UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

"APLICACION DE LA TECNOLOGIA ORIENTADA A OBJETOS, EL LENGUAJE DE MODELADO UNIFICADO (UML) Y EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE EN UN CASO DE ESTUDIO"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A:

EDDIE DAVID BASTO AGUILAR

DIRECTORA DE LA TESIS: DRA. HANNA OKTABA

MEXICO, D. F. 2001

Dedicado a mi familia,
Maria y Kevin,

a mis padres, hermanos y sobrinos

Agradecimientos:

A la Dra. Hanna Oktaba, por brindarme sus conocimientos y apoyo, en la realización de esta tesis. Gracias tutora.

Muy en especial a la M. en C. Guadalupe Ibargüengoitia, por su total disposición a ayudarme en todo momento. Muchísimas gracias Lupita.

También quiero agradecer a mis sinodales: Dr. Manuel Romero Salcedo, Mtro. Gustavo Márquez Flores, M. en C. Juan Jesús Moncada Bolón, por sus consejos tiempo y dedicación. A Lulú, Violeta y Juanita por su ayuda siempre sincera.

A la Universidad Autónoma de Campeche, por darme el apoyo necesario.

Eddie David Basto Aguilar.

Índice General

Capítulo 1

Introducción

Un proceso de producción de sistemas de software es el que define el qué, quién, cómo y cuándo lo hace o realiza éste proceso para llegar a un objetivo. Para la realización del proceso se necesita tener un cuerpo de conocimiento amplio, que nos ayude a realizar las tareas sistemáticamente. Estos conocimientos nos sirven para saber: el tipo de tecnología a utilizar, el modelo escogido para realizar el análisis y el diseño del sistema, el mismo proceso de desarrollo, así como los lenguajes y herramientas necesarias.

Para llegar al proceso de producción de un sistema de software utilizando la tecnología orientada a objetos, se tiene que tener conocimiento de:

- La tecnología orientada a objetos.
- El lenguaje de modelado unificado.
- Los lenguajes y herramientas de desarrollo.
- El proceso de desarrollo unificado de software.

En el capítulo 2 se describe al cuerpo de conocimientos utilizados en nuestro caso de estudio [1].

1.1 Antecedentes

En los años 70´s las tecnología orientada a objetos se encontraba en sus primeras etapas: en el desarrollo de herramientas de software y lenguajes de alto nivel. En ese entonces estas herramientas no eran compatibles con el hardware existente, ya que éste tenía poca capacidad de procesamiento y memoria reducida. En la segunda mitad de la década de los 80´s fue cuando surgieron las herramientas y los lenguajes apropiados para el desarrollo de sistemas de software orientados a objetos. Además que los nuevos microprocesadores eran más rápidos y eficientes, así también las memorias de las computadoras poseían mayor capacidad, haciendo posible ejecutar estos sistemas de software de manera eficiente.

Para el desarrollo de sistemas de software con tecnologías anteriores a la orientada a objetos, se utilizaron herramientas y técnicas de modelado como los diagramas de flujo, diagramas de flujo de datos, algoritmos, seudo-código y los lenguajes de especificaciones formales. Con la nueva tecnología se necesitaba contar con nuevas herramientas y técnicas de modelado que se ajustaran a la filosofía de la orientación a objetos.

Tres métodos fueron muy bien aceptados a principios de la década de los 90´s. Métodos creados por investigadores de renombre como: Grady Booch, Ivar

Jacobson y James Rumbaugh. Los mismos autores a finales de la década de los 90´s decidieron unificar sus propuestas creando el Lenguaje de Modelado Unificado (UML por sus siglas en inglés). UML ofrece una simbología única para el análisis, diseño y desarrollo de los sistemas de software con tecnología orientada a objetos.

Después de obtenerse un lenguaje de modelado, el siguiente paso lógico a seguir era el de trabajar para obtener un proceso de software acorde con las necesidades de la tecnología orientada a objetos y de esta manera surgió el proceso unificado de desarrollo de software.

Por lo mencionado anteriormente podemos concluir que la tecnología orientada a objetos está a la vanguardia en el desarrollo de productos de software.

1.2 Objetivos

En éste documento se realiza una descripción detallada de cómo se utilizó la tecnología orientada a objetos en un caso de estudio. Éste caso de estudio es un sistema de software necesario para la inscripción de aspirantes, al posgrado en Ciencia e Ingeniería de la Computación de la Universidad Nacional Autónoma de México. El sistema se desarrolló por un equipo 8 personas, formado por alumnos de éste posgrado. Durante el desarrollo de la materia Ingeniería de Software Orientada a Objetos. Impartida en el ciclo escolar septiembre del 99 – enero del

2000, en las instalaciones del Instituto de Investigaciones en Matemáticas Aplicadas y de Sistemas (IIMAS).

La presente tesis pretende hacer un concentrado de todo el proceso de desarrollo de software haciendo énfasis en los aspectos más importantes de éste desarrollo, creando con esto un documento accesible y de fácil comprensión a todo lector con interés en el Proceso de Desarrollo Unificado de Software.

1.3 Organización de la tesis

En el capítulo 2 se da un panorama general del cuerpo de conocimiento necesario para realizar el proceso de desarrollo orientado a objetos. Se define a la tecnología orientada a objetos y el lenguaje de modelado unificado. Así como al proceso de desarrollo de software y las herramientas y lenguajes de implementación.

En el capítulo 3 se definen los conceptos para la formación de un efectivo equipo de desarrollo. Los roles que necesarios para la integración del equipo de desarrollo. La planeación de las tareas que cada miembro del equipo debe realizar de acuerdo a su rol dentro del equipo.

En el capítulo 4 se realiza una abstracción del proceso de desarrollo del proyecto SINSAS. Se presentan a las fases 3 primeras fases del proceso de desarrollo unificado de software (inicio, elaboración y construcción). Se incluyen conceptos sobre el manejo de riesgos y la forma de cómo medir al software (métricas).

En el capitulo 5 se presenta a la fase de transición de éste proceso, la que incluye a las pruebas y la liberación del producto. También se presentan las actividades de mantenimiento realizadas al software.

Por último el capítulo 6 presenta las conclusiones sobre los aspectos estudiados, las perspectivas así como algunos tópicos aún objeto de investigación. Al final se presenta a la bibliografía utilizada así como los anexos citados.

Capítulo 2

Cuerpo de conocimiento

En éste capítulo se presentan los conocimientos básicos que se deben tener para desarrollar un software con tecnología orientada a objetos. Estos conocimientos van de los teóricos a los prácticos. En los conocimientos teóricos se presenta la definición de la tecnología orientada a objetos hasta la definición del proceso de desarrollo de software orientado a objetos. En los prácticos se explican las técnicas, herramientas y lenguajes de desarrollo e implementación.

2.1 La tecnología orientada a objetos

La tecnología orientada a objetos abarca todas aquellas, herramientas, lenguajes, métodos y técnicos utilizados para modelar y desarrollar sistemas de software mediante una filosofía orientada a los objetos. Esta tecnología usa las ventajas de la reusabilidad y el encapsulamiento de la orientación a objetos.

Hace 30 años cuando se dio a conocer a la tecnología orientada a objetos, no se tenía el desarrollo tecnológico necesario en el hardware para desarrollar sistemas de software eficientes. Por ello esta tecnología no se popularizó de manera rápida como las otras que existían en esos momentos, pero esto no quiere decir que no se siguió investigando al respecto. Fue hasta mediados de la década de los 80´s que se contó con el hardware necesario para la realización a nivel general de sistemas orientados a objetos. Apareciendo muchos modelos para el proceso de desarrollo y producción de estos sistemas. A lo largo de estos años, se han desarrollado varios métodos sobre la orientación a objetos, abarcando principalmente el análisis, el diseño y la implementación [1].

2.1.1 La orientación a objetos

Para poder entender que es la orientación a objetos, debemos definir el concepto de objeto, clase, herencia y encapsulamiento.

Objeto:

Un objeto es una entidad que encierra sus propios datos y operaciones, es la instancia de la clase en tiempo de ejecución. El objeto se puede transformar a través de estímulos enviados por medio de mensajes que pueden modificar sus características particulares.

Clase:

La clase caracteriza a un conjunto de objetos. En esta se definen los atributos y métodos propios de cada objeto.

Herencia:

La herencia es una característica que tienen las clases de utilizar atributos y métodos de otras clases con el fin de reutilizar el código existente. De esta manera podemos crear nuevas clases a partir de otras ya existentes haciendo más claró y eficiente el modelado de sistemas.

Encapsulamiento:

El encapsulamiento es la propiedad de la orientación a objetos en la que se oculta la forma como se implementan los métodos y atributos de los objetos. Por esto solamente es necesario conocer los nombres de los métodos, ya que a través de estos se pasan los mensajes que estimulan al objeto modificando con esto sus atributos. El encapsulamiento oculta los atributos del objeto no permitiendo que otros objetos puedan modificarlos directamente a no ser que el objeto les de derecho a realizar estas modificaciones.

Estos conceptos fundamentales de la orientación a objetos son la base para modelar la funcionalidad de un sistema de software. El lenguaje de modelado unificado (UML), tiene un conjunto de símbolos y de diagramas basados en el

modelado de objetos para representar diferentes vistas de un sistema de software [2].

2.2 El Lenguaje de Modelado Unificado

El lenguaje de Modelado Unificado (UML siglas en inglés) es un lenguaje para la visualización, especificación, construcción y la documentación de productos de un sistema de software. Con el UML se obtiene un camino estándar para describir diferentes aspectos de un sistema, tales como el modelo del negocio y las funciones del sistema. Como su nombre lo indica, UML es un lenguaje unificado, esto quiere decir que fue creado con base en tres propuestas independientes, que finalmente se reunieron para conformar al UML [2].

El lenguaje está definido por una gramática formal y consta de símbolos visuales muy fáciles de interpretar, como lo son los diagramas, que visualizan las partes estáticas y dinámicas de un sistema.

Los diagramas de clase visualizan la parte estática del sistema, definen y caracterizan a un objeto, en cuanto a sus atributos y métodos así como las relaciones existentes entre ellos.

Los diagramas de interacción, sirven para visualizar la parte dinámica del sistema. Entre estos diagramas se encuentran los de diagramas de secuencia y de colaboración. Otro tipo de diagramas dinámicos son los diagramas de casos de uso, estos diagramas son muy importantes ya que son un primer acercamiento al modelado del sistema, dentro del análisis de los requerimientos. A continuación

describiremos a los diagramas de casos de uso, de clase y de secuencia que son los diagramas más importantes en el modelado del sistema. Estos diagramas deberán ser generados en las etapas de concepción y elaboración del proceso de desarrollo unificado de software que describiremos más adelante.

2.2.1 Definición del modelo de caso de uso

Para guiar a los desarrolladores a una implementación eficiente del sistema, apegado a las necesidades del cliente: se utiliza el Modelo de Casos de Uso [2]. Para realizar el modelo de casos de uso, primero se debe realizar la captura de requerimientos. La captura de requerimientos tiene dos objetivos principales:

Encontrar los requerimientos reales:

Que los requerimientos cuando sean implementados reproduzcan el valor esperado por los usuarios.

Representar los requerimientos:

De una forma entendible para los usuarios, clientes y desarrolladores. Significa que la descripción resultante de los requerimientos deberá ser entendible por usuarios y clientes.

El conjunto de actores y de casos de uso de un sistema representan el Modelo de Casos de Uso, el cuál describe las funcionalidades completas del sistema. La

identificación de los actores y los casos de uso la podemos realizar con ayuda de la captura de los requerimientos.

El modelo de casos de uso tiene los siguientes elementos:

ACTOR:

Normalmente un sistema tiene muchos tipos de usuarios. Cada tipo de usuario es representado por un actor. Los actores usan el sistema tal como está definido en los casos (ver figura 2.1).

Figura 2.1: Actor.

CASO DE USO:

Un caso de uso es una parte de la funcionalidad del sistema que representa el valor de un resultado que va a ser proporcionado al usuario (ver figura 2.2).

Figura 2.2: Caso de uso.

Escenario:

Secuencia de acciones que el sistema ejecuta para ofrecer algún resultado a un actor. La relación existente entre los casos de uso y los escenarios, es que un caso de uso puede tener varios escenarios. La figura 2.3 muestra un ejemplo vinculado con nuestro caso de estudio (se utiliza el símbolo gráfico de una nota para describir el escenario).

1.- El coordinador accesa a la página del sistema SINSAS
2.- El sistema presenta una pantalla de Bienvenida
3.- El sistema pide password al coordinador
4.- El coordinador introduce password
5.- El sistema valida que el password sea correcto, enviando un mensaje de confirmación.
6.- El sistema presenta un menú que contiene las opciones siguientes:
    a) Verificar pago
    b) Asignar estatus de admisión
    c) Consultas
    d) Correo
    e) Archivo
7.- El coordinador selecciona una opción

FLUJO EXCEPCIONAL:

1.-En caso que el password sea incorrecto, el sistema elimina los datos introducidos y pide que sean introducidos nuevamente.

Figura 2.3: Escenario.

NOTA:

Se utiliza para escribir algún tipo de información que describa claramente al sistema en cuestión (ver figura 2.4).

Figura 2.4: Nota.

2.2.2 Diagramas de Clase

Los diagramas de clases [2] se encuentran en el modelado de los sistemas orientados a objetos. Un diagrama de clases muestra a un conjunto de clases y sus relaciones.

Antes de realizar un diagrama de clases, se debe realizar un análisis de estas clases, o sea definir que clases son las que estarán involucradas en nuestro sistema.

Las clases del análisis representan una abstracción de una o varias clases y/o subsistemas en el diseño de un sistema. Se enfoca en el manejo de los requerimientos funcionales. Existen tres tipos de clases de acuerdo a las tareas que éstas desempeñarán en el sistema. Se conocen como clases límite, entidad o de control. A continuación explicaremos su significado.

Clase límite:

Es utilizada para modelar una interacción entre el sistema y sus actores. La interacción a menudo involucra que los usuarios y sistemas externos reciban y soliciten información.

Clase entidad:

Es utilizada para modelar información que es de larga vida y a menudo persistente, modelan información y asocian la conducta de algún fenómeno o concepto tal como un objeto o evento individual o de la vida real.

Clase de control:

Representa coordinación, secuencia, transacción y control de otros objetos y son a menudo utilizado para encapsular control relacionado a un caso de uso específico (ver figura 2.5).

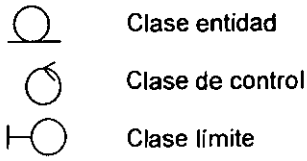


Figura 2.5: Tipos de clases.

Después de realizar esta identificación de clases se procede a realizar un mapeo de estos símbolos por el de un rectángulo con cuatro secciones. Como se muestra en la figura 2.6.

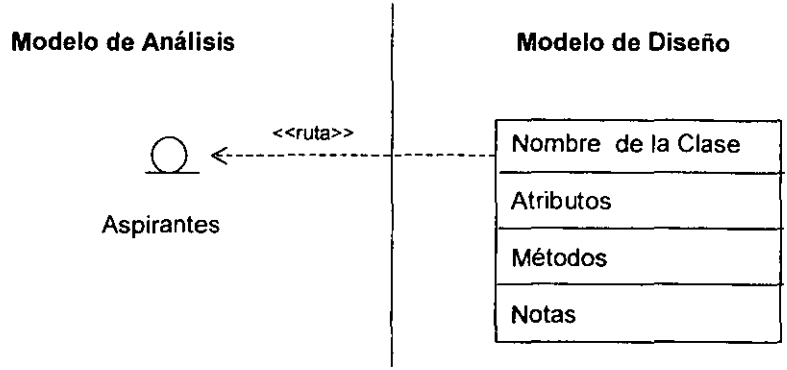


Figura 2.6: Pasando del modelo de análisis al de diseño.

En el diagrama de clases se identifican los nombres de las clases, sus atributos y métodos así como las notas pertinentes sobre la clase. Como se mencionó al principio de esta sección, el diagrama de clase también nos muestra la relación entre clases. La figura 2.7 muestra un ejemplo de un diagrama de clases.

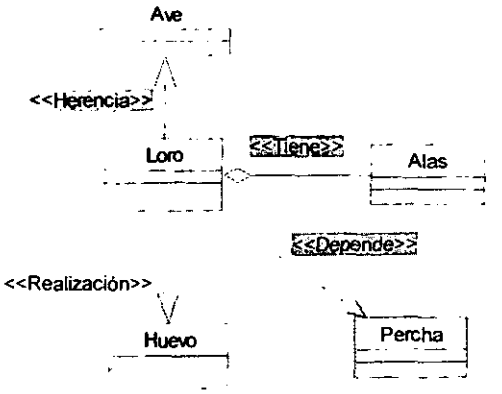


Figura 2.7: Diagrama de clases.

2.2.3 Diagramas de Secuencia

Un diagrama de secuencia [2] enfatiza en el orden de tiempo en que se ejecutan los métodos por medio de los mensajes a los objetos ya instanciados. La figura 2.8 muestra la secuencia en el tiempo en que los objetos participan en la interacción. A lo largo del eje X se presentan a los objetos involucrados, donde el objeto más a la izquierda es el objeto principal y los objetos más a la derecha son los subordinados, en el eje Y se tienen los mensajes transmitidos a los objetos subordinados y el tiempo de existencia en el ciclo de ejecución.

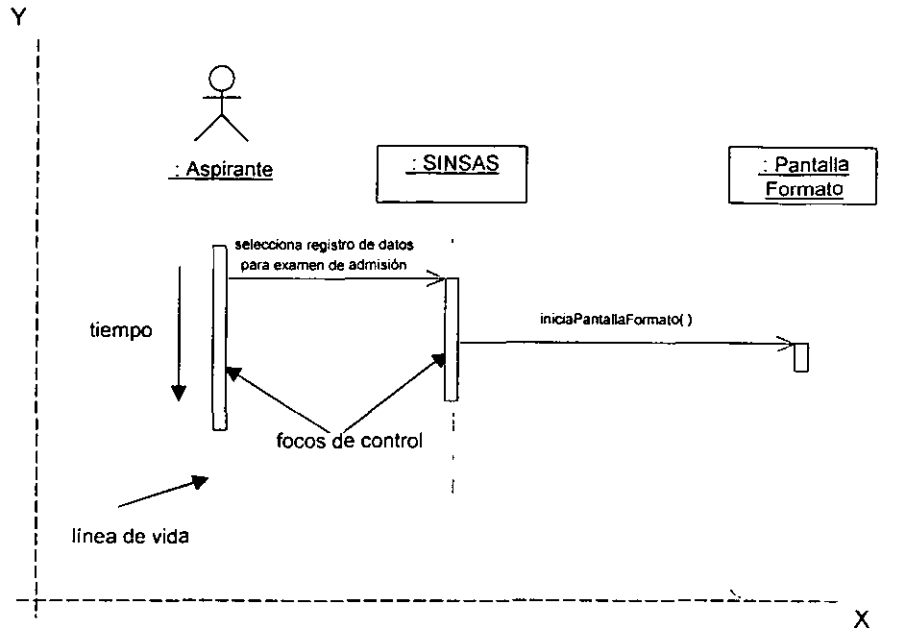


Figura 2.8: Diagrama de secuencia.

El diagrama anterior muestra cómo el usuario Aspirante accede al sistema, al seleccionar el registro de datos, el sistema es a su vez un objeto (:SINSAS) instanciado, que le manda un mensaje al objeto PantallaFormato para que despliegue la pantalla de captura de los datos del Aspirante.

2.2.4 Aplicación del Lenguaje de Modelado Unificado

Como se mencionó anteriormente UML es una gran herramienta para visualizar las partes estáticas y dinámicas de un sistema de software. Ahora debemos

concentrarnos sobre cuáles de estas partes son las que modelamos en nuestro caso de estudio y en el siguiente capitulo las abordaremos y relacionaremos con nuestro sistema con más detalle. Como comentario final podemos decir que, para modelar la parte estática del sistema se utilizaron los diagramas de clase y para la parte dinámica los diagramas de casos de uso y secuencia.

