos miembros del equipo.	Fallas en los estándares y la calidad.

Experiencias exitosas

Del equipo	Del proceso
Terminar el proceso y cumplir con las expectativas.	Entregar un sistema funcionando en su totalidad para el ciclo 1.
Repartición del trabajo exitosa.	

Comentarios

El equipo se desarrolló bien dentro de un marco algo ajustado, con algunas fortalezas y otras flaquezas, pero salió adelante con lo que se proponía en el ciclo 1.
Al principio parecía que el equipo no iba a funcionar, pero los integrantes nos pusimos la camiseta y sacamos el trabajo a tiempo.

Sugerencias de mejora

Para el proceso de desarrollo

Evitar los retrasos tanto en la elaboración de documentos como al programar.
Mejorar la comunicación entre cada uno de los desarrolladores.
Adherirnos más a lo que dice el estándar.
Cumplir con las revisiones de calidad.

Mejoras al equipo de desarrollo.

Mejorar la comunicación entre los integrantes que desarrollan las capas del sistema.
Aumentar la coordinación entre los integrantes que tienen que entregar documentación enlazada.

Informe de Mediciones

Tiempos por fase

Fase	Resumen de tiempos
Lanzamiento	420 min.
Estrategia	480 min.
Planeación	1060 min.
Requerimientos	1360 min.
Diseño	1470 min.
Implementación	1440 min.
Pruebas	300 min.
Cierre	900 min.
Tiempo total del ciclo	7430 min.

Tamaño de los productos

Fase	Tamaño	Unidades
Lanzamiento	20	hojas
Estrategia	13	hojas
Planeación	15	hojas
Requerimientos	63	hojas
Diseño	17	hojas
Implementación	8	hojas
Pruebas	3	hojas
Cierre	30	hojas

Defectos encontrados por producto

Fase	Número de Defectos
Lanzamiento	4
Estrategia	2
Planeación	2
Requerimientos	7
Diseño	5
Implementación	1
Pruebas	6
Cierre	0
Total de defectos	27

Productividad y calidad de los productos

Fase	Tamaño del código / tiempo total de desarrollo	Número de defectos / tamaño de producto específico
Fase de lanzamiento	420 min.	4/20 hojas
Fase de estrategia	480 min.	2/13 hojas
Fase de planeación	1060 min.	2/15 hojas
Fase de captura de requerimientos	1360 min.	7/63 hojas
Fase de diseño	1470 min.	5/17 hojas
Fase de implementación y de pruebas unitarias	1440 min.	1/8 hojas
Fase de pruebas del sistema	300 min.	6/3 hojas
Fase de cierre	900 min.	0/30 hojas

Forma del Informe del Estado de la Configuración

De acuerdo con el Plan de la configuración, al final del primer ciclo este es el estado final de la configuración

Componente y localización	Versión	Estado
HumanSoft/Proyecto/Ciclo1/LAN1/Agenda	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Minuta	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Objetivos	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Estándar	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/LAN1/Forma de registro de riesgos	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/EST1/Forma estrategia	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/EST1/Depósito	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/EST1/Plan de configuración	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/EST1/Informe del estado de la configuración	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/EST1/Informe semanal estado cambios	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/PLAN1/Plan equipo	1.2	Terminado
HumanSoft/Proyecto/Ciclo1/PLAN1/Forma de registro de defectos	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Texto definición del problema	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Glosario	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Requerimientos no funcionales	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Diagrama general de casos de uso	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Detalle de los casos de uso	1.3	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Prototipo de la interfaz del usuario	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/REQ1/Plan de pruebas del sistema	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Especificación ambiente implementación	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Estándar de diseño	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Arquitectura con diagrama de paquetes	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Diagrama de distribución	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Plan de pruebas de integración	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Diagramas de clases	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Diagramas de secuencia	1.2	Terminado
HumanSoft/Proyecto/Ciclo1/DIS1/Diagrama de estado	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/CONS1/Diagramas de clases refinados	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/CONS1/Código para las clases	1.3	Terminado
HumanSoft/Proyecto/Ciclo1/CONS1/Plan de pruebas unitarias	1.2	Terminado
HumanSoft/Proyecto/Ciclo1/PRU1/Reporte de las pruebas unitarias	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/PRU1/Manual de usuario	1.1	Terminado
HumanSoft/Proyecto/Ciclo1/PRU1/Manual de instalación	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/PRU1/Manual de desarrollo y mantenimiento	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/CIE1/Lecciones aprendidas sugerencias mejoras	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/CIE1/Forma de evaluación del equipo y personal	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/CIE1/Informe de mediciones	1.0	Terminado
HumanSoft/Proyecto/Ciclo1/CIE1/Forma informe estado de la configuración	1.0	Terminado

