UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Posgrado en Ciencia e Ingeniería de la Computación.

CONVERSION DE UN PROTOTIPO ACADEMICO DE SOFTWARE EN UN PRODUCTO TERMINADO:

Caso del Sistema de Inscripciones al Posgrado en Ciencia e Ingeniería de la Computación (SIPOCIC)

T E S I S

Que para obtener el Grado de

MAESTRO EN CIENCIAS de la Computación

P r e s e n t a

PABLO IGNACIO CASTILLEJO GARCIA

Asesora: DRA. HANNA OKTABA

Ciudad de México, Febrero del 2001

ÍNDICE

* * *
* *
*

Prefacio

La siguiente tesis presenta una metodología de desarrollo de sistemas de software, la cual fue aplicada para construir un sistema de software.

La metodología usada es combinación de las prácticas sugeridas por el Modelo de Madurez de Capacidades (CMM) y el Proceso Unificado, por lo que se enmarca éste trabajo en los ámbitos de la Ingeniería de Software.

Se retoman las experiencias del desarrollo de un sistema en versión prototipo y se construye una nueva versión, aplicando la metodología.

INTRODUCCIÓN

La Ingeniería de Software es la parte de las Ciencias de la Computación que trata sobre metodologías de trabajo para desarrollar sistemas de software. Como disciplina es una rama multidisciplinaria cuyos intereses están en ayudar a las personas a organizar un trabajo que la mayoría de las veces es complejo: Construir un software.

A mediados del año de 1999, salió a la luz pública una nueva metodología que prometía ser la culminación de los esfuerzos y experiencias de muchos años de trabajo. Estaba arropada por el nombre de tres profesionales de reconocido prestigio en el área del desarrollo de software. Su nombre *Proceso Unificado*.

Para esas fechas se impartió en el Posgrado en Ciencia e Ingeniería de la Computación, el curso de Ingeniería de Software Orientada a Objetos. A esto agreguemos que desde hacía tiempo existía la necesidad de un sistema de software para el área administrativa que auxiliara en el proceso de inscripción.

Surge la intención de trabajar la recién salida metodología para hacer el experimento académico de construir el sistema pedido.

Dados estos hechos, ésta tesis habla principalmente sobre la metodología del proceso unificado, las modificaciones que se hicieron para usarla y el resultado; así como el sistema final después de hacer una revisión del prototipo para tener una nueva versión.

El objetivo es convertir a un sistema de software en producto terminado, subsanando las faltas que dejó un sistema prototipo. Para ello se utilizará una versión modificada del Proceso Unificado y se reportarán los resultados.

La limitación principal que se enfrentó fue la falta de tiempo para hacer un grupo de trabajo para cubrir todas las ideas iniciales de mejora desde el prototipo.

Cabe hacerse la pregunta ¿es factible convertir en un producto terminado un sistema prototipo que viene de un proyecto académico?

Tomemos el caso del popular lenguaje de programación BASIC (*Beginner's All purpose Symbolic Instruction Code*) el cual originalmente fue creado con fines académicos exclusivamente, y que luego tuvo mucho éxito comercial. El BASIC fue creado por John G. Kemeny y Thomas E. Kurtz, del *Darmouth College* en 1964[1]. La intención del lenguaje era facilitar el uso de las computadoras a los estudiantes, y el proyecto contó con el patrocinio de la National Science Foundation de los EU. El BASIC llegó a ser tan exitoso que en cierto momento, el Consejo Nacional de Profesores de Matemáticas de Estados Unidos, debatió si debía apoyarse al lenguaje como una alternativa al FORTRAN.

La estructura del trabajo es la siguiente: el primer capítulo muestra las motivaciones que llevaron a hacer el sistema de software. Para el siguiente capítulo se mencionan a detalle las metodologías y guías que se ocuparon, y la herramienta más utilizada para documentar y entender el desarrollo del producto. En el tercer capítulo se narra la historia de la creación de la primera versión del Sistema de Inscripciones. El cuarto capítulo abre los caminos a seguir para trabajar con el sistema resultante, planteando metas y pretensiones. Se narran las actividades hechas y se mencionan resultados. Para el quinto capítulo se hace una comparación de las versiones del sistema, a manera de poder llegar a las conclusiones. Finalmente tenemos la bibliografía y dos apéndices.

[1] Véase [Freiberg1986]

CAPÍTULO 1

NACIMIENTO DE UN SISTEMA DE SOFTWARE

Cambio de costumbres por la aparición del Internet.
Durante la década de los noventas se vivió un fuerte impulso a la tecnología sobre Internet, alentada por las grandes empresas que ven en la red de redes, un potencial mundo de negocios donde el mercado de posibles compradores es simple y sencillamente, todo el mundo donde se cuente con una computadora y una línea telefónica.

Con el auge de las computadoras a finales del siglo XX, han ido cambiando diversos usos y costumbres de un número en continuo crecimiento de personas que toman a la computadora como herramienta esencial en el desarrollo de sus actividades diarias.