2.3 Los lenguajes y las herramientas de desarrollo

Se han tocado anteriormente otros puntos importantes para el desarrollo de sistemas orientados a objetos, pero éste es igual de importante ya que al final el lenguaje y las herramientas utilizadas crearán el producto final. Con esto es muy importante la aprobación o desaprobación del lenguaje y las herramientas de desarrollo por parte del cliente desde el inicio del proyecto. Con la llegada de las tecnologías orientadas a objetos, aparecieron varios lenguajes orientados a objetos, entre estos se encuentran: C++, Object Pascal, SmalkTalk, Java.

2.3.1 El lenguaje de desarrollo utilizado

Para el desarrollo de nuestro sistema se utilizó al lenguaje Java como el lenguaje de implentación. Ya que éste lenguaje es orientado a objetos cien por ciento y no un híbrido como C++. Además que tiene características de compatibilidad con las tecnologías WEB. Actualmente todos los visualizadores o navegadores de Internet incluyen a la maquina virtual Java. Éste es un punto muy importante en

nuestro trabajo porque nuestro sistema a desarrollar tenia que estar disponible en Internet.

Las versiones utilizadas de éste lenguaje para nuestro sistema fueron: la versión 1.2 de Java Sun Microsystem y Visual Café versión 2.0 de Symantec, esta última como herramienta de desarrollo de las interfaces de usuario.

2.3.2 La base de datos

Además de los lenguajes de programación se tienen que utilizar otras herramientas, como son los manejadores de base de datos. Existen muchos manejadores de base de datos, entre los más destacados en el mercado se encuentran:

Informix.- Utilizado para pequeñas y medianas bases de datos.

Oracle8.- Base de datos relacional utilizada para grandes sistemas.

Microsoft SQL Server.- Para utilizar en aplicaciones muy pequeñas.

Sybase System 20/11.- Utilizada en aplicaciones de medianas a grandes.

Sybase SQL Anywhere.- Es una base de datos que no necesita ejecutarse en un servidor.

En nuestro sistema se utilizó la base de datos Oracle8. Esta es una base de datos relacional. Tiene una gran aceptación en el mercado mundial, está bien probada y cuenta con un gran soporte técnico. Corre en equipos de cómputo con tecnología

RISC, además es usada para grandes bases de datos. Lo anterior nos da la confianza de usarla en nuestro sistema.

2.3.3 Las plataformas de desarrollo

La plataforma de desarrollo es de suma importancia y se tiene que escoger la adecuada. De acuerdo a los requerimientos, presupuestos y los equipos donde se pondrá en operación el sistema. Entre las plataformas más conocidas se encuentran:

- Windows NT.
- Solaris Sun
- Linux

En realidad existen pocas plataformas en nuestro medio para el desarrollo de sistemas y dentro de estas, Windows NT y Solaris presentan el mejor soporte en cuanto a software de desarrollo.

En cuanto a los equipos en donde se ejecutaría el sistema, se decidió que estos equipos fueran tecnologías RISC, como lo son los servidores y terminales Sparc de Sun.

La plataforma en la que se implementaría nuestro sistema fue el sistema operativo Solaris Sun. Sistema operativo multiusuario y multiproceso tipo Unix, ofreciendo con esto todas las ventajas de seguridad y eficiencia de los sistemas Unix.

2.3.4 Controladores nativos y de ODBC

Existen dos tipos de controladores de base de datos, los nativos y los OBDC. La diferencia entre ellos es que los nativos son controladores hechos especificamente para una determinada base de datos, obteniendo la ventaja de ser más rápidos. Los otros funcionan para cualquier base de datos y pueden ser más lentos a la hora de hacer accesos a la base de datos. Las bases de datos de gran volumen normalmente utilizan controladores nativos y se ejecutan en máquinas UNIX o mainframe. Los controladores ODBC son más flexibles respecto a las bases de datos que pueden manejar y también son menos costosos que sus contrapartes. El controlador es la conexión existente entre la aplicación creada en un lenguaje de programación determinado y la base de datos (ver figura 2.9).

En el proyecto se decidió la utilización de un controlador nativo incluido dentro del paquete Oracle8, ya que éste fue el único disponible para la plataforma elegida como lo fue el sistema operativo Solaris Sun. No fue posible conseguir un controlador ODBC que opere en equipos de computo con tecnología RISC.

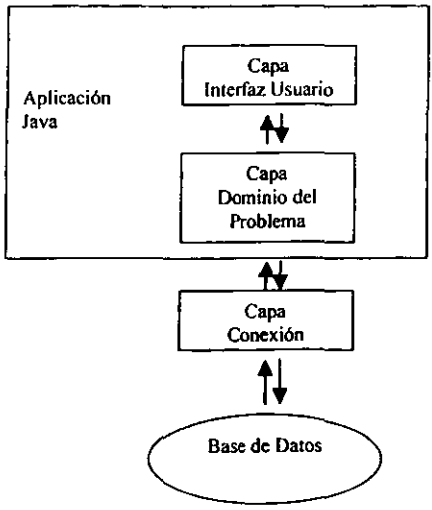


Figura 2.9: Conexión con la base de datos.

2.3.5 Otras Herramientas de desarrollo

Además de las herramientas anteriormente vistas, se necesitan otras de propósitos especiales como lo son las herramientas de administración de proyectos, las de diseño, y los procesadores de texto. Las siguientes fueron utilizadas en nuestro desarrollo:

Rational Rose.- Software para el diseño orientado a objetos.

Microsoft Project.- Herramienta para la administración de proyectos.

Editores de texto Word, Vi, etc.

2.4 El proceso unificado de desarrollo de software

Un proceso de producción de sistemas de software, es un conjunto de actividades necesarias para transformar los requerimientos de los usuarios en un sistema de software.

Un proceso de desarrollo unificado de software es un conjunto de fases y etapas en las cuales se realizan diferentes tareas sistemáticas, mediante diversos métodos que tienen como meta final la obtención de productos de software.

El proceso de desarrollo unificado utiliza al Lenguaje de Modelado Unificado (UML), cuando se realizan las tareas de definir los requerimientos, analizar y diseñar a los elementos de un sistema de software.

El proceso de desarrollo unificado tiene dos aspectos que lo distinguen:

- El manejo de los casos de uso.
- La arquitectura concéntrica, iterativa e incremental.

La figura 2.10 nos muestra de manera sencilla al proceso de desarrollo de software.

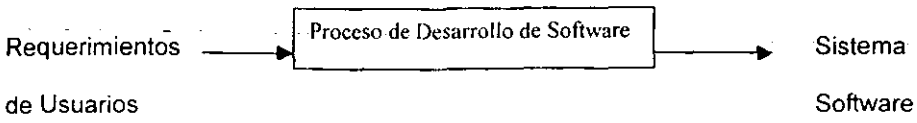


Figura 2.10: Proceso de desarrollo de software.

El proceso de desarrollo unificado presenta un esquema de dos dimensiones, en una dimensión se presentan a las fases: inicio, elaboración, construcción y transición. En la otra dimensión presentan a las cinco etapas: requerimientos, análisis, diseño, implementación, pruebas. La dimensión de las fases marca la diferencia entre los procesos de desarrollo de métodos anteriores. Los que presentan sólo a la dimensión de las etapas. De esta manera cada fase tiene a su vez cinco etapas, las que se tienen que completar para poder pasar a la siguiente fase (ver figura 2.11) [1].

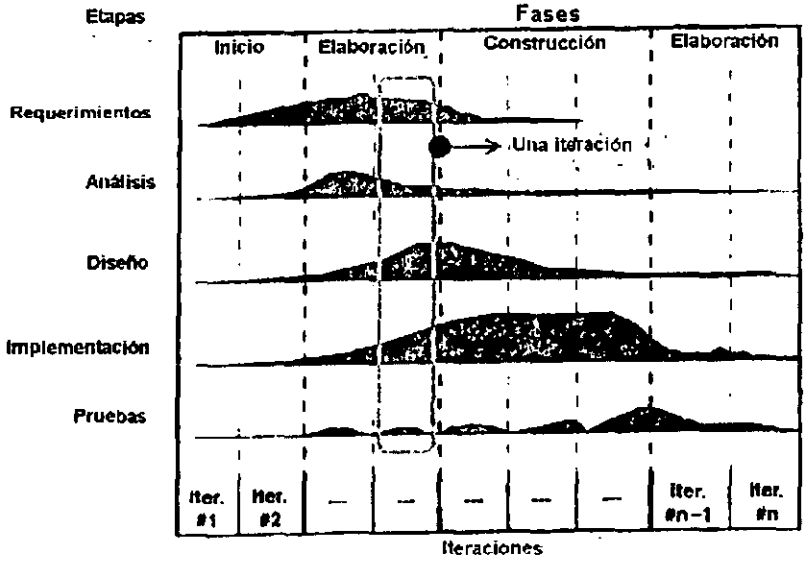


Figura 2.11: Etapas y fases del proceso de desarrollo unificado [1].

Capítulo 3

Formación del equipo de desarrollo del proyecto

La formación del equipo consiste en integrar un grupo de gente seleccionado de entre un número de individuos, quienes tienen un objetivo en común. Los cuales tengan características o cualidades de acuerdo al trabajo (rol) que deberían desempeñar, o sea construir un equipo con el mejor papel para cada individuo y con responsabilidades divididas.

Además se deben conocer otros conceptos importantes para poder organizar a un equipo.

3.1 Conceptos para la organización del equipo de trabajo

La organización del equipo de trabajo para un proyecto consiste en la integración de un conjunto de personas. Mediante una selección de acuerdo a sus habilidades

individuales. Cada individuo deberá tener características acordes al trabajo que desempeñará de acuerdo a su rol.

Para entender mejor a la organización de equipos de trabajo debemos definir lo que es un equipo, como construirlo y controlarlo.

- Para formar un equipo hay que estimar la complejidad del problema, pero se estima que deben ser entre 2 y 8 personas. Si se requieren más de 8 personas es mejor tener niveles jerárquicos de varios equipos pequeños para mantener el control.
- El tamaño del equipo depende de lo complejo del software y de la asignación apropiada de tareas.
- Un equipo pequeño debe producir el análisis de los requerimientos y el diseño.
- De ser necesario se debe agregar gente para la implementación en la fase de construcción.

3.1.1 ¿Qué es un equipo?

Hay muchas definiciones de lo que es un equipo pero una de las mejores es la de Dyer [6]. El dice que un equipo consiste de al menos dos personas, quienes trabajan hacia objetivos comunes, donde cada persona tiene asignado un rol o función y para completar la misión se requiere de la integración de los esfuerzos independientes realizados por cada miembro del equipo de trabajo.

3.1.2 ¿Cómo construir un equipo efectivo de trabajo?

Para construir un equipo efectivo de trabajo [6], éste debe tener características que indiquen un gran ánimo y un gusto por los grandes retos, para realizar el plan de trabajo y las tareas propias de éste. El equipo deberá tener una buena cohesión esto quiere decir que exista una integración armoniosa entre los elementos del equipo para que cada objetivo del proyecto se vea como un reto a vencer por el equipo en su conjunto. Retroalimentandose las veces que sea necesario de acuerdo a los objetivos y su alcance. Se deberá estimular un ambiente de trabajo común, haciendo sentir a cada individuo, que su rol tiene una igual importancia para el proyecto. Para que ellos se cuestionen sobre lo que deben conocer como:

- ¿Qué tareas debo hacer?
- ¿Cuándo?
- ¿En que orden?

3.1.3 Tipos de control en los equipos

Centralizado:

El jefe del equipo es responsable del diseño y detalles técnicos.

Es útil cuando se requieren resultados rápidos y el problema está bien entendido.

La desventaja es que hay un sólo responsable de todo, el jefe.

Descentralizado:

Se toman las decisiones por consenso, existe más satisfacción entre los miembros y se incentiva el trabajo.

Es útil en trabajos que llevan mucho tiempo, cuando se requiere mucha comunicación y para obtener una buena solución.

No es apropiado en equipos grandes.

Mixto:

Combina los beneficios de los dos enfoques.

Distingue personal senior-junior.

3.1.4 Tipos de roles en los equipos

Un rol es un tipo de trabajo con actividades relacionadas. A partir de los roles se forma un equipo de trabajo en el que se describe los roles necesarios más que a las personas que participarán. Según el proyecto una persona puede desarrollar uno o varios roles y un rol ser cubierto por una o varias personas [10].

Existen dos clases de Roles:

El rol interno.- Es aquel en el que una persona está cubriendo el rol para un sólo proyecto.

El externo.- Si lo puede hacer para varios proyectos.

Otro tipo de clasificación de los roles es de acuerdo a las actividades que realiza, y estas pueden ser administrativas o de control de proceso, de producción, de control del producto, reutilización, cliente, entre otros.

En nuestro proyecto de desarrollo se utilizaron los siguientes roles de acuerdo al Proceso Desarrollo Unificado: Administrador del Proyecto, Líder del Proyecto, Administrador de configuraciones, Administrador de Calidad, Administrador de Pruebas, Líder de construcción, Analista, Diseñador, Programador.

3.2 Integración del equipo de trabajo para el proyecto SINSAS

Para la organización del equipo de desarrollo del sistema del caso de estudio, que tuvo como nombre final SINSAS (Sistema de Inscripción de Aspirantes), se realizó lo siguiente:

Como toda organización formada de un grupo de individuos, primero tuvo que haber un conocimiento previo de los individuos. Este acercamiento puede ser con mucha anterioridad o en el momento de la integración del equipo. En nuestro caso ya existía un conocimiento anterior de las cualidades de cada individuo, así como sus limitaciones. El equipo consistió de 9 personas cada una de las cuales tuvo uno o dos roles dentro del proyecto. Los roles se asignaron de acuerdo a las cualidades de cada miembro del equipo. Se tomó la decisión de tener un tipo de control descentralizado. Aceptando las decisiones importantes del proyecto mediante un consenso de grupo.

3.2.1 Establecimiento y asignación de roles

La definición de roles se estableció mediante una junta en la cual se les preguntó a cada uno de los miembros del equipo sobre el tipo de rol que le gustaría desempeñar, y a los miembros con gustos iguales se les invitó a que se pusieran de acuerdo para que alguno eligiera otro rol. Al final a cada miembro del equipo se le asignó uno o dos roles, siendo estos de acuerdo al proceso de desarrollo unificado. El organigrama del anexo 1, nos muestra la jerarquía de los roles dentro del proyecto.

3.3 Planeación de las actividades para el proceso de software

Dependiendo del proyecto un plan de actividades puede ser simple o complejo. Un primer plan de trabajo siempre es muy importante ya que de no hacerlo se pueden tener sorpresas, y estas pueden ser muy desagradables y costosas para el proyecto. Cuando no se planean las actividades, se puede caer en una manera poco productiva de ejecutar las actividades además que otras se pueden olvidar por completo.

Para hacer un buen plan de trabajo, éste no tiene que ser siempre complejo. El plan de trabajo podrá ser realizado por el administrador del proyecto ó en su caso por el líder del proyecto. Otra figura que deberá estar involucrada en la planeación de las actividades es el administrador de configuraciones.

El plan de trabajo permite administrar las diferentes actividades y controlar el avance en cada una de las diferentes etapas del proyecto. Así como tomar las decisiones necesarias para el cumplimiento del mismo [6].

3.3.1 El plan del Proyecto

El objetivo del plan del proyecto es permitir la administración de las diferentes actividades. Así como controlar el avance en cada una de las diferentes etapas del proyecto tomando las decisiones necesarias para cumplimiento del mismo.

El plan del proyecto deberá contener: la definición del sistema, análisis de requerimientos, un diagrama de arquitectura de alto nivel, estimación del tiempo y el esfuerzo necesario para la entrega de productos [8].

En el plan de proyecto del sistema SINSAS, se realizó con base en el proceso de desarrollo unificado, dividiendo el plan de acuerdo a las fases de éste proceso.

Fase de inicio:

Contiene ocho tareas principales, integración del equipo de trabajo, establecimiento y asignación de roles, elaboración del plan de proyecto, identificación del problema, definición y especificación de requerimientos, elaboración del modelo de casos de uso nivel general y usuario, realización de un prototipo de interfaces de usuario, estimación de esfuerzo costo.

Fase de elaboración:

Contiene el análisis de requerimientos, el diseño del sistema, diseño de paquetes nivel general y el plan de riesgos.

Fase de construcción:

En donde se efectúa las tareas de actualizar los requerimientos del sistema, el refinamiento de la interfaz de usuario, la codificación e implementación del plan de construcción.

Fase de transición:

En donde se realizan las tareas de prueba del sistema, documentación del sistema, liberación, capacitación y el plan de pruebas.

Además de la especificación de tareas a realizar el plan de proyecto contiene el esfuerzo a realizar para cada tarea expresado en días, su fecha de inicio y término. La figura 3.1 presenta los porcentajes de esfuerzo a realizar para el proyecto por cada fase.

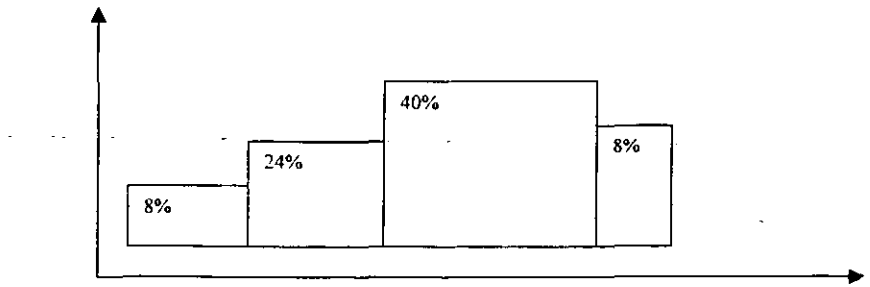


Figura 3.1: Esfuerzo.

El sistema al que se hace mención fue un proyecto de curso, por lo tanto no tuvo que ser estimado económicamente, y no se calculó un precio por el producto final. Para la realización de éste plan se utilizó el paquete de Microsoft Proyect. La figura 3.2 muestra un segmento del plan de proyecto, que utiliza una gráfica de Gatt correspondiente a la fase de inicio. Esta gráfica se realizó con el paquete Microsoft ProJect.

Proyecto: SINSAS

Fase: Inicio

Fecha: 13/sep/99

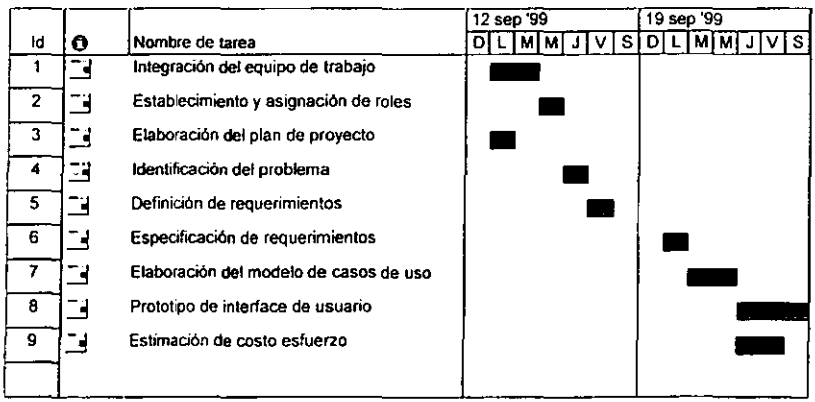


Figura 3.2: Segmento del plan de proyecto.

3.4 Ejecución de las tareas por roles

Después de tener un plan de proyecto, se deben de tener planes individuales. Estos planes deben de ir acordes al rol desempeñado. En estos planes se debe expresar con claridad las tareas que debe desempeñar cada rol y por ende cada miembro del equipo.

Es importante llevar un control de estas actividades y dentro del proceso de desarrollo del proyecto se utilizó un formato de registro. Éste se muestra en el anexo 2.

3.4.1 Objetivos y tareas por roles

Administrador de proyecto

Objetivo:

Es la planeación y control al proyecto, debe mediar en la comunicación entre el equipo técnico y el mundo exterior. Que puede ser el cliente, el usuario final y la administración general de la empresa.

Tareas:

Su trabajo se centra principalmente en hacer la planeación, las fases de definición y análisis en las que es intermediario entre el cliente y los técnicos. En las fases mas técnicas de diseño, programación, integración y prueba, casi no interviene pero las supervisa. Vuelve a ser relevante su trabajo para la aceptación del producto por parte del cliente.