Informe de la actividad de control de cambios

Formas de Solicitud de cambios:	A lo largo del ciclo
Enviadas	1
Aprobadas	0
Rechazadas	1
En reversa (cambio que debe deshacerse)	0
En proceso	0

9.4 Artículos de apoyo para la fase

Para esta última fase se recomienda la lectura de el código de ética de los ingenieros de software, en el que se describen los principios que debe respetar todo individuo que se precie de ser un ingeniero de software. Entre los aspectos a destacar contenidos en el código propuesto por ACM y la IEEE están aquéllos que tienen que ver con atributos sociales, *Público, Cliente y Empleador, Personal y Colegas,* y aquéllos que tienen que ver más con la función que se desempeña como lo son *Producto, Juicio, Administración y Profesión.*

Por último se recomienda la lectura de las diferentes normas o modelos de calidad propuestas por distintas instituciones, como lo son CMMI en Estados Unidos de América y MoProSoft en México.

"Código de ética"
http://www.acm.org/serving/se/code.htm
"CMMI"
http://www.sei.cmu.edu/cmmi
"MoProSoft"
http://victoria.fciencias.unam.mx/MoProSoftV1.3/espaniol/MoProSoft.html

Conclusiones.

A lo largo de la realización de este trabajo se reafirma la importancia del proceso de desarrollo de software que se sigue en el curso de Ingeniería de Software, ya que no sólo se logra mostrar al proceso unificado, sino que este proceso eleva sustancialmente la calidad del producto al término de cada ciclo, mostrando a los alumnos que lo llevaron a cabo las ventajas que trajo y los posibles problemas que se evitaron al organizar de manera efectiva todo el desarrollo del sistema y del equipo. Aprovechando la experiencia obtenida en las demás asignaturas de la Licenciatura en Ciencias de la Computación pude observar detalles que pasé por alto cuando cursé la materia, los cuales intenté recalcar para que los alumnos que recurran a este trabajo los noten desde el principio y aprovechen lo mejor posible su curso.

Un aspecto muy importante fue la posibilidad de tener acceso al material generado por alumnos a lo largo de sus respectivos cursos en licenciatura y posgrado, con ello pude notar a qué características se tendría que prestar más atención al generar los productos requeridos en el proceso. Para los alumnos, este trabajo será un buen auxiliar para llevar un proceso de desarrollo de software todavía más provechoso. Al final, se ha cumplido el objetivo propuesto desde un inicio de que este trabajo sirva como un apoyo didáctico para el curso de Ingeniería de Software.

Por otro lado el incluir textos de profundización o familiarización con algunos temas al final de cada capítulo, tiene el objetivo de que el alumno se interese y tenga una referencia casi inmediata de diversos tópicos que mejorarán la calidad de su proceso y facilitarán su entendimiento al aplicarlos por sí mismo, además se pretende que forme el hábito de leer artículos especializados para que se mantenga actualizado en su vida profesional.

En cuanto a la página, se han incorporado los ejemplos y las ligas a los artículos en la sección de “Apoyos al libro de Ingeniería de Software Pragmática”, pero además hemos anexado y enriquecido algunas de las definiciones que se encontraban en este sitio.