La computadora, al ser una invención que trata de emular al cerebro humano, pretende sustentar la creación o mejora de un ambiente de trabajo, diversión, producción o aprendizaje, que agilice y potencíe posibilidades, aumentando opciones y capacidades que no se habían alcanzado antes; de ahí que se vea ahora a la computadora como una extensión del hombre. Y la columna vertebral que hace de las computadoras una herramienta tan usada y versátil, es su posibilidad de comunicarse con otras, a través de la construcción de redes que toman gradualmente niveles tan grandes que tienen alcances mundiales. Esto es precisamente lo que hoy conocemos como Internet: una red de redes que facilita el acceso a recursos entre computadoras que están tan lejanas como el otro lado del mundo, o tan cercanas como el cuarto de al lado.

La Internet se vuelve una herramienta que sin duda es imprescindible para el mundo moderno. En los negocios se necesita tener información inmediata de todo y para todo: cerrar un negocio, decidir acciones a futuro, modificar políticas, armar estrategias, crear nuevos productos, cambiar ideas, vender acciones, coordinar recursos, etc.

En el campo académico, hoy es cosa común, la integración de grupos de trabajo con personas que están en diferentes países, y como la Internet permite comunicarse rápida y económicamente, así como compartir archivos de datos, artículos o cualquier producto de sus investigaciones, se tiene la sensación de tener al compañero en el escritorio de al lado.

La Internet se ha convertido en un espacio con posibilidades ilimitadas: Es desde una biblioteca de dimensiones sorprendentes con varios miles de millones de "ejemplares" disponibles, hasta el lugar de trabajo, ocio, diversión o consumo. Sería más fácil decir qué cosa no existe en la Internet, que hablar de su contenido.

Internet como fenómeno de la interacción humana es hoy en día un tema de estudio usual entre científicos sociales. Hay incluso quienes hablan de una nueva era dominada por la abrumadora presencia de la información, elevada a niveles de adoración.

Con la llegada del Internet al alcance de cada vez más personas, se intenta hacer "todo" lo humanamente imaginable a través de Internet.

Sobre los trámites administrativos y las posibilidades de hacerlos a distancia con el uso del Internet.

Una de tantas actividades que hacemos, en tanto sociedad organizada en torno a instituciones, es la realización de trámites de tipo administrativo: comprar un producto, solicitar una inscripción, pagar un impuesto, inscribir al recién nacido al registro civil, sacar dinero del banco, vender un terreno, etc.

Una idea básica de la utilización de los sistemas de cómputo, es la de auxiliar en procesos de tipo administrativo y reducir los tiempos de atención al mínimo para así evitar errores humanos en los procesos comunes, como el llenado de formas que avalan un procedimiento cualquiera; esto simplifica los procesos, y evita muchas veces el tener que presentarse físicamente en lugares que no siempre tienen la capacidad de espacio u organización para atender grandes demandas para ciertos periodos de tiempo.

Trámites en los centros educativos.

Parte característica de una institución académica, es la de realizar evaluaciones periódicas donde se aprecie el grado de asimilación y avance de los estudiantes.

En caso de que estos demuestren haber adquirido los suficientes conocimientos como para manejarlos con cierto grado de destreza, se procede a abrirles el camino hacia nuevos cursos, a menos que ya hayan finalizado sus estudios.

La realización de procesos de inscripción o reinscripción se presenta como un proceso cíclico en toda institución académica.

Ya que existe la posibilidad de hacer trámites de relativa sencillez, por medio del Internet, en el presente trabajo se toma un trámite común y repetitivo en cualquier institución académica: Automatizar el proceso de inscripción/reinscripción por parte de los educandos.

La Ingeniería de Software.
Una de las áreas más importantes dentro del campo de las Ciencias de la Computación, es la Ingeniería de Software, cuyo interés principal está en la organización y el aseguramiento del éxito de la construcción o mantenimiento de software.

Durante el semestre 2000-I, en el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, se impartió el curso Ingeniería de Software Orientado a Objetos, a cargo de la Dra. Hanna Oktaba y la MC. Guadalupe Ibargüengoitia. El periodo de tiempo en que transcurrió el semestre fue del 30 de Agosto al 15 de Diciembre de 1999.

Desde un principio se planteó hacer un sistema de software que cubriera una necesidad muy específica del posgrado: automatización del proceso de inscripción/reinscripción.

Entre las motivaciones para hacerlo, se hayaban las siguientes:

- El proceso era lento.
- Había errores de llenado de formas o falta de requisitos.
- Se consumía demasiado tiempo del personal administrativo.
- Ocasionalmente había que esperar demasiado tiempo para terminar el trámite.
- Había necesidad de instrucciones del personal administrativo para guiar el proceso.