Administrador de configuraciones

Objetivo:

Establecer y mantener la integridad de los productos de software durante el ciclo de vida del sistema.

Tareas:

Planear las actividades de la administración de configuraciones. seleccionar los producto de software para su identificación. Identificar los cambios en los productos software, evaluarlos y controlarlos. Informar al equipo de desarrollo de los cambios, el estatus y los contenidos de las líneas base [9].

Administrador de calidad

Objetivo:

La revisión de calidad (QA) es una actividad fuera del contexto de un proyecto, mas bien es cuestión de estándares de calidad de la empresa desarrolladora quien certifica que se cumplieron dichos estándares en el sistema que se desarrolló.

Tareas:

El encargado de la revisión supervisa todo el proceso desde el análisis, diseño e implementación. Se busca que los entregables cumplan con los estándares y la documentación sea consistente.

Se debe revisar tanto el proceso de producción como al producto. Se miden todos los productos y los prototipos.

Administrador de pruebas

Objetivo:

Se encarga de probar el código.

Tareas:

Genera casos de prueba y escenarios de prueba en conjunto con el cliente. El diseño de estos casos de prueba se hace junto con la codificación pero la prueba se hace hasta que el código se encuentra listo. La prueba se va haciendo en paralelo a la codificación.

Líder de proyecto

Objetivo:

Su objetivo es producir un sistema de calidad. Este rol tiene varios nombres en la bibliografía pero con bastantes similitudes, líder de proyectos, arquitecto de proyecto, líder de grupo, arquitecto de sistema.

Tareas:

Es alguien que lideréa el aspecto técnico y el proceso de producción, selecciona al personal que formará parte del equipo de desarrollo, los motiva y les asigna sus tareas. Es responsable de la organización moral del equipo.

Líder de construcción

Objetivo:

Administrar la fase de construcción, estableciendo las lineas base de esta fase.

Tareas:

Integra las estrategias en el desarrollo del sistema. Supervisa la Implementación de las clases, la interfase de usuario. Realiza la integración incremental del sistema así como la integración final. Documenta al proceso y las actualizaciones de los productos generados. Define y controla los riesgos existentes, asignando responsabilidades a cada miembro del equipo de construcción. Establece el calendario de trabajo y controla los cambios solicitados, los documenta y los registra. Distribuye las cargas de trabajo según el rol de cada miembro [8].

Analista

Objetivo:

Su objetivo es construir un modelo del sistema.

Tareas:

Este rol es parte del personal técnico que tiene mucha interacción con los usuarios, para lo cual se entrevista con los usuarios, revisa los documentos del problema.

Son desarrolladores experimentados. Puede construir un prototipo rápido que le sirva para comunicar al usuario una versión ejecutable del sistema, para lo cual utiliza herramientas apropiadas. No se fija en caracteristicas de eficiencia, rapidez pues luego se deshecha el prototipo.

Diseñador

Objetivo:

A éste rol también se le conoce con el nombre de arquitecto del sistema.

Su objetivo es entender las necesidades del cliente y definir la estructura del sistema.

Tareas:

Controla la evolución a largo plazo del sistema. Organiza las necesidades en categorías conceptuales. Entiende la dinámica de la arquitectura.

Es un buen desarrollador, propone modelos que deben cumplir con los requerimientos de calidad del proyecto. Puede construir un prototipo para asegurarse que su diseño puede construirse.

Debe ser un buen documentador para explicar su modelo al resto del equipo técnico y al ususario.

Programador

Objetivo:

Recibe un diseño a nivel medio y lo codifica en algún lenguaje de programación.

Tareas:

Programa el sistema, planea su prueba, la documenta y realiza la prueba.

Cada programador debe ser dueño del código que produce. Participa en el proceso de integración incorporando su código.

Capítulo 4

Proceso de desarrollo del proyecto SINSAS

En los capítulos anteriores se describió al cuerpo de conocimiento necesario para el proceso de desarrollo unificado de software con tecnología orientada a objetos. Ahora mencionaremos como surgió la necesidad de desarrollar el proyecto de software SINSAS.

Éste proyecto surgió al estar cursando la materia de Ingeniería de Software Orientada a Objetos I, misma que se encuentra dentro del plan de estudios de la Maestría en Ciencia e Ingeniería de la Computación, impartida en el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, perteneciente a la Universidad Nacional Autónoma de México y dentro del ciclo escolar 99-II.

La idea para el desarrollo del plan de la materia, era la de encontrar una necesidad real para un sistema de software, el cual se desarrollaría por un equipo de ocho alumnos aplicando la tecnología orientada a objetos estudiada. El problema que se encontró era el de la inscripción de aspirantes al examen de admisión del posgrado en Ciencia e Ingeniería de la Computación ya mencionado.

La idea general era la de un sistema que permitiera a cualquier persona que cubriera los requisitos. Inscribirse al examen de admisión de éste posgrado a través Internet.

El uso de la tecnología de Internet nos llevó a utilizar páginas WEB (archivos HTML), mismas que posteriormente serian montadas en un servidor WEB. Estas páginas, que a pesar que no se necesitaron modelar de manera orientada a objetos, no así sus ligas a applets que son archivos que ejecutan aplicaciones dentro de una página WEB escritos en el lenguaje Java. Ahora bien el lenguaje Java [4] es un lenguaje cien por ciento orientado a objetos, entonces el sistema a desarrollar era el idóneo, por las características del lenguaje a utilizar. En la sección 1.4 se menciona con más detalle las características del lenguaje Java.

A continuación se describirá el proceso de desarrollo utilizado en el caso de estudio, partiendo desde la fase de inicio.

4.1 La fase de inicio

Durante la fase de inicio, se definirá la factibilidad del proyecto: cuanto costará y cuanto redituará. Se deberá especificar los requerimientos y realizar el análisis del sistema. Con el fin de tener una idea clara del alcance del proyecto.

En nuestro proyecto no se hizo un análisis de costo. Por que éste era un proyecto de escuela y para nuestro propio instituto.

4.1.1 Los requerimientos

En los requerimientos se incluyó: la identificación del problema, la definición de requerimientos y la especificación de requerimientos.

La identificación del problema:

El problema en general consistía en que no se tenía un sistema de inscripción de aspirantes al posgrado que opere en Internet. Por lo que la solución era la de desarrollar un sistema de software que solucionará éste problema.

Definición de requerimientos:

En la definición de requerimientos se realizarón cuatro entrevistas con el cliente (coordinador del posgrado) y siete con el usuario final (personal auxiliar del posgrado). Con el cliente se definieron los requerimientos mediante entrevistas grabadas para tener un primer entendimiento del sistema. El cliente planteó lo que el sistema debería de realizar para solucionar su problema. A estas reuniones, asistieron el administrador de proyectos, el líder de proyecto, el analista y el diseñador.

Especificación de requerimientos:

Después de la definición de requerimientos el líder del proyecto le entregó esta al analista. En conjunto con el diseñador y el programador realizaron la especificación de requerimientos. La especificación de requerimientos es un

documento que describe con detalle a los elementos del sistema, sus características y procesos internos del mismo. A continuación se muestra un segmento de la especificación de requerimientos y en el anexo 3 se muestra a estos de manera completa.

Especificación de requerimientos

Versión 1.0

Sistema "SINSAS"

La Coordinación de la Maestría en Ciencia e Ingeniería de la Computación de la UNAM, con sede en el IIMAS, requiere un sistema de inscripción de aspirantes al examen de admisión a Maestría y Doctorado a través de Internet. Este sistema contará con dos tipos de acceso; en el primero tendrán acceso los aspirantes para ingresar sus datos personales y en el segundo tendrá acceso la Coordinadora del Posgrado, para consultar y actualizar información relacionada con el exámen de admisión.

En esta página electrónica los aspirantes ingresarán: nombre, dirección, ciudad, país, código postal), Dirección electrónica, teléfono (clave del país y clave de la ciudad), universidad de procedencia, si es titulado o pasante, si quiere entrar a la maestría o al doctorado y el número de ficha de pago, la fecha de pago en el banco; la fecha de inscripción al exámen de admisión la asignará el sistema en forma automática.

4.1.2 El análisis

Además de los requerimientos del sistema, en esta fase se tiene que realizar un análisis del proyecto. Mediante el modelo de casos de uso que a su vez contiene a los diagramas de casos de uso y los escenarios de los casos de uso.

Los productos que se generaron durante esta etapa fueron: diagramas de casos de uso nivel general y los escenarios de los casos de uso. El diagrama de casos de uso nivel general muestra el flujo de control del sistema con el usuario en su totalidad. Con el fin de ser más claros, la presente tesis sólo ejemplificará los requerimientos que involucren al usuario aspirante. Para la realización de estos diagramas se utilizó la herramienta Rational Rose 98.

Diagrama de casos de uso nivel general :

La figura 4.1 presenta el diagrama de casos de uso nivel general, que involucra al usuario aspirante.

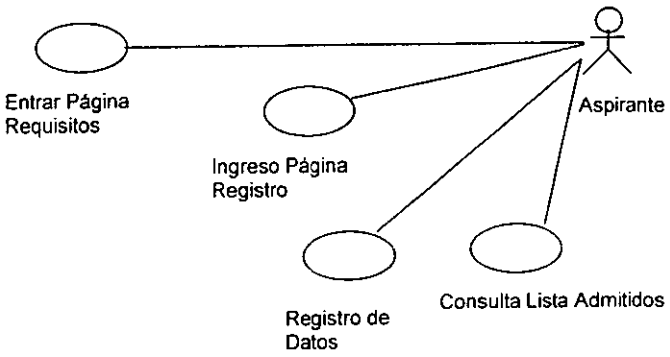
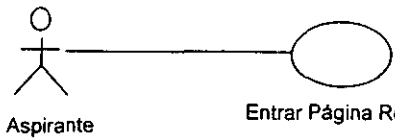


Figura 4.1: Diagrama de casos de uso nivel general.

Escenarios de los casos de uso:

Los escenarios de los casos de uso especifican con mayor claridad las acciones que el usuario debe realizar para poder interactuar con el sistema. La figura 4.2 presenta el escenario para el caso de uso entrar página de requisitos.



1.-El aspirante accesa a la página del sistema SINSAS
2.-El sistema presenta información relacionada a los requisitos y procedimientos para presentar el examen de admisión.

Figura 4.2: Escenario del caso de uso entrar página de requisitos.

4.1.3 Los procedimientos en el proceso de desarrollo de software

Los procedimientos son las normas, reglas y reglamentos que deben ser llevadas a cabo, para que las tareas de un proyecto sean realizadas con éxito. En el proyecto SINSAS se escribieron y documentaron los procedimientos necesarios

para cada realizar cada tarea. Y poder así evaluar si las tareas en el proyecto eran llevadas a cabo de acuerdo a su procedimiento. Para ejemplificar lo mencionado anteriormente expondré a continuación el procedimiento realizado por la mesa de control de cambios.

La mesa de control de cambios:

> En la fase de inicio se debe establecer la mesa de control de cambios. Esta mesa se formó con la representación de los administradores de proyecto, configuraciones, calidad así como líder de proyecto. Su tarea principal era revisar las solicitudes de cambios y en su caso autorizar los cambios. Otra de las tareas realizadas por la mesa de control de cambios es la aceptación de las nuevas versiones de los productos generados. Las cuales son registradas y resguardadas por la administración de configuraciones.

Para solicitar un cambio a la mesa de control de cambio, se utilizó el formato mostrado en el anexo 4.

Éste formato contiene tiene cuatro apartados: el que solicita el cambio, la mesa de control, implementador y control de calidad. Cada apartado es importante en cada uno se escriben datos importantes que se registran y archivan. Estos datos son del cambio que se quiere realizar y de las personas que lo ejecutarán y supervisarán.

Cabe mencionar que además de tener una mesa de control de cambios como ya se mencionó. También se implantó un proceso de revisión por pares, el que consistió en que las tareas que realizaba un miembro del equipo de trabajo, eran

revisadas por su pareja. Esta pareja fue permanente durante todo el proceso y estuvo registrada en un documento archivo por la administración de configuraciones.

Los procedimientos de la mesa de control de cambios:

La mesa de control tiene que estar sustentada mediante procedimientos que especifiquen la manera de cómo se deben realizar estos. Por ejemplo, el procedimiento de una solicitud de cambio, debe estar avalado mediante un documento que especifique los pasos para realizar con éxito éste proceso. El anexo 5 presenta el procedimiento de solicitud de cambio que se implantó en nuestro proceso de desarrollo de software.

Control de calidad:

Para el control de calidad de nuestro proceso de desarrollo se nos pidió llevar un modelo de control de calidad. Éste modelo fue CMM (The Capability Maturity model), CMM presenta cinco niveles para obtener la máxima calidad en el proceso. En nuestro desarrollo se nos pidió llevar nuestro proceso al nivel dos. CMM en su nivel dos nos indica que todos nuestros procedimientos deben de estar documentados. Para cada una de las tareas que realicen los roles involucradas en el proceso de desarrollo de software. Esto con el fin de tener un control de las tareas y responsabilidades de cada miembro del equipo de desarrollo. El presente trabajo no tratará de profundizar en el tema ya que éste es todo un campo de estudio [9].

Como comentario final, en éste apartado describimos la forma de cómo realizamos nuestros procedimientos y los documentamos, utilizando un modelo de calidad. Aunque no se profundizó en el modelo CMM creo que sirvió de referencia para la implementación de los procedimientos en nuestro proceso de desarrollo de software.

4.1.4 Documentación, respaldo y resguardo del proceso de desarrollo

En la fase de inicio se deben de realizar los procedimientos para la documentación del proyecto. La documentación del proyecto consiste en la generación de productos y documentos generados durante el tiempo que el proyecto duró. Asi como el respaldo y resguardo estos. Se debe documentar desde la fase de inicio hasta la fase de transición. Es responsabilidad del administrador de configuraciones el identificar, clasificar, ordenar, respaldar y resguardar los productos y documentos generados por el proyecto. Por ejemplo se debe resguardar el documento que contenga el procedimiento y se debe respaldar y resguardar el producto como lo es un archivo del código fuente del sistema de software, en un medio magnético. Es importante destacar que la documentación no sólo son los manuales del sistema de software, estos son una parte de la documentación del proyecto.

4.2 La fase de Elaboración

En esta fase ya se especificaron los requerimientos, pero hace falta comprender mejor el problema.

- ¿Que es lo que se va a construir en realidad?
- ¿Cómo lo va a construir?
- ¿Qué tecnología empleará?

Las tareas principales en esta fase para nuestro proceso de desarrollo fueron:

- La planeación de la fase de elaboración la cual estuvo a cargo del administrador e incluida dentro del plan general.
- El manejo de riesgos.
- La realización del análisis y diseño.

En esta fase, ya se posee una idea de los requerimientos y lo decimos así por que los requerimientos también pueden tener cambios. Estos se pueden realizar en la fase de elaboración, utilizando una solicitud de cambio y presentándola a la mesa de control de cambios.

4.2.1 Manejo de riesgos

Es importante el manejo de riesgos en el desarrollo del proyecto. Para poder saber ¿cuáles son los factores que pueden descarrilarlo?. Si un riesgo es mayor entonces a éste deberá prestarle mayor atención.

Se pueden clasificar tres tipos de riesgos:

- Riesgos de requerimientos. La principal tarea del manejo de éste riesgos es el peligro de que se construya un sistema erróneo, un sistema que no haga lo que el usuario quiere.
- Riesgos técnicos. Corresponden al manejo de la tecnología empleada. Si se usan objetos, el lenguaje a utilizar, las herramientas de desarrollo a utilizar etc. Esto con el fin de saber si se tienen los conocimientos necesarios sobre la tecnología a utilizar.
- Riesgos de habilidad. Consiste en saber si se puede conseguir la asesoría y los expertos necesarios.
- Riesgos políticos. Consistente en manejar a las fuerzas políticas que se puedan interponer en el proyecto.

En el proyecto SINSAS, básicamente se manejaron los tres primeros tipos de riesgos que se incluyen implícitamente en el plan de riesgos presentado mas adelante.

La manera en que se manejaron estos riesgos dentro del proyecto fue mediante un reporte o informe semanal de actividades. Este lo realizaba el administrador de

proyecto con observaciones de los miembros del equipo de desarrollo y confrontado con el plan de riesgos. Para realizar el análisis de riesgos cada semana se reunía el equipo de administradores y se confrontaba al plan de actividades programadas con el avance real. De esta forma se realizó la estimación de los riesgos. El anexo 6 nos presenta al plan de riesgos utilizado en nuestro proyecto.

4.2.2 Métricas

Además del plan de riesgos se deben tener otras herramientas de evaluación de riesgos dentro del proyecto. Estas son las métricas en proceso de software que miden el progreso y el éxito del proyecto. Las métricas se aplican a la administración del proyecto y al producto de software generado [11].

La medida del progreso y del éxito comprende:

El seguimiento del presupuesto.- Valora cuándo la cantidad que se ha gastado en el proyecto, en un momento dado, está conforme lo planeado.

Desarrollo de métricas.- Están enfocadas al producto y sirven para medir el tamaño original planeado y el tamaño actual real.

Reporte del estado.- Diseñado para conocer el estado en general del proyecto, su progreso y estabilidad. El intervalo de tiempo entre cada reporte deberá ser estimado de acuerdo a los tiempos de entrega del producto así como la importancia del proyecto . Por lo general estos son reportes semanales o mensuales.

El reporte sirve para:

- Detectar y valorar tempranamente los riesgos del programa.
- Comunicar riesgos a administradores y clientes, prever soluciones.

Y deberá contener los siguientes apartados.

- Titulo de página. Nombre del proyecto, fecha, administrador del proyecto
- Novedades. Lista información de interés.
- Valoración de riesgos. Resumen visual del estado de los riesgos.
- Análisis del valor ganado. Seguimiento del presupuesto, métricas.
- Seguimiento del calendario y estabilidad. Resumen en una gráfica de Gantt
- Seguimiento del desarrollo y estabilidad. Gráficas de entregables para: requerimientos, diagramas de casos de uso, diagramas de clases etc.
- Otras métricas. Otras gráficas estratégicas como: datos del equipo de trabajo, análisis de defectos, estado de las pruebas del sistema.
- Revisión de elementos. Gráfica del estado de los elementos de acción generados durante las revisiones previas: fecha, elemento, estado, salida.

El anexo 7 nos presenta el formato para el reporte del proyecto

4.2.3 La realización del diseño

En la etapa de análisis se dió prioridad al conocimiento de los requerimientos, los conceptos y las operaciones relacionadas con el sistema. Durante el ciclo de desarrollo iterativo es posible pasar a la fase de diseño, una vez terminados estos documentos del análisis. Durante éste paso se logra una solución lógica que se funda en el paradigma orientado a objetos. Su esencia es la elaboración de diagramas de interacción, que muestran gráficamente cómo los objetos se comunicarán entre ellos a fin de cumplir con los requerimientos.

Los diagramas de interacción nos permite dibujar el diseño de clases que resumen la definición de las clases (e interfaces) implementables en el software. En la fase de construcción examinaremos la creación de estos artefactos. Los diagramas de interacción son los más importantes de ellos y exigen una gran dedicación y esfuerzo creativos. En el proyecto SINSAS se generaron básicamente los siguientes productos: diagrama de red, diagramas de clases, diagramas de interacción.

4.2.4 Diagrama de red para el sistema SINSAS

Al construir un sistema de software, se consideran dimensiones físicas y lógicas. Por un lado se encontrarán elementos físicos, tales como, las clases e interacciones y por otro los componentes. Estos representan el empaquetado físico de los elementos lógicos; y nodos, los cuales representan el hardware en el cual esos componentes se podrán en marcha y se ejecutarán.

Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, generalmente tiene memoria y capacidad de procesamiento.

Los nodos se pueden organizar especificando relaciones entre ellos. Esta organización se llama diagrama de red.

En el sistema SINSAS la relación establecida será la de asociación, la cual permitirá representar conexiones físicas entre nodos. La figura 4.3 muestra el diagrama de red del sistema SINSAS.