En lo personal la realización de este trabajo me deja un firme conocimiento del tema, tanto en su aspecto teórico como el lado práctico. Así mismo puedo constatar el haber experimentado un sentido de orgullo y satisfacción al haber elaborado este trabajo y poder dejar un escrito que espero sea utilizado por las próximas generaciones, que les aclare sus dudas y que los incite a mejorarlo. Y el interés no disminuyó, pienso seguir en esta área de las Ciencias de Computación ya que todavía hay mucho que explorar.

Este trabajo ha cumplido con su objetivo, una parte del cual era proveer de ejemplos al libro de texto y la página del curso de “Ingeniería del Software”, la otra parte está siendo realizada por aquellos alumnos que cursan esta materia en la Facultad de Ciencias y serán ellos quienes decidan qué tan didácticos y útiles fueron los ejemplos propuestos aquí.

Bibliografía.

[1] Braude, Eric J.
"Ingeniería de Software: una perspectiva orientada a objetos", Boston Univ. 2001.

[2] Humphrey, Watts S.
"Introduction to the Team Software Process", Addison-Wesley, 2000.

[3] Ibargüengoitia, G., Oktaba, H.
"*Ingeniería de Software Pragmática"*. Por publicarse.

[4] Jaote, P.
"A*n Integrated Approach to Software Engineering"*. Springer, 3a edición, 2005.

[5] Sommerville, I.
"*Software Engineering"*. Pearson Education, 5a edición, 1996.

Artículos

Alegría V., Mónica.
"*El ABC de un taller de requerimientos*", en: Software Guru. 02. 03
México D.F. 2006. pp.34-35

Bautista, Y., Calleja, R.
"*Administración de cambios del software*", en: Software Guru. 03. 02
México D.F. 2007. pp.30-32

Cuellar, Luis.
"*La cultura de medir*", en: Software Guru. 02. 05 México D.F. 2006. p.10

Cuesta R., Saúl.
"*Patrones de casos de uso*", en: Software Guru. 01. 06
México D.F. 2005. pp.40-42

Esparza, Alfonso.
"*Usabilidad*", en: Software Guru. 02. 05
México D.F. 2006. p.46

Gómez, John.
"*Metodologías ágiles*", en: Software Guru. 01. 06
México D.F. 2005. pp. 38-39

León, Luis.
"*El proceso de la prueba del software*", en: Software Guru. 01. 06
México D.F. 2005. pp. 46-47

Maurer, Donna.
"*¿Qué es usabilidad?*", en: Software Guru. 02. 05
México D.F. 2006. pp. 25-27

Orozco, Sergio.
"*Componiendo lo descompuesto*", en: Software Guru. 03. 02
México D.F. 2007. pp. 44-45

Orozco, Sergio.
"*Del negocio al sistema: El diagrama de actividad*", en: Software Guru. 02. 04
México D.F. 2006. pp. 46-47

Orozco, Sergio.
"*Divide y vencerás*", en: Software Guru. 02. 05
México D.F. 2006. pp. 42-43

Orozco, Sergio.
"*La vida de un objeto*", en: Software Guru. 02. 03
México D.F. 2006. pp. 44-45

Orozco, Sergio.
"*Modelo conceptual*", en: Software Guru. 01. 06
México D.F. 2005. pp. 44-45

Súcari, Ariel.
"*Desarrollo guiado por pruebas*", en: Software Guru. 02. 03
México D.F. 2006. pp. 14-15

Referencias electrónicas.

[6] *CMMI* [en línea]. Disponible en World Wide Web: <http://www.sei.cmu.edu/cmmi>.

[7] *Código de ética* [en línea]. Disponible en World Wide Web:
<http://www.acm.org/serving/se/code.htm>

[8] *MoProSoft* [en línea]. Disponible en World Wide Web:
<http://victoria.fciencias.unam.mx/MoProSoftV1.3/espaniol/MoProSoft.html>

[9] *Página del curso de Ingeniería de Software* [en línea]. Disponible en World Wide Web:
<http://victoria.fciencias.unam.mx/cursois >.

[10] *Software Engineering* [en línea]. Disponible en World Wide Web:
<http://www.comp.lancs.ac.uk/computing/resources/ser>.

[11] *Software Guru* [en línea]. Disponible en World Wide Web: <http://www.sg.com.mx>.