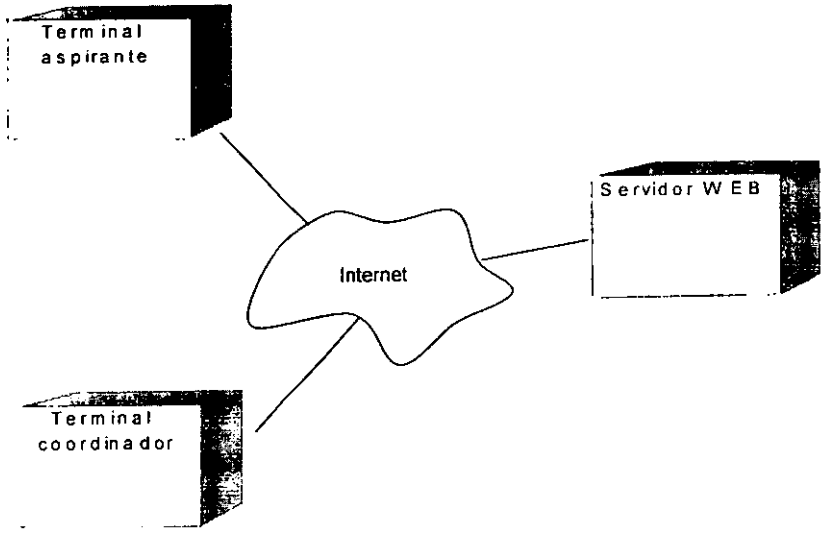


Figura 4.3: Diagrama de red.

4.2.5 Diagrama de clases del sistema SINSAS

En el capítulo 1 se definió lo que es un diagrama de clases. En el proyecto SINSAS para realizar éste diagrama primero se identificaron todas las clases que nuestro sistema tendría. Alguno de los nombres de clase fueron extraídos de la especificación de requerimientos. Para luego definir si estas clases eran clases: límite, control o entidad, véase anexo 8. El siguiente punto a realizar es la definición de los atributos y métodos de la clase. Los atributos se pueden obtener con más facilidad y pensamos en ellos como adjetivos calificativos de un objeto. Los métodos pueden ser identificados por los verbos usados en los enunciados de los requerimientos. También se pueden obtener mediante la realización previa de los diagramas de secuencia (ver diagramas de interacción). La figura 4.4 muestra el diagrama de clases del sistema SINSAS. Este diagrama presenta otras clases que no están dentro de la lista del anexo 8. Pero que fueron incluidas en el diagrama para poder implementar el código del sistema.

El diagrama de clases es un primer acercamiento a la codificación del sistema, y no tiene que representar exactamente como deberá quedar el código. Esto es por que en la codificación se presentan muchos detalles técnicos propios de los lenguajes de implementación, como se mostrará más adelante.

Por otra parte el responsable de elaborar el diagrama de clases debe tener la experiencia necesaria, para no crear clases innecesarias o por el contrario que hagan falta clases. Al final el diagrama de clases deberá contener el modelo estático que represente de la mejor manera al sistema.

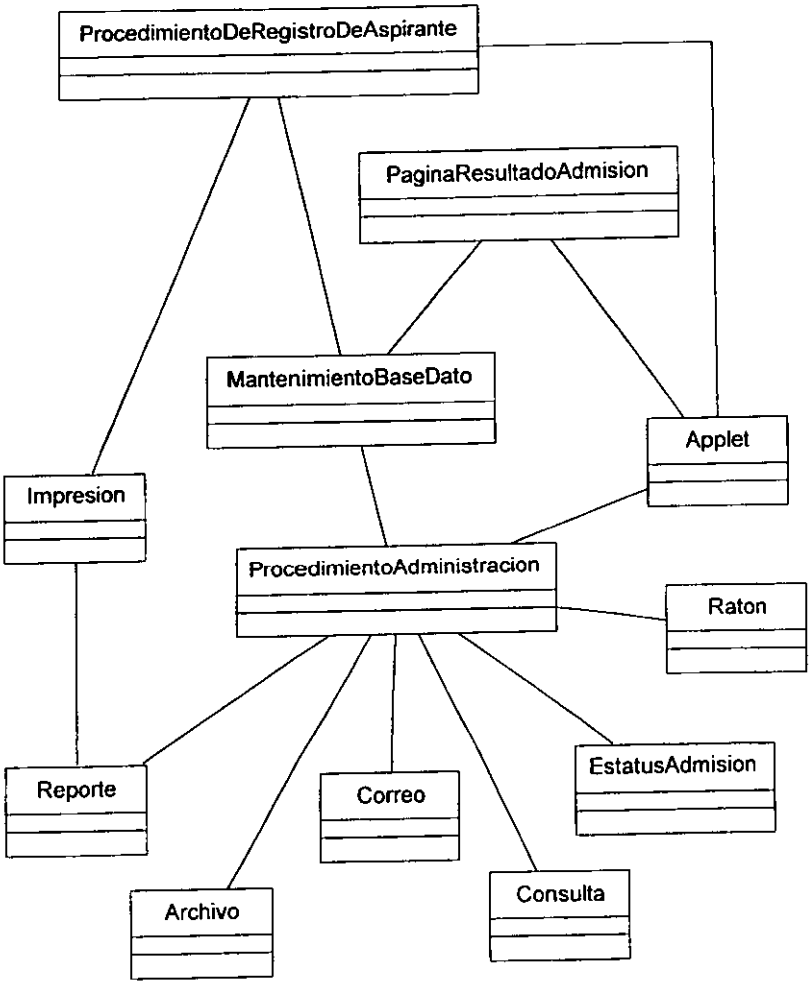
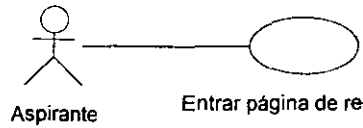


Figura 4.4: Diagrama de clases general

4.2.6 Diagramas de interacción del sistema SINSAS

En el capítulo 1 se mencionó que existen 2 tipos de diagramas dentro de los diagramas de interacción: los diagramas de secuencia y los de colaboración. En el diseño del sistema SINSAS sólo se realizaron los diagramas de secuencia, por lo que no presentaremos a los diagramas de colaboración. Los diagramas de secuencia fueron realizados a partir de la identificación de las clases y el seguimiento de los escenarios por caso de uso. Es importante mencionar que estos diagramas fueron difíciles de realizar y tuvieron que ser revisados varias veces. El anexo 9 muestra a todos los diagramas de secuencia para cada caso de uso.

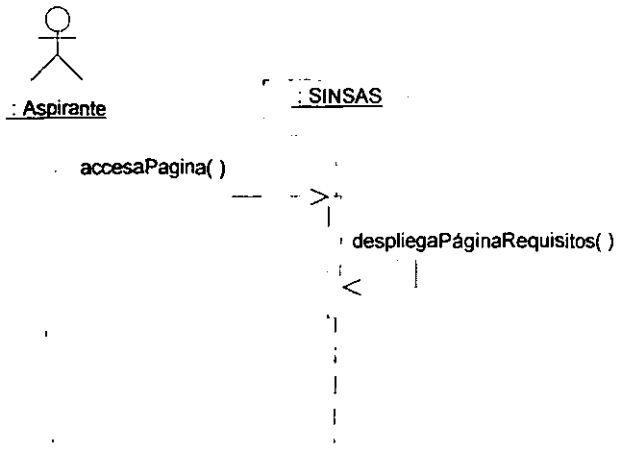
La figura 4.5 (a) muestra al diagrama de caso de uso "Entrar página requisitos" su correspondiente escenario figura 4.5 (b) y diagrama de secuencia figura 4.5 (c).



a) Caso de uso.

1. El aspirante accesa a la página del sistema SINSAS
2. El sistema presenta información relacionada a los requisitos y procedimientos necesarios para presentar el examen de admisión al posgrado. Se presentan dos alternativas:
   a) registrar datos del aspirante
   b) ver resultados de admisión

b) Escenario.



c) Secuencia.

Figura 4.5: Diagramas.

4.3 La fase de Construcción

El objetivo de la fase de construcción es la de mapear los productos del diseño, para realizar la codificación en un lenguaje. Una vez concluido los diagramas de clases del diseño y secuencia destinados al ciclo de desarrollo actual del sistema. La fase de construcción es el siguiente paso en el desarrollo del software, ya que en las fases anteriores el equipo de desarrollo se dedicó a entender el problema y encontrarle una solución. En esta fase se construye la solución por lo que antes de comenzar se debe de contar con:

- El diseño de las clases, incluyendo sus métodos

- Una vista dinámica de todos los vínculos que existen para los casos de uso
- Un completo y funcional ambiente de desarrollo

En la fase de construcción se deberá perseguir cuatro puntos:

- Planear la integración del sistema para cada iteración, por medio de una sucesión de pequeños pasos.
- Distribuir el sistema, ubicando a los archivos ejecutables en sus nodos correspondientes.
- Implementar a las clases y sus subsistemas encontradas en el diseño, en particular las clases se implementan como archivos que contienen código fuente.
- La realización de pruebas individuales así como en el momento de su integración, observando su compilación y su ligado con uno o más ejecutables.

Las actividades de la fase de construcción dentro de un ciclo de desarrollo son: implementar las definiciones de clase, implementar los métodos, implementar interfaz de usuario, implementar esquema de la base de datos, escribir código de prueba. En general la implementación en esta fase, significa el definir los atributos y métodos en clase mismos que serán especificados en la codificación.

Para ejemplificar esto, tomemos a la clase ProcedimientoDeRegistroDeAspirantes mencionada anteriormente y agreguemos otras clases necesarias para su implementación como muestra la figura 4.6.

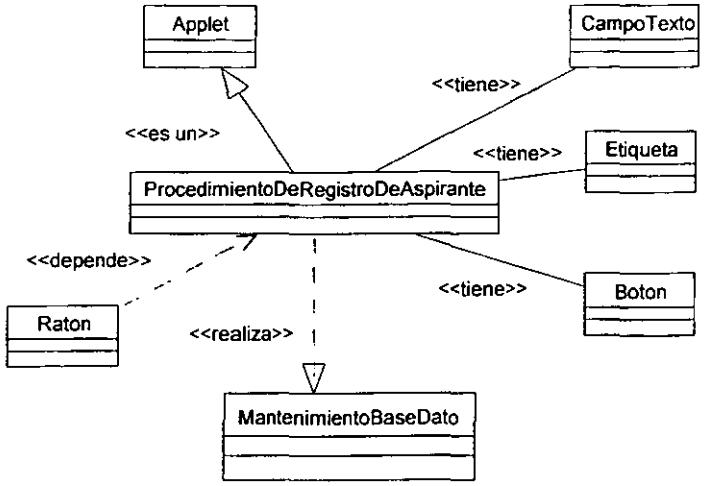


Figura 4.6: Diagrama de clases sobre ProcedimientoDeRegistroDeAspirante

Para luego poder especificar sus atributos y métodos de una manera más fácil como lo muestra la figura 4.7. Note que en esta nueva representación ya se tiene especificado los atributos y los métodos de la clase procedimientoDeRegistroDeAspirante.

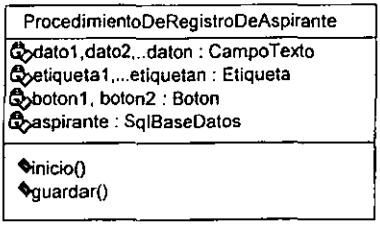


Figura 4.7: Especificación de atributos de clase.

El código fuente que se genera para esta clase se presenta en el anexo 10. Como se verá éste código contiene otros objetos no presentados en la especificación. Pero son necesarios para la codificación de la clase y no por esto nuestra definición de clase deja de ser correcta. La especificación de la clase es un acercamiento a la codificación y no se pretende que sea el código mismo.

4.3.1 Productos generados

Para el sistema SINSAS los productos generados fueron:

- Archivos ejecutables.
- Archivos que contienen código fuente.
- Páginas web, archivos html.
- Tablas de la base de datos
- Documentos: reportes, bitácoras, minutas, planes etc.

Para la realización de estos productos se utilizaron las herramientas de software mencionados en el capítulo 1.

4.3.2 Procedimientos de la fase de construcción

Esta sección plantea la metodología a seguir para la etapa de construcción, en la que se eligió un patrón de arquitectura por casos de uso. Este patrón consiste en

dividir el trabajo de codificación por casos de uso. Existe otra forma de organizar el trabajo de codificación y es utilizar un patrón por clases.

No importando cuál de los dos tipos de patrones será utilizado, el problema siempre serán las clases de control y entidad que por lo general tienen larga vida (ver sección 1.3.2). Por estar relacionadas con otras clases que implementan uno o varios casos de uso.

Dentro del proyecto SINSAS las actividades realizadas para esta fase fueron:

1. Reunión de trabajo para la elaboración del plan de trabajo y su calendarización.
2. Distribución de cargas de trabajo de acuerdo al patrón elegido.
3. Estudio por los integrantes del grupo de construcción del análisis y el diseño.
4. Inicio de la codificación del producto en forma paralela.
5. Integración de los módulos en páginas html.
6. Seguimiento, evaluación y control de los productos de acuerdo a los estándares de UML, calidad y pruebas.
7. Elaboración de bitácoras y reportes.
8. Realización de las actividades de retroalimentación iterativa e incremental.
9. Realizar pruebas a los prototipos y a la versión final.
10. Interacción con los equipos de Diseño y Pruebas, para las realizaciones de las solicitudes de cambio.
11. Respaldo de los productos generados.

12. Entrega de productos generados al líder de proyecto.

4.3.3 Planes y herramientas de control

Dentro de las tareas de la fase de construcción se tienen que elaborar los planes y las herramientas necesarias para su seguimiento. Dentro del proyecto SINSAS se elaboraron los planes de:

Validación.- Sirve para verificar que exista el apego a los estandares en el diseño realizado mediante el proceso unificado, así como las normas en la códificación.

Contingencia.- Prever de acuerdo al caso en cuestión, debiendo tomar las medidas necesarias para prevenir: fallo en el hardware, retraso en la codificación, perdida de archivos, casos no resueltos.

Comunicación.- Debe de facilitar el flujo de datos e información por parte de los integrantes del grupo de construcción.

Herramientas de control.- Se utilizaron formados que a continuación de describen.

Validación de código.- Utilizado para pruebas de escritorio del código.

Control de versiones.- Guardan y registran las versiones del código.

Bitácoras.- Registran todo suceso durante el proceso de forma continua.

Reportes.- Se involucran a todos los reportes del proyecto.

Registros de tiempos.- Estos registran informan de los tiempos de trabajo efectivo de cada integran involucrado en esta fase.

4.4 Discusión

De éste capítulo se puede discutir la existencia de una línea divisoria que nos marque el inicio y el fin entre una fase y otra ó entre las etapas. La verdad es que no la hay tal línea y decimos esto por que en el diseño se dejan clases que no tienen que estar completamente definidas en la fase de elaboración. Por ejemplo en un diagrama de clases se puede sólo nombrar a la clase y dejar la definición de los atributos y métodos para la fase de elaboración. Con lo anterior decimos que de ninguna manera se debe de pensar que lo descrito en éste capítulo debe de seguirse rígidamente.

Otra discusión importante es como podemos ir creando nuestros diagramas de clases y secuencia de una manera fácil. La experiencia nos dice que podemos sacar de los requerimientos funcionales los sustantivos que servirán para nombrar a las clases del diagrama de clases. Con los verbos de nuestros requerimientos podemos obtener nuestro diagrama de casos de uso. Y con éste diagrama

obtener los diagramas de secuencia. Con los diagramas de secuencia y diagramas de clase podemos definir los metodos y atributos de las clases.

Como comentario final de éste capítulo, se debe de tener mucho cuidado en hacer las cosas correctamente. Por que los costos de corregir los errores en las fases posteriores se multiplicarán asombrosamente.

Capítulo 5

La fase de transición del Caso de Estudio

Hasta ahora se ha llevado a cabo las tres primeras etapas del proceso de desarrollo. Durante el inicio, se estableció la razón de ser del proyecto y se determinó su alcance. En la elaboración, se reunieron los requerimientos más detallados, se hicieron análisis y diseños de alto nivel, a fin de establecer una arquitectura base, y se creó un primer plan de construcción. En la etapa de construcción se utilizó el análisis y el diseño de alto nivel para crear el software mediante las herramientas y lenguajes de implementación requeridos en la fase de inicio.

La fase de transición tiene como objetivo final la aceptación del sistema por parte del cliente, en el tiempo y presupuestos estimados. Sus actividades principales son:

- Realización de pruebas operacionales.
- Generación y disposición de los reportes de problemas.
- Priorizar y reparar los defectos.

Es importante mencionar que el mantenimiento del software no es parte del proceso de desarrollo de software. Pero en éste documento se incluye como aportación al proyecto.

Para realizar las actividades antes mencionadas se debe hacer un plan de la fase de transición. Este plan deberá contener lo que se debe hacer en esta fase:

- Establecer el responsable o responsables de la fase
- Definir los requerimientos necesarios para el logro exitoso de la fase: recursos materiales y humanos.
- Especificar que tipo de productos se deben de generar durante el proceso ejemplo: reportes de pruebas, documentos de aceptación de productos, etc.
- Crear los calendarios de actividades.
- Y otros, como pueden ser los procesos administrativos y de control.

Los planes de cada fase en general tienen los mismos puntos y sólo difieren en lo especifico, por ejemplo: en los productos generados, ya que mientras en la fase de construcción se genera código fuente, en la fase de transición se generan reportes de pruebas al código ejecutable. A pesar de que cada etapa tiene su fase de pruebas , en la fase de transición se le da mayor atención a esta actividad. Así en la etapa de transición se prueba, con más énfasis los productos generados en la fase de construcción.

En el proyecto SINSAS el administrador de pruebas se encargó de realizar las pruebas en los productos generados en la fase de construcción. Y es en estas pruebas en donde se concentraron los esfuerzos de la fase de transición.

El diagrama de la figura 5.1 representa el proceso iterativo de la fase de transición utilizado en el proyecto SINSAS. En éste diagrama se puede ver qué, para liberar la versión final del producto de software se tuvieron que realizar varias pruebas al mismo, haciendo de esta manera un ciclo iterativo e incremental

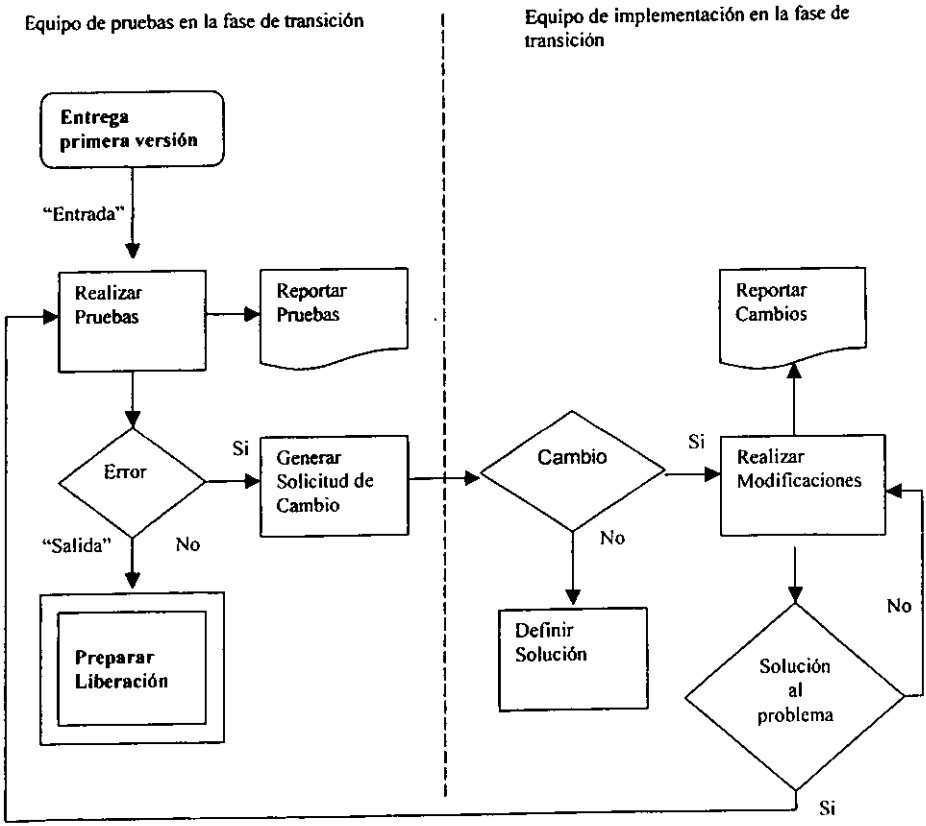


Figura 5.1: Diagrama iterativo del proceso de la fase de transición.

Tanto los casos de pruebas como el diagrama de proceso, fueron la parte fundamental de nuestra etapa de pruebas.

5.1 Realización de pruebas en el software SINSAS

La etapa de pruebas de un sistema es la parte más compleja y difícil de verificar. Por ejemplo al realizar la verificación del código es fácil perderse en todo ese mar de líneas. Y esto es porque cada persona tiene distinta forma de codificar, que viene ligado a la forma de pensar de cada persona.

En el proyecto SINSAS se realizaron pruebas a las etapas de diseño e implementación del proceso de desarrollo de software. A pesar que existen otros tipos de pruebas, estas no se realizaron en los tiempos de previstos en la entrega del producto de software [7].

Las pruebas se llevaron a cabo previa realización de un plan de pruebas, éste plan contiene:

- La estrategia para llevar a cabo las pruebas en el sistema SINSAS, basada en el uso de prueba de validación de un contexto. Se centra en las acciones visibles y las salidas reconocidas por el usuario.
- A los elementos del sistema que se probarán: requerimientos, análisis, diseño, implementación y la documentación.
- Los elementos mínimos requeridos para la aplicación de pruebas.
- El personal requerido para el proceso de pruebas dado el número de integrantes del equipo.
- El calendario del plan de pruebas.

Como se mencionó anteriormente en esta fase se hizo énfasis en las pruebas a la fase de construcción y específicamente a la etapa de implementación.

Toda prueba de la etapa de implementación tiene su base en los programas fuentes y los tipos de pruebas concernientes fueron:

- Integridad de los datos y controles implementados
- Reglas de autorización de ingreso a los datos y al sistema
- Integridad de archivos y controles implementados
- Evaluación del diseño de interfaz de usuario
- Sistema de archivos
- Procesos de seguridad
- Los programas cumplen con: metodología, diseño, es portable y mantenible
- pruebas de concurrencia

Cada tipo de prueba puede contener uno o más criterios de prueba y esto depende del alcance de cada prueba en especial. Cada criterio de prueba nos servirá para elaborar los reactivos de la prueba y estos son:

1. tiene procesos que indiquen el manejo de transacciones.
2. tiene validaciones para entradas a datos y procesos.
3. tiene controles que anticipen entradas erróneas.
4. se tiene control sobre los errores no previstos.
5. tiene procedimientos establecidos para el manejo de interfaz.
6. tiene alta cohesión y bajo acoplamiento.
7. maneja el encapsulamiento.

5.1.1 Reporte de problemas en el proceso

El reporte de los problemas del proceso es el mecanismo por el cual los probadores notifican al equipo de los problemas captados. Esto es muy importante para que los problemas sean ubicados lo más pronto posible. Estos problemas por lo general surgen al no cumplirse los requerimientos funcionales y no funcionales del sistema. Y pueden ser ordenados en las siguientes categorías:

No reproducibles.- Consiste en el reporte de los problemas que no pueden ser reproducidos en el laboratorio del equipo de desarrollo. Por ejemplo, problemas con la plataforma utilizada (sistema operativo) y programas de soporte (compiladores y librerías). Este reporte no produce un cambio en el código del sistema de software.

Reproducibles.- Estos si pueden ser reproducidos en el laboratorio y por esto son más fáciles de corregir. Por ejemplo problemas con la implementación de una clase, o los más críticos como los problemas de diseño.

El formato utilizado para el reporte de problemas en el proyecto SINSAS se muestra en la figura 5.2. Este formato nos registra con claridad: fecha, nombre, versión, detalles del problema, quien reporta el problema, quien lo soluciona. Y tres aspectos más que son importantes:

- Si el problema es o no un problema reproducible en el laboratorio
- Si es necesario trabajar con el diseño, lo que sería un problema muy serio.

- Y si es necesario realizar una solicitud de cambio.

"FORMATO DEREPORTE DE PROBLEMAS EN EL PROCESO"

Nombre del problema_____ Fecha:_____

Número de versión_____

Detalles del problema

Reportado por_____

DISPOSICIÓN

Comentarios

No reproducible

Trabajar con el diseño

Requerimiento de cambio

Eliminar problema por_____ Fecha_____

Figura 5.2: Formato de problemas en el proceso.

5.1.2 Prueba de interfaz de usuario

Hasta ahora sabemos qué elementos debemos de probar y cómo lo debemos reportar, pero no sabemos qué procedimiento se debe utilizar. En procedimiento

que se utilizó en el proyecto SINSAS para realizar la prueba de interfaz de usuario consistió en :

- Especificar el tipo de prueba, que para nuestro ejemplo es la prueba de interfaz de usuario
- Elaborar el instrumento de prueba.
- Ejecutar la prueba de acuerdo al patrón adecuado.
- Realizar el reporte de los resultados de la prueba.

Es importante mencionar que esta no fue la única prueba realizada al proyecto, ya que se realizaron otras como:: a) prueba de integridad de los datos y controles implementados, b) reglas de autorización de ingreso a los datos y al sistema, c) integridad de archivos y controles implementados [7].

Después de definir al tipo de prueba se debe de crear el instrumento de prueba. Y esto no es tarea fácil, ya que debe cubrir todos los criterios que la prueba exige. La figura 5.3 muestra el instrumento de prueba de la interfaz de usuario correspondiente al criterio: validación para las entradas a datos y procesos.

Prueba de interfaz de usuario

Criterio: Validación para las entradas a datos y procesos
Etapa: Construcción
Aplicada por: Líder de pruebas
Fecha: 01/10/1999
Hora: 10:30 a.m.
Tiempo de aplicación: 1 hra.

Marque con una x dentro del recuadro correspondiente al reactivo, si su respuesta es afirmativa a la pregunta que éste formula.

1. ¿Están los campos para el ingreso de información en un orden de acuerdo a las reglas de diseño de interfaz gráficas? ☐
2. ¿Tienen los campos etiquetas que guíen al usuario sobre el tipo de dato que solicitan? ☐
3. ¿Los campos de entrada de datos validan automáticamente las entradas evitando el ingreso de caracteres no válidos? ☐
4. ¿Las teclas o botones de control envían mensajes de error si existen datos inválidos en los campos? ☐
5. ¿Tienen los botones de control etiquetas que faciliten la identificación del proceso que inician? ☐

Comentarios:

Figura 5.3: Prueba de interfaz de usuario.

Aunque en nuestro ejemplo de prueba esta tiene un criterio con varios reactivos, se pueden diseñar pruebas más completas que contengan varios criterios con muchos reactivos cada uno de ellos.

5.1.3 Patrón de prueba

El patrón que se debe utilizar en estas pruebas es el de por casos de uso. En las que se prueba la funcionalidad del sistema y la interacción con el usuario. Seguir el patrón por clases sería indicado utilizar en otro tipo de prueba. Por ejemplo en la prueba de integridad de los datos y controles implementados. Ya que de utilizar otra (casos de uso) se corre el riesgo de pasar por alto las relaciones entre clases, los atributos de estas y los mensajes que se pasan unas a otras. Y la prueba de integridad trata de buscar errores en el manejo de datos y los datos no son otra cosas que los atributos de las clases.

Una vez diseñada el instrumento de prueba se ejecuta la prueba de acuerdo al patrón adecuado. Y se realiza el reporte de la prueba utilizando el formato de reporte de problemas junto con la prueba.

Para realizar el reporte de prueba se debe de utilizar tantos reportes de problemas como criterios fueron probados.

Es importante mencionar que los criterios pueden ser evaluados nuevamente en otro tipo de prueba, no importando que estos hayan sido probados previamente. Por ejemplo en la prueba de interfaz de usuario, se tiene que probar que las entradas realizadas por el usuario sean validas (criterio de validación para

entradas a datos y procesos). En el proyecto SINSAS se aplico éste procedimiento, así como procedimientos similares para el resto de las pruebas. Por último de existir reportes de problemas, estos deberán ser corregidos por los responsables y deberán ser probados nuevamente. Lo deseado es que no exista problemas desde la primera prueba, pero esto es poco probable. Por lo general se requieren varios ciclos de pruebas para liberar un producto. Como comentario final los procedimientos para realizar los otros tipos de prueba de esta fase se realizaron de manera similar, en el proyecto SINSAS [7].

5.2 Liberación del producto

En la fase de transición todas las pruebas que se realizaron, son el camino a la liberación del producto software. La obtención de una versión beta es el principal objetivo del proceso de desarrollo de software. Obtenida esta versión se pasa al proceso de instalación y pruebas con el usuario. En el proyecto SINSAS éste procedimiento fue el utilizado por el equipo de transición. A continuación se presentan las tareas que se realizaron en éste proceso:

- Preparación de la versión beta, obtenida mediante el proceso iterativo e incremental de la fase de transición.
- Instalación del producto en el equipo final, del cliente o usuario.
- Actividades de retroalimentación y pruebas con el cliente o usuario.
- Actividades de capacitación con el cliente y con los usuarios finales.

Hasta aquí, si todo el proceso de transición fue correcto y las pruebas con el cliente y los usuarios fueron satisfactorias. Entonces se procede a liberar la primera versión. En el proyecto SINSAS la versión final fue la 1.0, quedando con esto finalizado nuestro proceso de desarrollo unificado de software.

5.3 Mantenimiento y actualización del sistema

Los cambios o actualizaciones así como las mejoras o incorporaciones de nuevos requerimientos, se determinan como el mantenimiento del sistema y no se incluyen dentro del paquete de requerimientos establecidos en la primera versión del sistema o producto de software. El mantenimiento de sistemas de software se puede dar por la obsolescencia o por nuevas estrategias del modelo del negocio. Por ejemplo, en el sistema SINSAS, al liberarse la versión 1.0 el cliente solicitó modificaciones a los requerimientos. El cliente argumentó que las nuevas políticas del modelo del negocio exigían la modificación de los requerimientos y de no hacerse no podría ser puesto en operación el producto final. En una situación como esta el equipo de desarrollo no tiene la culpa de estos cambios inesperados. Y la solución a un problema como éste es realizar actividades de mantenimiento contratadas como un nuevo servicio con cargo al cliente.

5.4 Mantenimiento del proyecto SINSAS

Los cambios en las políticas de una empresa pueden dar como resultados cambios en el modelo del negocio, mismos que pueden repercutir en los requerimientos del sistema de software solicitado. Los cambios en los requerimientos modifican el diseño y pueden ser modificaciones grandes o pequeñas, pero que al final son cambios. Esto fue lo que ocurrió en el proyecto SINSAS, ya que una nueva política modificó sustancialmente el diseño del sistema.

La nueva política determinó que un cierto cobro ya no iba a ser necesario que el sistema lo registrase. Aunque éste cambio parece ser trivial la verdad fue otra, ya que éste cambio generaba hacer una serie de modificaciones a los requerimientos. Los cambios en los requerimientos necesitan del mayor esfuerzo ya que ellos exigen cambios en todas las fases y etapas del proceso.

Una cosa importante que hay que hacer notar es que en un sistema de software los componentes no son independientes. Por ejemplo: en nuestro caso el hecho de modificar un caso de uso nos con lleva a tener que modificar a las clases entidad, control y límites. Además se tiene que examinar que estos cambios no tengan efectos colaterales en el sistema. Por ejemplo: el hecho de eliminar ciertos atributos de una entrada en un caso de uso, pueden producir cálculos erróneos a otro caso de uso que utilice esta información.

5.4.1 Modificando el caso de uso

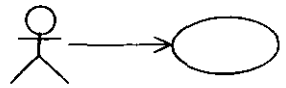
Después de haber solicitado el mantenimiento y expresado el cliente las modificaciones a sus requerimientos originales, se pasa a efectuar los cambios necesarios al diseño.

A continuación se presenta un fragmento de los requerimientos que se ven involucrados en estos cambios:

En esta página electrónica los aspirantes ingresarán: nombre, dirección(ciudad, país, código postal), Dirección electrónica, teléfono (clave del país y clave de la ciudad), universidad de procedencia, si es titulado o pasante, si quiere entrar a la maestría o al doctorado y el número de ficha de pago, la fecha de pago en el banco; la fecha de inscripción al examen de admisión la asignará el sistema en forma automática

En éste fragmento se aprecian dos entradas las cuales deberán ser modificadas y estas son: el número de ficha de pago, la fecha de pago en el banco. Ubicado el cambio en los requerimientos, se pasa a la modificación de los escenarios de los casos de uso. En el proyecto SINSAS el caso de uso y su escenario anterior y modificado se presenta en la figura 5.4 (a) , (b) y (c) respectivamente.

Las modificaciones hechas al escenario pareciera no afectar en mucho al sistema, pero en realidad esto es una apreciación errónea. Ya que existen clases de conexión a la base de datos, las que fueron diseñadas con los atributos número de ficha de pago y fecha de pago. Como se puede comprender estas clases también deberán ser modificadas y a su vez todas aquellas que utilicen a estas clases de conexión.



a) caso de uso Registro de Datos

1. Viene de Entrar página de
2. El sistema presenta una pantalla la cual contiene los datos que el usuario debe incresar: nombre,dirección,ciudad,cp,e-mail,telefono,universidad procedencia, titulado, maestria # ficha de pago, fecha de pago.

b) Escenario SINSAS versión 1.0

1. Viene de Entrar página de
2. El sistema presenta una pantalla la cual contiene los datos que el usuario debe incresar: nombre,dirección,ciudad,cp,e-mail,telefono,universidad procedencia, titulado, maestria.

c) Escenario SINSAS nueva versión

Figura 5.4: Modificaciones al caso de uso.

Con sólo unas pequeñas modificaciones en el diseño, se puede tener todo un efecto domino que multiplique el trabajo de mantenimiento.

Después se deberá hacer modificaciones a los diagramas de clase y de secuencia, para luego pasar a la codificación. Y con esto realizar de nueva cuanta el proceso de desarrollo unificado definido en los capítulos anteriores. Pasando por las cuatro fases de éste proceso.

Para el caso de uso registro de Datos el sistema SINSAS tiene una interfaz gráfica que captura las entradas del usuario. Esta interfaz se implementó mediante un Applet el cual se conecta con la base de datos. La figura 5.5 presenta la pantalla de captura correspondiente al caso de uso Registro de Datos, del proyecto SINSAS versión 1.0.

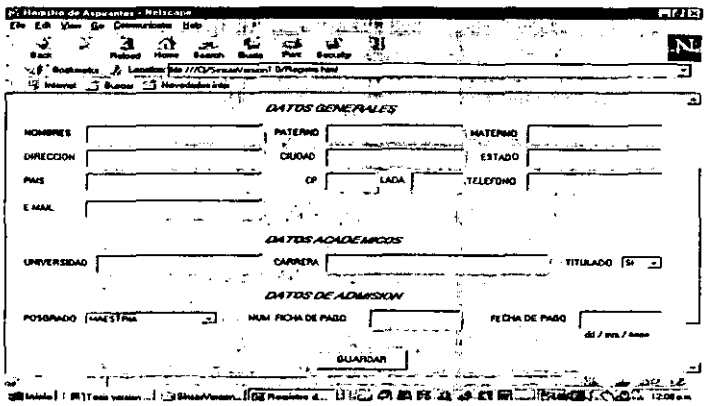


Figura 5.5: Pantalla de captura.

Como se puede apreciar, en esta interfaz se tienen los campos de texto NUM. FICHA DE PAGO y FECHA DE PAGO. Campos que no deberá tener la nueva versión del sistema de software.

Las actividades de mantenimiento realizadas al caso de uso Registro de Aspirantes:

- Modificar la base de datos.
- Modificar la interfaz de usuario.

- Revisar el diagrama de clases
- Revisar los diagramas de secuencia.
- Modificar las clase ProcedimientoDeRegistroDeAspirantes
- Modificar las clases asociadas.
- Implementar las modificaciones.
- Probar las modificaciones.
- Documentar al proceso de mantenimiento.

Es importante mencionar que para realizar estas actividades se siguieron los lineamientos del proceso de desarrollo unificado de software, mencionados en los capítulos anteriores. Por tanto no mencionaremos más de éste proceso por ocuparnos de los detalles prácticos.

5.4.1.1 Modificaciones a la base de datos

Para la realizar la modificación en la base de datos Oracle8 se utilizo el SQLplus versión 3.3 de Oracle. Eliminando de la tabla aspirantes las columnas numfichapago y fechapago. La figura 5.6 (a) muestra la estructura de la tabla de aspirantes antes de su modificación y la figura 5.6 (b) después de esta modificación.

Campo	Restricción	Tipo
NOMBRE	NOT NULL	VARCHAR2(20)
APELLIDOP	NOT NULL	VARCHAR2(15)
APELLIDOM		VARCHAR2(15)
DIRECCION		VARCHAR2(50)
CIUDAD		VARCHAR2(20)
ESTADO		VARCHAR2(20)
PAIS		VARCHAR2(20)
CODIGOPOSTAL		NUMBER(6)
DIRECCIONELECTRONICA		VARCHAR2(30)
LADA		NUMBER(6)
TELEFONO		VARCHAR2(15)
UNIVERSIDAD		VARCHAR2(50)
CARRERA		VARCHAR2(30)
TITULADO		VARCHAR2(2)
POSGRADO		VARCHAR2(9)
NUMFICHAPAGO		NUMBER(6)
FECHAPAGO		DATE
NUMEXAMEN		NUMBER(3)
ESTATUSADMISION		VARCHAR2(12)
ESTATUSPAGO		VARCHAR2(2)

a) Tabla antes de la modificación

Campo	Restricción	Tipo
NUMERO	NOT NULL	NUMBER(3)
NOMBRE	NOT NULL	VARCHAR2(20)
APELLIDOP	NOT NULL	VARCHAR2(15)
APELLIDOM		VARCHAR2(15)
DIRECCION		VARCHAR2(50)
CIUDAD		VARCHAR2(20)
ESTADO		VARCHAR2(20)
PAIS		VARCHAR2(20)
CODIGOPOSTAL		NUMBER(6)
DIRECCIONELECTRONICA		VARCHAR2(30)
LADA		NUMBER(6)
TELEFONO		VARCHAR2(15)
UNIVERSIDAD		VARCHAR2(50)
CARRERA		VARCHAR2(30)
TITULADO		VARCHAR2(2)
POSGRADO		VARCHAR2(9)
NUMEXAMEN		NUMBER(3)
ESTATUSADMISION		VARCHAR2(12)

b) Tabla después de la modificación

Figura 5.6: Estructura de la tabla aspirantes antes y después de la modificación.

5.4.1.2 Modificaciones a la interfaz de usuario

Estas modificaciones se realizaron mediante la herramienta de diseño de Applet incluidas en Visual Café versión 2.0. con esta herramienta se pueden colocar los componentes (campos de datos, etiquetas, etc.) dentro del Applet con gran facilidad. Es importante mencionar que éste software sólo sirvió para implementar la interfaz de usuario y no para la implementación del dominio del problema. La figura 5.7 muestra la nueva pantalla de captura para el caso de uso Registro de Datos.

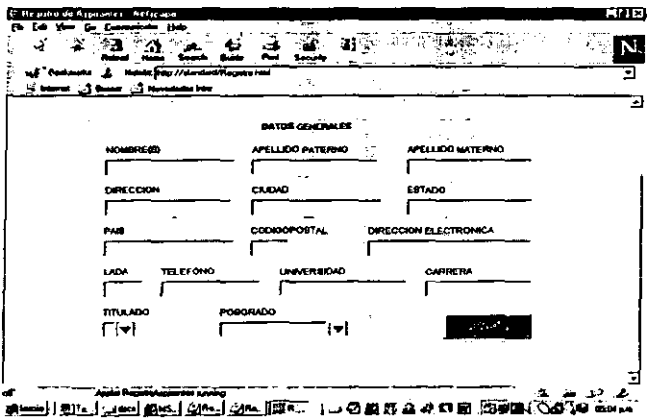


Figura 5.7: Nueva pantalla de captura.

5.4.1.3 Revisión de los diagramas de clase y secuencia

La revisión de estos diagramas se realizaron con la finalidad de determinar si se deben realizar modificaciones estructurales del sistema. Por ejemplo: el saber si

se debe modificar la relación de una clase con otra. El hecho de existir una modificación estructural, representa una modificación lógica del sistema. Siendo esto una tarea más difícil de realizar que en forma crítica representa el rediseño completo del sistema. Por suerte el mantenimiento requerido para éste sistema no fue estructural. Ya que los modificaciones exigidas fueron sobre los atributos de una clase. Lo anterior no quiere decir que no se tengan que realizar modificaciones a ciertas clases involucradas con el nuevo requerimiento.

La revisión de los diagramas de secuencia ayuda a saber si se deben eliminar, crear o modificaciones métodos a las clases involucradas. En éste caso no fue necesario crear o eliminar métodos, pero si modificar algunos.

Podemos deducir con esto que la modificación de un sistema de software puede ser sobre la organización de los objetos o sobre el objeto.

5.4.1.4 Modificación de las clases y el diagrama de secuencia

Al modificar la clase ProcedimientoDeRegistroDeAspirante eliminando las atributos "numFichapago" y "fechaPago", se modificó también a la clase MantenimientoBaseDato, esta clase es la que inserta, modifica y borra registros en la base de datos. Se conecta con la base de datos Oracle mediante un Driver JDBC. La figura 5.8 muestra el diagrama de clases correspondiente al caso de uso Registro de Datos. Este diagrama representa como la clase ProcedimientoDeRegistroDeAspirante tiene una relación de realización con la clase MantenimientoBaseDato.

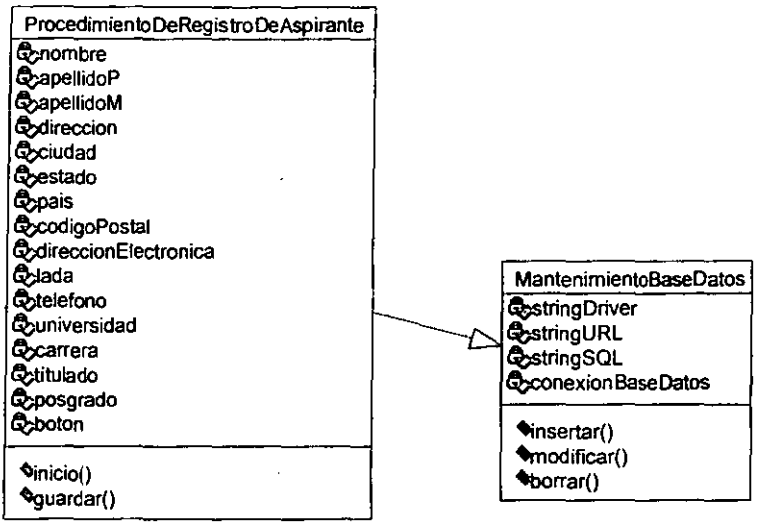
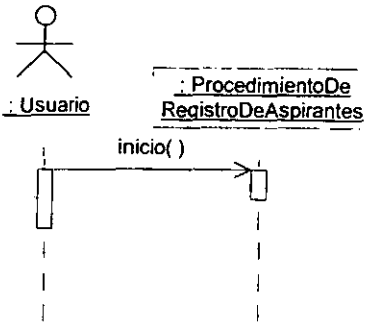


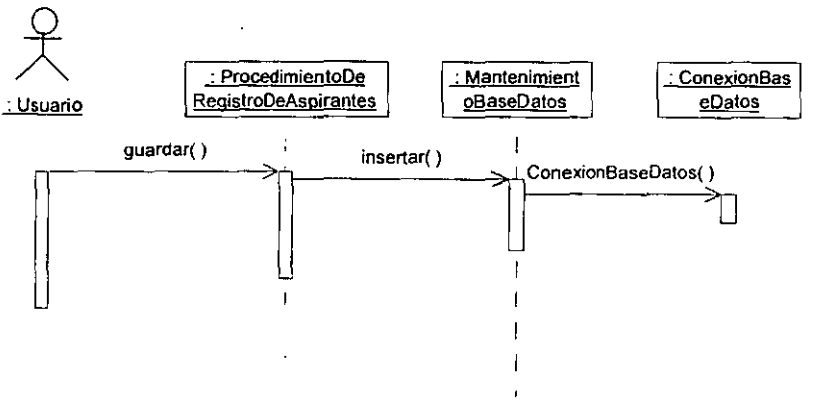
Figura 5.8: Diagrama de clases del caso de uso registro de datos.

Es importante mencionar que el diagrama anterior nos muestra las modificaciones realizadas a las clases utilizadas en el caso de uso Registro de Datos. Pero fue necesario modificar todas las clases relacionadas con alguna de estas. Revisando el diagrama de clases general para estimar el impacto de estos modificaciones sobre otros casos de uso (efecto domino).

El eliminar o crear métodos da como resultado cambios en los diagramas de secuencia ya que estos tan dinamismo al objeto. Esto no ocurrió en éste caso de uso pero si es conveniente elaborar los diagramas de secuencia correspondientes al caso de uso. La figura 5.9 muestra los diagramas de secuencia del caso de uso Registro de Datos.



a)



b)

Figura 5.9: Diagramas de secuencia.

La figura 5.9 (a) nos muestra como el usuario instancia el objeto ProcedimientoDeRegistroDeAspirante mediante el método inicio especificado en esta misma clase. La figura 5.9 (b) muestra la secuencia de objetos instanciados,

necesarios para registrar los datos de un usuario. Mediante el método guardar de la clase ProcedimientoDeRegistroDeAspirante.

Para la realización de los diagramas de clase y de secuencia se utilizó el producto de software Rational Rose 98.

5.4.1.4 La implementación del código

Para escribir del código fuente se utilizó: Rational Rose 98 y Visual café 2.0 como herramientas de ayuda. Las realización del código para las especificaciones de los métodos fueron realizadas sobre un editor de textos convencional. El copilador utilizado fue el JDK versión 1.2 de Sun Macrosystem. La herramienta Visual café sirvió como ayuda para escribir el código fuente de la interfaz de usuario. Y la herramienta Rational Rose 98, como ayuda para generar el código fuente inical del caso de uso.

A continuación se presenta el código generado por la herramienta Rational Rose 98, para la clase ProcedimientoDeRegistroDeAspirante.

```
// Source file: ProcedimientoDeRegistroDeAspirante.java

public class ProcedimientoDeRegistroDeAspirante {

   private String nombre;
   private String apellidoP;
   private String apellidoM;
   private String direccion;
   private String ciudad;
   private String estado;
   private String pais;
   private String codigoPostal;
   private String direccionElectronica;
```

```
    private String lada;
    private String telefono;
    private String universidad;
    private String carrera;
    private String titulado;
    private String posgrado;
    private int boton;

    ProcedimientoDeRegistroDeAspirante() {
    }

    public void inicio() {
    }

    public void guardar() {
    }

}
```

El anexo 11 nos muestra el código final Java para esta clase. Note que se realizaron cambios sustanciales en el código. Al incluirle objetos mas poderosos de Java y eliminar el método constructor de la clase, por un objeto de tipo Applet [4], ya que estos tienen en lugar de un método constructor un método de inicio (init). También se eliminaron clases Query2, Aspirantes, FechaValida, al hacer la clase MantenimeintoBaseDatos más eficiente.

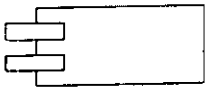
5.5 Diagramas de instalación

Un diagrama de instalación, muestra al conjunto de paquetes de clase que se utilizan en el sistema. Estos paquetes pueden ser un conjunto de clases ó un conjunto de componentes [2].

Un componente representa a un modulo de software (código fuente, código binario, ejecutable, DLL, etc.). El nombre típico de un componente es el nombre de un archivo que representa al componente. Un componente puede relacionarse en forma dependiente de otro componente.

La figura 5.9 (a) y (b) muestra los símbolos que representan al paquete y al componente respectivamente.

a) Paquete



b) Componente

Figura 5.9: Paquete y Componente.

Para realizar los diagramas de instalación debemos basarnos en el modelo por capas. Este modelo exige que todo sistema de software cuente con las siguientes capas :

Capa 1.- Conexión a la base de datos y/o al sistema operativo.

Capa 2.- Dominio del problema.

Capa 3.- Interfaz de usuario.

La capa 1 es una capa de nivel bajo, se utiliza para realizar transacciones con otros sistemas (bases de datos, sistema operativo, etc.). La capa 2 ó del dominio del problema, incluye a los requerimientos funcionales del sistema (operaciones fundamentales del sistema). Y por último la capa 3 ó interfaz de usuario que es la que comunica al usuario final con el sistema (ver figura 2.9).

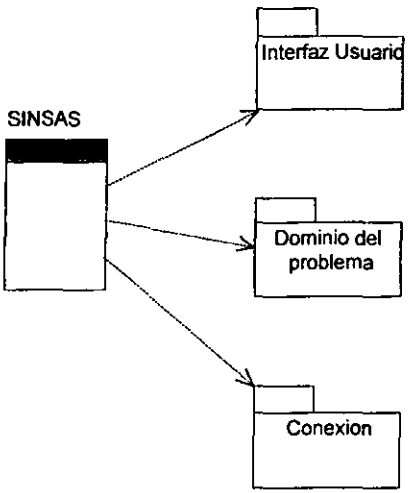
Definido el modelo de tres capas podemos entonces dividir nuestros conjuntos de clases y componentes de acuerdo a estas capas. Así a cada capa le corresponderán un conjunto de clases y/o componentes.

5.5.1 Diagramas de paquetes de clases del proyecto SINSAS

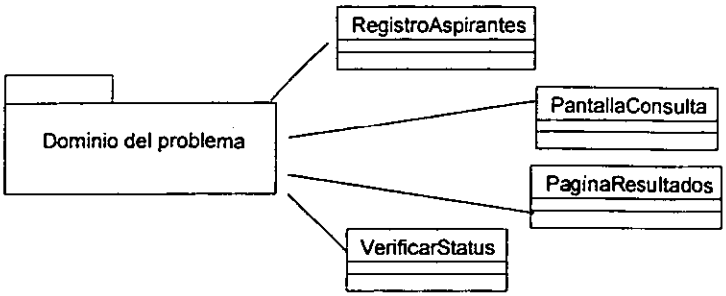
Definidos los conceptos presentados en la sección 4.5 podemos pasar a los diagramas de instalación del proyecto. Para realizar estos diagramas se utilizaron los conceptos presentados en la sección anterior. Utilizando el modelo de tres capas se utilizó un paquete por cada capa. Y un diagrama para representar a los paquetes de clases y otro para los paquetes de componentes. La figura 5.10 (a) muestra al diagrama de paquetes de clases del proyecto SINSAS. En éste diagrama se puede ver al sistema y las dependencias que tiene con los paquetes.

Además podemos notar que los nombres de los paquetes tienen nombres similares a los de sus respectivas capas. Pero estos pueden ser nombrados de otra manera según convenga. Un diagrama de paquetes de clases puede ser tan extenso como sea necesario ó estar compuesto por varios sub-diagramas.

La figura 5.10 (b) muestra al diagrama para el paquete dominio del problema. Como se puede apreciar éste paquete contiene las clases correspondientes a los requerimientos funcionales del sistema.



a) Diagrama de paquetes de clases.



b) Diagrama del paquete Dominio del problema.

Figura 5.10: Diagramas de Paquetes.

5.5.2 Diagrama de componentes del proyecto SINSAS

El diagrama de componentes muestra el paquete donde quedarán instalados los componentes (archivos: ejecutables, *.class, *.html, etc.). El sistema SINSAS básicamente consta de paginas WEB que utilizan Applets, que ejecutan los requerimientos funcionales del sistema. El diagrama de componentes para representar esta instalación se muestra en la figura 5.11.

Como comentario podemos decir que un paquete se puede entender como un directorio del sistema de archivos. Donde se quedarán instalados de manera física los archivos del sistema.

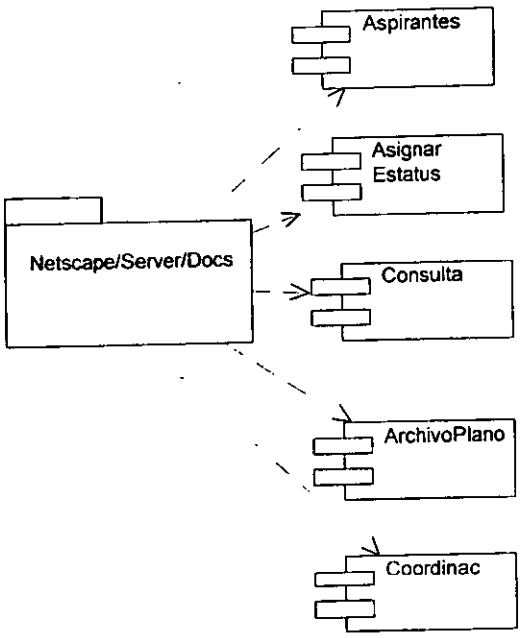


Figura 5.11: Diagrama de Componentes.

5.6 Discusión

Que importante es tener un buen diseño en la fase de construcción, por que de no ser así se pueden crear clases: no necesarias y/o con atributos y métodos mal definidos. Esto representará perdidas en la eficiencia del sistema, incremento en los costos de producción entre otras cosas negativas. Por esto, es necesario que

las personas que diseñan tengan la experiencia necesaria para poder tomar las decisiones correctas.

En el sistemas SINSAS esto ocurrió, y los costos de dar mantenimiento al sistema se incrementaron. Es aconsejable que el equipo de construcción no realice la implementación sin una discusión previa con el equipo de diseño.

Capítulo 6

Conclusiones

En nuestro entorno real lo que tenemos a nuestro alrededor son objetos, estos tienen características propias y se transforman. Esta forma de pensar deberán de tener los nuevos especialistas en el desarrollo de software orientado a objetos.

La tecnología orientada a objetos es la punta de lanza de los nuevos sistemas de software. El uso adecuado de estas tecnologías, nos proporciona las herramientas superiores para un mayor especialización de nuestra ciencia. UML y El Proceso Unificado de Desarrollo de Software, nos dan el potencial de desarrollar sistemas de software de alta calidad.

Por ser una tecnología nueva, existe poca bibliografía referente al tema, siendo pocos los libros y tesis que manejan éste conocimiento. Y casi todas estas se enfocan a una área especifica de la tecnología. A diferencia del presente trabajo, que presenta un amplio panorama de toda la tecnologia.

Al participar en un proyecto donde se aplica esta tecnología, se obtiene una experiencia importante. Y un conocimiento amplio sobre el proceso de desarrollo de software orientado a objetos.

Éste trabajo reúne los conocimientos teóricos y prácticos necesarios, para realizar un proyecto de software orientado a objetos de principio a fin. Aterriza el conocimiento practico al presentar un producto de software.

Los conocimientos prácticos son dan importantes como los teóricos, ya que para poder llevar a cabo nuestro trabajo de investigación, se tuvieron que salvar muchas barreras técnicas. Y esto lo demostramos con el siguiente ejemplo:

Se puede tener un buen diseño estático y dinámico de nuestras sistema, y escrito los códigos fuentes necesarios. Pero estos no pueden ser probados si no se tienen instalados los Servidores WEB y de la Base de Datos. Y es en esta parte donde se tiene que aplicar estos conocimientos prácticos. Todo teoría necesita de una tecnología para poder ser probada. Y una tecnología necesita de una técnica para poder realizarse.

El conocimiento de la parte técnica es muy importante si se quiere tener el control del proyecto. Como experiencia se puede aconsejar que se deben tener estos conocimientos. De esta manera se dependerá menos de otras personas y esto en la practica es suele ser bueno.

Además se debe tener conocimientos de otras áreas no sola el área de su especialidad. Un diseñador no puede realizar un diseño adecuado si no conoce el lenguaje en el que será implementado el sistema de software.

Al final se puede decir que éste trabajo realiza una abstracción y presentación de los: conocimientos, procedimientos, actividades, herramientas y técnicas

necesarias para desarrollar un producto de software. Desde la organización del equipo de desarrollo hasta el mantenimiento.

Como perspectiva de investigación y tema para un trabajo doctoral, se propone la creación de un nuevo modelo de desarrollo de software orientado a objetos basado en 3 dimensiones. En lugar de las 2 dimensiones que presenta el proceso de desarrollo unificado de software.

Bibliografía de consulta

[1] Ivar Jacobson (1999), The Unified Software Development Process, Addison Wesley, U.S.A.

[2] Grady Booch (1999), The Unified Modeling Language User Guide, Addison Wesley, U.S.A.

[3] Terry Quatrani (1998), Visual Modeling With Rational Rose Anb UML, Addison Wesley, U.S.A

[4] James Gosling (1996), The Java Lenguage Specification, Addison Wesley, U.S.A.

[5] Craig Larman (1999), UML y Patrones, Prentice Hall, México.

[6] Watts S. Humphrey (2000), Introducción to the Team Software Process, Addison Wesley, U.S.A.

[7] William Perry (1995), Effective Methods for Software Testing, Wiley, U.S.A.

[8] Murray R. Cantor (1998), Object-Oriented Project Management with UML, Wiley, U.S.A.

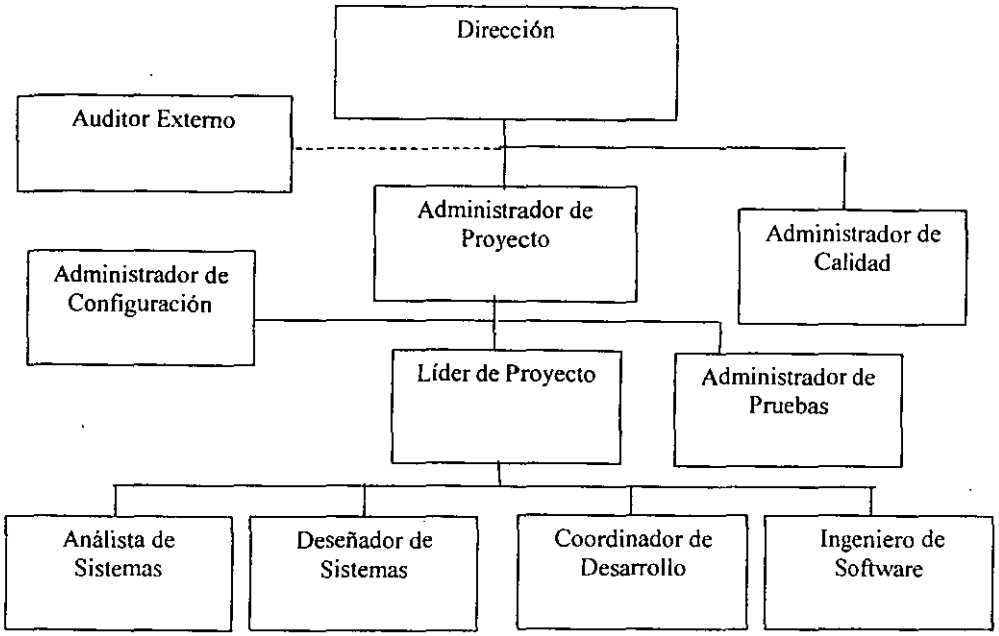
[9] Carnegie Mellon University (1995), The Capability Maturity Model, Addison Wesley, U.S.A.

[10] Goldberg Adele, Kenneth S. Rubín (1995), Succeeding With Objects, Addison Wesley, U.S.A.

[11] Norman E. Fenton (19919, Software Metrics, Chapman & Hall, U.S.A.

ANEXO 1

Organigrama del proyecto.



ANEXO 2

Registro de Tiempos versión 2.0

Nombre _____ Folio _____
Instructor _____ Program # _____

Fecha	Inicio Horas	Fin Horas	Interrupción horas	Delta Tiempo hrs	Fase	Comentarios

ANEXO 3

ESPECIFICACION DE REQUERIMIENTOS

Versión 1.0

INSCRIPCION A EXÁMEN DE ADMISION

La Coordinación de la Maestría en Ciencia e Ingeniería de la Computación de la UNAM, con sede en el IIMAS, requiere un sistema de inscripción de aspirantes al exámen de admisión a Maestría y Doctorado a través de Internet. Este sistema contará con dos tipos de acceso; en el primero tendrán acceso los aspirantes para ingresar sus datos personales y en el segundo tendrá acceso la Coordinadora del Posgrado, para consultar y actualizar información relacionada con el exámen de admisión.

En esta página electrónica los aspirantes ingresarán: nombre, dirección(ciudad, país, código postal), Dirección electrónica, teléfono (clave del país y clave de la ciudad), universidad de procedencia, si es titulado o pasante, si quiere entrar a la maestría o al doctorado y el número de ficha de pago, la fecha de pago en el banco; la fecha de inscripción al exámen de admisión la asignará el sistema en forma automática.

La coordinadora será la encargada de marcar en el sistema a todos aquellos aspirantes que hayan cumplido con todos los requisitos para presentar el exámen de admisión.

El Sistema seleccionará a los aspirantes que ya hayan sido registrados para derecho a exámen y les asignará un número de exámen.

Una vez que la Coordinadora de la Maestría tenga los resultados de los exámenes de admisión de los aspirantes ingresará en el sistema el estatus de cada uno de ellos (Admitido, No admitido, Admitido con condición).

El sistema, mostrará consultas por pantalla de los aspirantes inscritos: por fecha de inscripción, por nombre y por estatus. Así como la impresión de los listados correspondientes.

El sistema contará con una opción para redactar cartas sencillas las cuales podrán ser enviadas por correo electrónico de acuerdo a las alternativas de selección siguientes (por parte de la coordinadora):

a) Enviar a aspirantes Admitidos
b) Enviar a aspirantes No Admitidos
c) Enviar a aspirantes Admitidos con condición

La coordinara enviará mensajes individuales a los aspirantes, introduciendo el nombre de aspirante.

El sistema contará con una opción para publicar en la página del IIMAS(Instituto de Investigación en Matemáticas Aplicadas y en Sistemas) un listado de los aspirantes admitidos y otro de los admitidos con condición; en estos listados no aparecerá el nombre del aspirante, sino únicamente su número de exámen.

El sistema permitirá que se genere un archivo respaldo para poner migrar la información a otros sistemas y plataformas.

ANEXO 4

FORMATO DE SOLICITUD DE CAMBIOS

Llenado por el solicitante

Proyecto: _____ Versión\ant.:_____
Elemento de cambio:_____ Versión\nue.:_____
Descripción del cambio:_____
Razones para el cambio:_____
Solicitante:_____ Fecha:_____

Llenado por la mesa de control

Aprobado\rechazado por :_____ Fecha\Recibido:_____
Valoración del cambio:_____ Fecha\Decisión:_____
Prioridad del cambio:_____ Costo;_____
Comentarios:_____

Llenado por el implementador

Implementador: _____
Cambios\Elemento:_____ Versión:_____
Fecha de Inicio:_____ Fecha de Término:_____
Comentarios:_____

Llenado por control de calidad

Autoridad:_____ Comentarios: _____
Fecha\Entrega a SCM:_____
Fecha\Entrega a calidad:_____
Fecha\Término:_____

ANEXO 5

PROCEDIMIENTO PARA EL CONTROL DE CAMBIOS.

En cuanto a su mecánica.

1. El solicitante deberá llenar el formato de control de cambios de acuerdo al procedimiento de llenado del formato SCM\05.
2. El solicitante revisará y entregará al responsable de la mesa de control (administrador de configuraciones):
3. El responsable de la mesa de control en conjunto con los demás miembros de la mesa de control revisarán y evaluarán la solicitud.
4. La mesa de control aprobará o desaprobará la solicitud.
5. El responsable de la mesa de control notificará mediante el mismo formato al solicitante y en caso de ser aprobado al implementador. De la aprobación o desaprobación de la solicitud. Esta notificación no deberá ser mayor a cinco días hábiles después de la fecha de entrega de la solicitud.
6. El responsable de la mesa de control entregará al administrador de configuraciones las solicitudes de cambios y documentos y productos anexos a esta solicitud.
7. El administrador de configuraciones clasificará, respaldará y archivará estos productos.

En cuanto al almacenamiento de cada producto.

Se acordó que la clasificación de cada producto generado será almacenado en un dispositivo magnético, tal dispositivo será identificado con el nombre del rol de la persona que lo realizó o el nombre de la etapa.

En cuanto a la generación de código del software se acordó la utilización de dispositivos magnéticos para cada versión, identificados en su etiqueta por el nombre del sistema.y su número de la versión.

Responsable.

El Administrado de Configuraciones será la persona responsable de clasificar ordenadamente cada producto.

Perioricidad.

La revisión y control a los cambios de versiones será realizada por el Administrador de Configuraciones cada tres días, esto variará dependiendo del flujo de trabajo y de los productos entregables.

ANEXO 6

PLAN DE RIESGOS

Objetivo:

Servir de base para garantizar el éxito del proyecto, y tomar las medidas correctivas en el momento oportuno.

Tabla de contenido

NUM	RIESGO	METODOS DE CONTROL	REFERENCIAS
1	Cambio en los Requerimientos	Uso de técnicas orientadas al cliente	Capítulo 10, Desarrollo orientado al Cliente
2	Requerimientos superfluos o personal de desarrollo meticuloso	Filtrado de requerimientos.	Capítulo 14, Filtrado de requerimientos.
		Uso de Entrega por etapas	Capítulo 36, Entrega por etapas
3	Recorte de la Calidad	Deje tiempo a las actividades de control de calidad y preste atención a las bases del control de calidad	Sección 4.3 Bases del Control de Calidad
4	Planificación demasiado optimista	Utilización de varias técnicas de estimación.	Capitulo 8, Estimación.
5	Diseño inadecuado	Tener tiempo suficiente para una actividad de diseño y planificación explícitos.	Capitulo 7, Planificación del ciclo de vida y Diseño en la Sección 4.2
		Tener inspecciones de diseño	
6	Síndrome de la panacea	Ser escépticos sobre los problemas de la productividad	Sección 15.5 Síndrome de la panacea
		Configurar un grupo de herramientas de software	
7	Desarrollo Orientado a la Investigación	No intente investigar y maximizar la velocidad de desarrollo al mismo tiempo.	Sección 3.3 Desarrollo Orientado a la Investigación
		Utilice un ciclo de vida orientado al riesgo.	Capitulo 7, Planificación del ciclo de vida
8	Personal Mediocre	Personal con talento	Capitulo 12, Equipo de Trabajo
		Contratar y planificar los miembros clave del equipo mucho antes de que comience el proyecto	Capitulo 13, Estructura de equipos
		Preparación.	
		Definición del equipo	Capitulo 13, Estructura de equipos
9	Problemas con el personal contratado	Pedir referencias	Capitulo 28 Desarrollo Externo
		Estimar la capacidad del personal contratado antes contratarlo.	
		Tener buenas relaciones con el personal contratado	
10	Problemas entre el personal de desarrollo y el cliente	Utilizar técnicas orientadas al cliente	Capitulo 10, Desarrollo orientado al Cliente

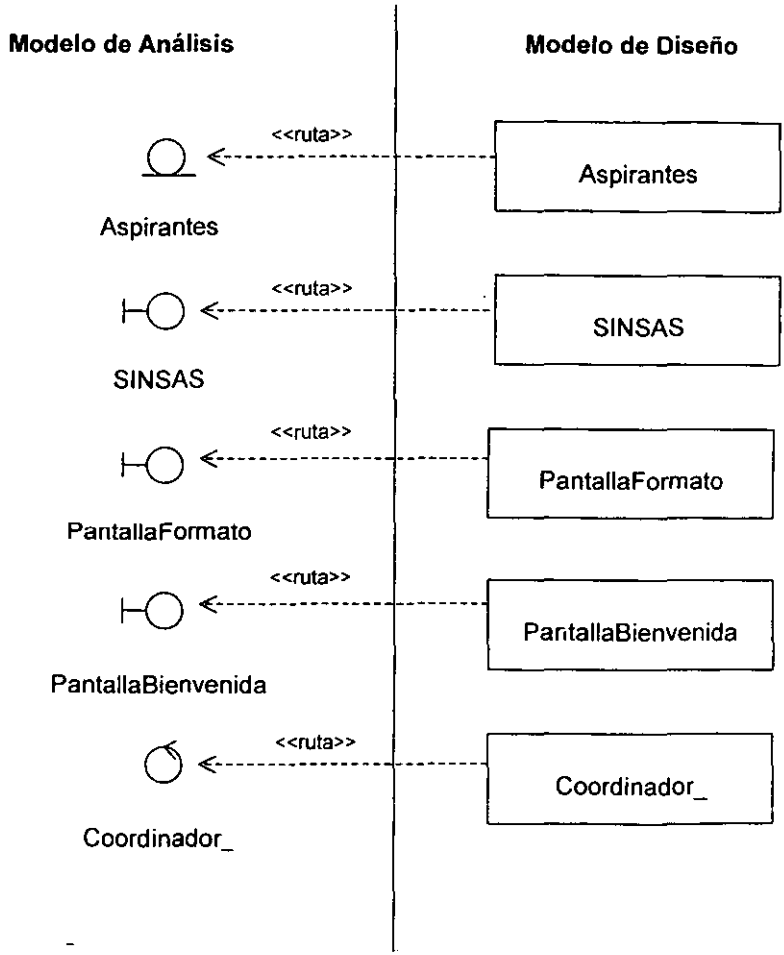
ANEXO 7

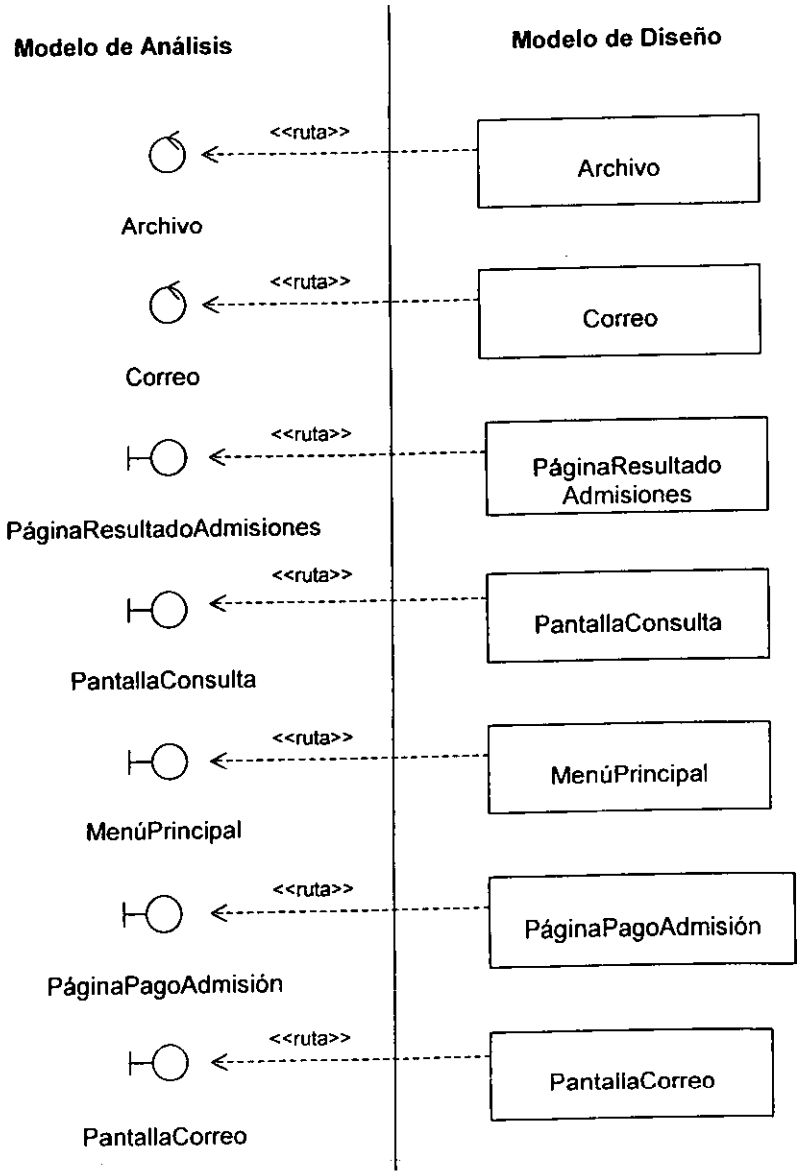
Reporte del Proyecto

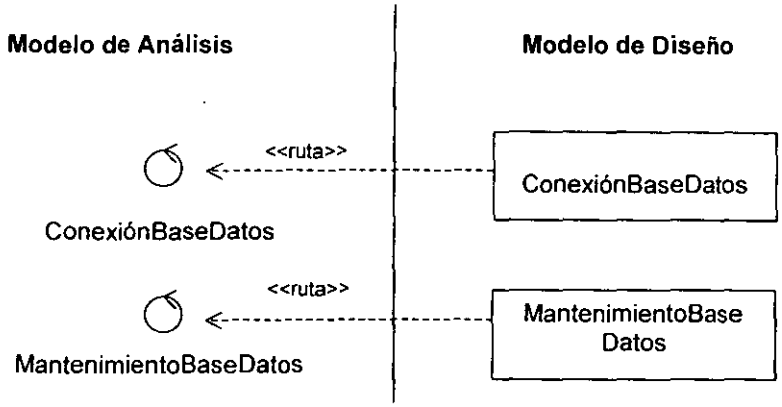
Nombre del proyecto:
Fecha:
administrador del proyecto:

Novedades:
Valoración de riesgos:
Análisis del valor ganado:
Seguimiento del calendario y estabilidad:
Gráficas de entregables para:
Otras métricas:
Revisión de elementos:

ANEXO 8


Modelo de Análisis
Modelo de Diseño
<<ruta>>
Aspirantes
Aspirantes
<<ruta>>
SINSAS
SINSAS
<<ruta>>
PantallaFormato
PantallaFormato
<<ruta>>
PantallaBienvenida
PantallaBienvenida
<<ruta>>
Coordinador_
Coordinador_


Modelo de Análisis
Modelo de Diseño
<<ruta>>
Archivo
Archivo
<<ruta>>
Correo
Correo
<<ruta>>
PáginaResultado Admisiones
PáginaResultadoAdmisiones
<<ruta>>
PantallaConsulta
PantallaConsulta
<<ruta>>
MenúPrincipal
MenúPrincipal
<<ruta>>
PáginaPagoAdmisión
PáginaPagoAdmisión
<<ruta>>
PantallaCorreo
PantallaCorreo


Modelo de Análisis
Modelo de Diseño
<<ruta>>
ConexiónBaseDatos
ConexiónBaseDatos
<<ruta>>
MantenimientoBase Datos
MantenimientoBaseDatos

ANEXO 10

```
/*
        Nombre de Archivo: ProcedimientoDeRegistroDeAspirante.java
        Lenguaje: Java, JDK 1.1.5,
        s.o.: Solaris Sun (UNIX)
        Descripción: applet que captura los datos de los aspirantes
        Version 1.0
        Autor: Eddie David Basto Aguilar
        Fecha creación: 30/nov/1999
*/

import java.awt.*;
import java.applet.*;
import Query2;
import Aspirantes;
import java.util.*;
import FechaValida;

public class ProcedimientoDeRegistroDeAspirante extends Applet{

        //Declaraciones de componentes o atributos

        Vector numCampos= new Vector();
        String valores;
        Query2 aspirantes;
        Label cadena;
        FechaValida validar;
        java.awt.Panel pane1;
        java.awt.Panel pane2;
        java.awt.Panel pane3;
        java.awt.Label labelSeGrabo;
        TextField obtieneNombre;
        TextField obtieneApellidoP;
        TextField obtieneApellidoM;
        java.awt.Label labelNombres;
        java.awt.Label labelDatoGenarales;
        java.awt.Label labelPaterno;
        java.awt.Label labelMaterno;
        TextField obtieneDireccion;
        TextField obtieneCiudad;
        TextField obtieneEstado;
        java.awt.Label labelDireccion;
        java.awt.Label labelCiudad;
        java.awt.Label labelEstado;
```

```
TextField obtinePais;
TextField obtieneCp;
TextField obtieneLada;
java.awt.Label labelPais;
java.awt.Label labelCp;
java.awt.Label labelLada;
java.awt.Label labelTelefono;
java.awt.Label labelDireccionElectronica;
TextField obtieneTelefono;
TextField obtieneDireccionElectronica;
java.awt.Label labelDatosAcademicos;
java.awt.Label labelUniversidad;
java.awt.Label labelCarrera;
java.awt.Label labelTitulado;
TextField obtieneUniversidad;
TextField obtieneCarrera;
Choice obtieneTitulo;
java.awt.Label labelDatosAdmision;
java.awt.Label labelposgrado;
java.awt.Label labelnumFichaPago;
java.awt.Label labelFechaPago;
Choice obtienePosgrado;
TextField obtieneNumFichaPago;
TextField obtieneFechaPago;
java.awt.Button continuar;
java.awt.Button guardar;

//inicia el applet e instancia los objetos

public void init(){

        validar=new FechaValida();
        valores=new String();
        aspirantes= new Query2();
        setSize(770,470);
        setLayout(null);
        pane1 = new java.awt.Panel();
        pane1.setLayout(null);
        pane1.setBounds(0,0,800,500);
        pane1.setBackground(new Color(12632256));
        add(pane1);

        //mensaje datos no validos
        pane2 = new java.awt.Panel();
        pane2.setBounds(300,200,200,120);
        pane2.setBackground(new Color(12632235));
        pane2.setVisible(false);
```

```
pane1.add(pane2);
Label mensaje0=new Label("¡¡ ERROR !!");
Label mensaje1=new Label("Posibles campos vacios ó");
Label mensaje2=new Label("Tipos de datos no Validos");
continuar=new Button("Continuar");
pane2.add(mensaje0);
pane2.add(mensaje1);
pane2.add(mensaje2);
pane2.add(continuar);

//mensaje se Grabo
pane3 = new java.awt.Panel();
pane3.setBounds(300,200,200,60);
pane3.setBackground(new Color(12632235));
pane3.setVisible(false);
pane1.add(pane3);
Label mensaje3=new Label("!Se grabo tu solicitud¡");
Label mensaje4=new Label("Estudia mucho y suerte");
pane3.add(mensaje3);
pane3.add(mensaje4);



obtieneNombre = new TextField();
obtieneNombre.setBounds(84,48,204,28);
pane1.add(obtieneNombre);
obtieneApellidoP = new TextField();
obtieneApellidoP.setBounds(360,48,156,28);
pane1.add(obtieneApellidoP);
obtieneApellidoM = new TextField();
obtieneApellidoM.setBounds(588,48,156,28);
pane1.add(obtieneApellidoM);
labelNombres = new java.awt.Label("NOMBRES");
labelNombres.setBounds(12,48,60,23);
pane1.add(labelNombres);
labelDatoGenarales = new java.awt.Label("DATOS GENERALES");
labelDatoGenarales.setBounds(288,12,156,23);
labelDatoGenarales.setFont(new Font("Dialog",
Font.BOLD|Font.ITALIC, 14));
pane1.add(labelDatoGenarales);
labelPaterno = new java.awt.Label("PATERNO");
labelPaterno.setBounds(300,48,60,24);
pane1.add(labelPaterno);
labelMaterno = new java.awt.Label("MATERNO");
labelMaterno.setBounds(523,48,64,24);
pane1.add(labelMaterno);
obtieneDireccion = new TextField();
```

```
obtieneDireccion.setBounds(84,84,204,28);
pane1.add(obtieneDireccion);
obtieneCiudad = new TextField();
obtieneCiudad.setBounds(360,84,157,28);
pane1.add(obtieneCiudad);
obtieneEstado = new TextField();
obtieneEstado.setBounds(588,84,154,28);
pane1.add(obtieneEstado);
labelDireccion = new java.awt.Label("DIRECCION");
labelDireccion.setBounds(12,84,72,24);
pane1.add(labelDireccion);
labelCiudad = new java.awt.Label("CIUDAD");
labelCiudad.setBounds(307,84,52,24);
pane1.add(labelCiudad);
labelEstado = new java.awt.Label("ESTADO");
labelEstado.setBounds(535,84,52,24);
pane1.add(labelEstado);
obtinePais = new TextField();
obtinePais.setBounds(84,120,204,28);
pane1.add(obtinePais);
obtieneCp = new TextField();
obtieneCp.setBounds(360,120,56,28);
pane1.add(obtieneCp);
obtieneLada = new TextField();
obtieneLada.setBounds(456,120,60,28);
pane1.add(obtieneLada);
labelPais = new java.awt.Label("PAIS");
labelPais.setBounds(12,120,36,24);
pane1.add(labelPais);
labelCp = new java.awt.Label("CP");
labelCp.setBounds(336,120,24,24);
pane1.add(labelCp);
labelLada = new java.awt.Label("LADA");
labelLada.setBounds(420,120,36,24);
pane1.add(labelLada);
labelTelefono = new java.awt.Label(" TELEFONO");
labelTelefono.setBounds(516,120,72,24);
pane1.add(labelTelefono);
labelDireccionElectronica = new java.awt.Label("E-MAIL");
labelDireccionElectronica.setBounds(12,160,72,24);
pane1.add(labelDireccionElectronica);
obtieneDireccionElectronica = new TextField();
obtieneDireccionElectronica.setBounds(84,160,204,28);
pane1.add(obtieneDireccionElectronica);
obtieneTelefono = new TextField();
obtieneTelefono.setBounds(588,120,156,28);
pane1.add(obtieneTelefono);
```

```
labelDatosAcademicos = new java.awt.Label("DATOS
ACADEMICOS");
labelDatosAcademicos.setBounds(288,206,170,28);
labelDatosAcademicos.setFont(new Font("Dialog",
Font.BOLD|Font.ITALIC, 14));
pane1.add(labelDatosAcademicos);
labelUniversidad = new java.awt.Label("UNIVERSIDAD");
labelUniversidad.setBounds(12,242,84,24);
pane1.add(labelUniversidad);
labelCarrera = new java.awt.Label("CARRERA");
labelCarrera.setBounds(300,242,60,24);
pane1.add(labelCarrera);
labelTitulado = new java.awt.Label("TITULADO");
labelTitulado.setBounds(631,242,64,24);
pane1.add(labelTitulado);
obtieneUniversidad = new TextField();
obtieneUniversidad.setBounds(96,242,192,28);
pane1.add(obtieneUniversidad);
obtieneCarrera = new TextField();
obtieneCarrera.setBounds(360,242,252,28);
pane1.add(obtieneCarrera);
obtieneTitulo = new Choice();
obtieneTitulo.setBounds(696,242,48,25);
obtieneTitulo.addItem("SI");
obtieneTitulo.addItem("NO");
pane1.add(obtieneTitulo);
labelDatosAdmision = new java.awt.Label("DATOS DE ADMISION");
labelDatosAdmision.setBounds(288,290,170,28);
labelDatosAdmision.setFont(new Font("Dialog",
Font.BOLD|Font.ITALIC, 14));
pane1.add(labelDatosAdmision);
labelposgrado = new java.awt.Label("POSGRADO");
labelposgrado.setBounds(12,326,72,24);
pane1.add(labelposgrado);
labelnumFichaPago = new java.awt.Label("NUM. FICHA DE PAGO");
labelnumFichaPago.setBounds(271,326,136,24);
pane1.add(labelnumFichaPago);
labelFechaPago = new java.awt.Label("FECHA DE PAGO");
labelFechaPago.setBounds(547,326,100,24);
pane1.add(labelFechaPago);
obtienePosgrado = new Choice();
obtienePosgrado.setBounds(84,326,152,24);
obtienePosgrado.addItem("MAESTRIA");
obtienePosgrado.addItem("DOCTORADO");
pane1.add(obtienePosgrado);
obtieneNumFichaPago = new TextField();
obtieneNumFichaPago.setBounds(408,326,105,28);
```

```
pane1.add(obtieneNumFichaPago);
obtieneFechaPago = new TextField();
obtieneFechaPago.setBounds(648,326,96,28);
pane1.add(obtieneFechaPago);
Label etiquetaFecha=new Label("  dd / mm / aaaa");
etiquetaFecha.setBounds(648,348,96,28);
pane1.add(etiquetaFecha);
showStatus("Inserta tus datos");
guardar = new java.awt.Button();
guardar.setActionCommand("button");
guardar.setLabel("GUARDAR");
guardar.setBounds(350,386,101,30);
guardar.setForeground(new Color(0));
guardar.setBackground(new Color(12632259));
pane1.add(guardar);
//Registros de eventos
Mouse mouse = new Mouse();
guardar.addMouseListener(mouse);
continuar.addMouseListener(mouse);
}

//clase que controla los eventos del mouse


class Mouse extends java.awt.event.MouseAdapter  {
public void mouseClicked(java.awt.event.MouseEvent event)
{       Object object = event.getSource();
        if (object == guardar){
                guardar_MouseClick(event);
                }
        if (object == continuar){
                continuar_MouseClick(event);
                }
}
}


//si se oprime el botón guardar
void guardar_MouseClick(java.awt.event.MouseEvent event)
{
guardar.setEnabled(false);
//si son validos los datos se guardan
try{


        if (
           obtieneNombre.getText()!=""
```

```
        && obtieneApellidoP.getText()!=""
        && obtieneApellidoM.getText()!=""
        && obtieneDireccion.getText()!=""
        && obtieneCiudad.getText()!=""
        && obtieneEstado.getText()!=""
        && obtinePais.getText()!=""
        && obtieneDireccionElectronica.getText()!=""
        //&& obtieneCp.getText()!=""
        //&& obtieneLada.getText()!=""
        //&& obtieneNumFichaPago.getText()!=""
        && obtieneTelefono.getText()!=""
        && obtieneUniversidad.getText()!=""
        && obtieneCarrera.getText()!=""
        && obtieneFechaPago.getText()!=""
        && Integer.parseInt(obtieneCp.getText()) >= 0
        && Integer.parseInt(obtieneLada.getText()) >= 0
        && Integer.parseInt(obtieneNumFichaPago.getText()) >=0
        &&validar.fechaValida(obtieneFechaPago.getText())==true)
{

numCampos=aspirantes.consulta("select * from aspirantes");
//int num=numCampos.size();

Aspirantes asp= new Aspirantes();

asp.capturarNombre(obtieneNombre.getText());
asp.capturarApellidoP(obtieneApellidoP.getText());
asp.capturarApellidoM(obtieneApellidoM.getText());
asp.capturarDireccion(obtieneDireccion.getText());
asp.capturarCiudad(obtieneCiudad.getText());
asp.capturarEstado(obtieneEstado.getText());
asp.capturarPais(obtinePais.getText());
asp.capturarDireccionElectronica(obtieneDireccionElectronica.getTex
t());
asp.capturarCodigoPostal(Integer.parseInt(obtieneCp.getText()));
asp.capturarLada(Integer.parseInt(obtieneLada.getText()));
asp.capturarNumFichaPago(Integer.parseInt(obtieneNumFichaPago.
getText()));
asp.capturarTelefono(obtieneTelefono.getText());
asp.capturarUniversidad(obtieneUniversidad.getText());
asp.capturarCarrera(obtieneCarrera.getText());
asp.capturarTitulado(obtieneTitulo.getSelectedItem());
asp.capturarPosgrado(obtienePosgrado.getSelectedItem());
asp.capturarNumExamen(numCampos.size()+1);
asp.capturarFechaPago(obtieneFechaPago.getText());
asp.capturarEstatusAdmision("NO ADMITIDO");
asp.capturarEstatusPago("NO");
```

```
            System.out.println("salida de los captura y voy al insert");
            System.out.println(obtieneNombre.getText());
            //Inserta los campos en la tabla y avisa del suceso
            aspirantes.Inserta(asp);

            System.out.println("salida del insert");
            pane3.setVisible(true);
            pane1.setEnabled(false);
            stop();
            }
            }
            //Si no son validos lanza el aviso
            catch(Exception e){
                  pane2.setVisible(true);
                  guardar.setEnabled(false);
                  }
            guardar.setEnabled(true);
      }


      void continuar_MouseClick(java.awt.event.MouseEvent event)
            {
                  pane2.setVisible(false);
                  guardar.setEnabled(true);
            }


}
```

ANEXO 11

```
/*
        Nombre de Archivo: ProcedimientoDeRegistroDeAspirante.java
        Lenguaje: Java, JDK 1.1.5,
        s.o.: Solaris Sun (UNIX)
        Descripción: applet que captura los datos de los aspirantes
        Version 1.0
        Autor: Eddie David Basto Aguilar
        Fecha creación: 30/nov/1999
        Fecha de modificación 25/nov/2000
*/


import java.awt.*;
import java.applet.*;

public class ProcedimientoDeRegistroDeAspirante extends Applet
{

        String sql;
        Mantenimiento c;

        java.awt.Panel panel1;
        java.awt.Label DATOSLabel;
        java.awt.Label NOMBRELabel;
        java.awt.Label APELLIDOPLabel;
        java.awt.Label APELLIDOMLabel;
        java.awt.Label DIRECCIONLabel;
        java.awt.Label CIUDADLabel;
        java.awt.Label ESTADOLabel;
        java.awt.Label PAISLabel;
        java.awt.Label CODIGOPOSTALLabel;
        java.awt.Label DIRECCIONELECTRONICALabel;
        java.awt.Label LADALabel;
        java.awt.Label TELEFONOLabel;
        java.awt.Label UNIVERSIDADLabel;
        java.awt.Label CARRERALabel;
        java.awt.Label TITULADOLabel;
        java.awt.Label POSGRADOLabel;
        MiTextField NOMBRE;
        MiTextField APELLIDOP;
        MiTextField APELLIDOM;
        MiTextField DIRECCION;
```

```
MiTextField CIUDAD;
MiTextField ESTADO;
MiTextField PAIS;
MiTextField CODIGOPOSTAL;
MiTextField DIRECCIONELECTRONICA;
MiTextField LADA;
MiTextField TELEFONO;
MiTextField UNIVERSIDAD;
MiTextField CARRERA;
Choice     TITULADO;
Choice     POSGRADO;
Button     GUARDAR;

Panel mensaje;
Label lmensaje;

Panel mensaje1;
Label lmensaje1;
Button continuar;


public void init()
{
        setLayout(null);
        setSize(576,336);
        setBackground(new Color(16777215));


        mensaje = new java.awt.Panel();
        mensaje.setLayout(null);
        mensaje.setBounds(250,48,192,72);
        mensaje.setBackground(new Color(13290239));
        mensaje.setVisible(false);
        add(mensaje);
        lmensaje = new java.awt.Label("\"Se graba su información\"");
        lmensaje.setBounds(24,24,156,24);
        lmensaje.setFont(new Font("Dialog", Font.BOLD, 12));
        mensaje.add(lmensaje);

        mensaje1 = new java.awt.Panel();
        mensaje1.setLayout(null);
        mensaje1.setBounds(250,48,228,84);
        mensaje1.setBackground(new Color(13290239));
        mensaje1.setVisible(false);
        add(mensaje1);
        lmensaje1 = new java.awt.Label("\"Error: No se grabo su
        información\"");
```

```
lmensaje1.setBounds(12,12,204,24);
lmensaje1.setFont(new Font("Dialog", Font.BOLD, 12));
mensaje1.add(lmensaje1);
continuar = new java.awt.Button();
continuar.setActionCommand("button");
continuar.setLabel("Continuar");
continuar.setBounds(72,48,84,24);
continuar.setBackground(new Color(12632256));
mensaje1.add(continuar);


panel1 = new java.awt.Panel();
panel1.setLayout(null);
panel1.setBounds(0,0,576,336);
panel1.setBackground(new Color(12632256));
panel1.setVisible(true);
add(panel1);
DATOSLabel = new java.awt.Label("DATOS GENERALES");
DATOSLabel.setBounds(216,0,120,24);
DATOSLabel.setFont(new Font("Dialog", Font.BOLD, 12));
panel1.add(DATOSLabel);
NOMBRELabel = new java.awt.Label("NOMBRE(S)");
NOMBRELabel.setBounds(24,36,84,20);
panel1.add(NOMBRELabel);
APELLIDOPLabel = new java.awt.Label("APELLIDO PATERNO");
APELLIDOPLabel.setBounds(204,36,132,20);
panel1.add(APELLIDOPLabel);
APELLIDOMLabel = new java.awt.Label("APELLIDO MATERNO");
APELLIDOMLabel.setBounds(396,36,132,20);
panel1.add(APELLIDOMLabel);
DIRECCIONLabel = new java.awt.Label("DIRECCION");
DIRECCIONLabel.setBounds(24,96,84,20);
panel1.add(DIRECCIONLabel);
CIUDADLabel = new java.awt.Label("CIUDAD");
CIUDADLabel.setBounds(204,96,60,20);
panel1.add(CIUDADLabel);
ESTADOLabel = new java.awt.Label("ESTADO");
ESTADOLabel.setBounds(396,96,60,20);
panel1.add(ESTADOLabel);
PAISLabel = new java.awt.Label("PAIS");
PAISLabel.setBounds(24,156,48,20);
panel1.add(PAISLabel);
CODIGOPOSTALLabel = new java.awt.Label("CODIGOPOSTAL");
CODIGOPOSTALLabel.setBounds(204,156,108,20);
panel1.add(CODIGOPOSTALLabel);
DIRECCIONELECTRONICALabel = new
java.awt.Label("DIRECCION ELECTRONICA");
```

```
DIRECCIONELECTRONICALabel.setBounds(348,156,168,20);
panel1.add(DIRECCIONELECTRONICALabel);
LADALabel = new java.awt.Label("LADA");
LADALabel.setBounds(24,216,48,20);
panel1.add(LADALabel);
TELEFONOLabel = new java.awt.Label("TELEFONO");
TELEFONOLabel.setBounds(96,216,72,20);
panel1.add(TELEFONOLabel);
UNIVERSIDADLabel = new java.awt.Label("UNIVERSIDAD");
UNIVERSIDADLabel.setBounds(240,216,96,20);
panel1.add(UNIVERSIDADLabel);
CARRERALabel = new java.awt.Label("CARRERA");
CARRERALabel.setBounds(420,216,72,20);
panel1.add(CARRERALabel);
TITULADOLabel = new java.awt.Label("TITULADO");
TITULADOLabel.setBounds(24,276,72,20);
panel1.add(TITULADOLabel);
POSGRADOLabel = new java.awt.Label("POSGRADO");
POSGRADOLabel.setBounds(168,276,84,20);
panel1.add(POSGRADOLabel);
NOMBRE = new MiTextField(20);
NOMBRE.setBounds(24,60,160,24);
NOMBRE.setBackground(new Color(16777215));
panel1.add(NOMBRE);
NOMBRE.setCursor(new Cursor(Cursor.TEXT_CURSOR));
APELLIDOP = new MiTextField(15);
APELLIDOP.setBounds(204,60,156,24);
APELLIDOP.setBackground(new Color(16777215));
panel1.add(APELLIDOP);
APELLIDOP.setCursor(new Cursor(Cursor.TEXT_CURSOR));
APELLIDOM = new MiTextField(15);
APELLIDOM.setBounds(396,60,156,24);
APELLIDOM.setBackground(new Color(16777215));
panel1.add(APELLIDOM);
APELLIDOM.setCursor(new Cursor(Cursor.TEXT_CURSOR));
DIRECCION = new MiTextField(50);
DIRECCION.setBounds(24,120,156,24);
DIRECCION.setBackground(new Color(16777215));
panel1.add(DIRECCION);
DIRECCION.setCursor(new Cursor(Cursor.TEXT_CURSOR));
CIUDAD = new MiTextField(20);
CIUDAD.setBounds(204,120,160,24);
CIUDAD.setBackground(new Color(16777215));
panel1.add(CIUDAD);
CIUDAD.setCursor(new Cursor(Cursor.TEXT_CURSOR));
ESTADO = new MiTextField(20);
ESTADO.setBounds(396,120,156,24);
```

```
ESTADO.setBackground(new Color(16777215));
panel1.add(ESTADO);
ESTADO.setCursor(new Cursor(Cursor.TEXT_CURSOR));
PAIS = new MiTextField(20);
PAIS.setBounds(24,180,160,24);
PAIS.setBackground(new Color(16777215));
panel1.add(PAIS);
PAIS.setCursor(new Cursor(Cursor.TEXT_CURSOR));
CODIGOPOSTAL = new MiTextField(5);
CODIGOPOSTAL.setMask("99999");
CODIGOPOSTAL.setBounds(204,180,48,24);
CODIGOPOSTAL.setBackground(new Color(16777215));
panel1.add(CODIGOPOSTAL);
CODIGOPOSTAL.setCursor(new Cursor(Cursor.TEXT_CURSOR));
DIRECCIONELECTRONICA = new MiTextField(30);
DIRECCIONELECTRONICA.setBounds(348,180,204,24);
DIRECCIONELECTRONICA.setBackground(new Color(16777215));
panel1.add(DIRECCIONELECTRONICA);
DIRECCIONELECTRONICA.setCursor(new
Cursor(Cursor.TEXT_CURSOR));
LADA = new MiTextField(5);
LADA.setMask("99999");
LADA.setBounds(24,240,48,24);
LADA.setBackground(new Color(16777215));
panel1.add(LADA);
LADA.setCursor(new Cursor(Cursor.TEXT_CURSOR));
TELEFONO = new MiTextField(8);
TELEFONO.setMask("99999999");
TELEFONO.setBounds(96,240,120,24);
TELEFONO.setBackground(new Color(16777215));
panel1.add(TELEFONO);
TELEFONO.setCursor(new Cursor(Cursor.TEXT_CURSOR));
UNIVERSIDAD = new MiTextField(50);
UNIVERSIDAD.setBounds(240,240,156,24);
UNIVERSIDAD.setBackground(new Color(16777215));
panel1.add(UNIVERSIDAD);
UNIVERSIDAD.setCursor(new Cursor(Cursor.TEXT_CURSOR));
CARRERA = new MiTextField(30);
CARRERA.setBounds(420,240,132,24);
CARRERA.setBackground(new Color(16777215));
panel1.add(CARRERA);
CARRERA.setCursor(new Cursor(Cursor.TEXT_CURSOR));
TITULADO = new Choice();
TITULADO.setBounds(24,300,40,20);
TITULADO.addItem("SI");
TITULADO.addItem("NO");
panel1.add(TITULADO);
```

```
            POSGRADO = new Choice();
            POSGRADO.setBounds(168,300,156,20);
            POSGRADO.addItem("MAESTRIA");
            POSGRADO.addItem("DOCTORADO");
            panel1.add(POSGRADO);
            GUARDAR = new java.awt.Button();
            GUARDAR.setActionCommand("button");
            GUARDAR.setLabel("GUARDAR");
            GUARDAR.setBounds(444,288,108,36);
            GUARDAR.setBackground(new Color(8421504));
            panel1.add(GUARDAR);

            NOMBRE.requestFocus();

            c=new Mantenimiento();

      }


public boolean action(Event evt, Object obj)
      {
            if("GUARDAR".equals(obj))
                  enviar();
            else if("Continuar".equals(obj))
                  continuar();
            return true;
      }


      void enviar()

      {

       sql="insert into aspirantes values("     + "contador.nextval"        +
                              " ,'" + NOMBRE.getUpperText()                +
                              "','" + APELLIDOP.getUpperText()             +
                              "','" + APELLIDOM.getUpperText()             +
                              "','" + DIRECCION.getUpperText()             +
                              "','" + CIUDAD.getUpperText()                +
                              "','" + ESTADO.getUpperText()                +
                              "','" + PAIS.getUpperText()                  +
                              "','" + CODIGOPOSTAL.getText()               +
                              "','" + DIRECCIONELECTRONICA.getText()       +
                              "','" + LADA.getText()                       +
                              "','" + TELEFONO.getText()                   +
                              "','" + UNIVERSIDAD.getUpperText()           +
                              "','" + CARRERA.getUpperText()               +
```

```
                    "','" + TITULADO.getSelectedItem()          +
                    "','" + POSGRADO.getSelectedItem()          +
                    "','" +     ""                              +
                    "','" +     ""                              +
                    "','" +     "0"                             +
                    "','" +     "NO ADMITIDO"                   +
                    "','" +     "NO"                            + "')";

            c.insert(sql);


        if(c.getSuccess()){
        panel1.setEnabled(false);
        mensaje.setVisible(true);

        }
        else
        {
        panel1.setEnabled(false);
        mensaje1.setVisible(true);
        }
        }

        void continuar()
        {
        mensaje1.setVisible(false);
        panel1.setEnabled(true);
        NOMBRE.requestFocus();
        }


}
```