Así que la respuesta natural a estas deficiencias, fue la construcción de un sistema de software que ahorrara tiempo, esfuerzo, espacio y capacidad. Y que además dicho sistema operara por Internet.

De esta forma tomó cuerpo la propuesta y se plantearon las primeras restricciones para el desarrollo del sistema, y se estableció lo siguiente a manera de reglas:

- Trabajar con alguna metodología de software.
- Hacer que el sistema opere por Internet.
- Estandarizar los productos de trabajo.
- Contar con algún mecanismo de control de calidad.

Como parte de los objetivos del curso mencionado, se contemplaba hacer un software que fuera útil y no quedara abandonado como "un experimento más".

Posibilidades del sistema para liberarse como producto para otros centros de estudios.
Sería una aportación interesante llevar el sistema de software hasta un grado en que pudiera adaptarse a cualquier tipo de requisitos involucrados el proceso de inscripción / reinscripción de cualquier institución académica.

Hipótesis

Apoyándose en la metodología del Proceso Unificado, es posible transformar un prototipo de software académico en un producto terminado, dentro de un periodo relativamente corto de tiempo.

El caso se refiere a un Sistema de Inscripciones de una entidad académica funcionando en Internet, funcional, robusto, fácilmente mantenible, que cubra las necesidades tanto de un usuario administrativo como de un usuario solicitante de inscripción o reinscripción.

CAPÍTULO 2

MARCO TEÓRICO PARA LA CONSTRUCCIÓN DEL SISTEMA

I. Objetivo y necesidades.

El objetivo es dar por terminada la construcción de un sistema de software que satisfaga plenamente las necesidades de un cliente.

Una de las primeras necesidades al desarrollar un software, es la representación de los aspectos que conforman el sistema. Por ello se utilizará el *Lenguaje Unificado de Modelación*.

Para conducir todo el proceso de creación, se procuró seguir el *Proceso Unificado*.

Y a manera de llevar un mejor control del proceso, se tomaron las guías que sugiere el *Modelo de Madurez de Capacidades*, y de ésta forma sustentar lo mejor posible al Proceso Unificado.

A continuación se presentarán las ideas principales de la herramienta de modelación, del proceso para desarrollar el software y del conjunto de guías recomendadas:

- Lenguaje Unificado de Modelación (UML)
- Proceso Unificado de Desarrollo de Software
- Modelo de Madurez de Capacidades (CMM)

II. El Lenguaje Unificado de Modelación

1. Necesidad de un lenguaje estándar, el nacimiento del UML.

A través de la breve historia del desarrollo de sistemas de software, los desarrolladores (analistas, diseñadores, programadores, etc.) han expresado sus ideas y conceptos en diferentes formas y diagramas o, mejor dicho, en modelos. Aunque en otras disciplina científicas[2] ya se cuenta desde tiempo atrás con diagramas estándar de expresión, en el área del desarrollo de sistemas, aún no había mucho al respecto. Se requería una forma estándar para comunicar los modelos entre los miembros del desarrollo del sistema, y también para las futuras generaciones de desarrolladores.

Se necesitaba "algo", para comunicarse con los demás, útil como plataforma en la cual los desarrolladores pudieran analizar, pensar y proponer soluciones. Y también para registrar toda la información importante en papel o en almacenamiento electrónico.

Para 1993 existían varias metodologías del paradigma de la programación Orientada a Objetos, y cada una tenía su propia notación, lo cual convertía el área de desarrollo de sistemas en una Torre de Babel.

Algunas de las metodologías más usadas eran: Booch, Coad-Yourdon, OMT, Martín-Odell, Fusion, OMT, Shlaer-Mellor, Objectory, etc.

De todas estas, las más populares eran la metodología de Booch[3] y la OMT[4] de Rumbaugh. Estos dos se unieron el 1994 para construir una sola metodología que recogiera lo mejor de ambas. La llamaron pomposamente *El Proceso Unificado*. Las aportaciones más importantes de cada metodología fueron:

Del método de Booch:

- Notación para el análisis y los modelos iniciales de diseño.
- Credibilidad ante la comunidad del los negocios y la administración.

Del método de Rumbaugh:

- Solidez en tecnología y diseño.

---

[2] El ejemplo inmediato es la Arquitectura.

[3] Véase [Booch1994]

[4] Siglas de *Object Modeling Technique*. Véase [Rumbaugh1991]

- Notación detallada para los modelos de diseño.
- Credibilidad ante la comunidad técnica y científica.

Para Octubre de 1995 ya se tenía la versión 0.8 del Proceso Unificado. Un año después se les unió Jacobson, autor de *Object Factory*[5], quien colaboró con dos aspectos importantes desprendidos de su metodología:

- La utilización de los casos de uso.
- La modelación de la interacción de los objetos.

Para entonces el Proceso Unificado cambió su nombre. Ahora sería el **Lenguaje Unificado de Modelación**[6] o simplemente UML, cuya versión 0.91 está lista en Octubre de 1996.

La idea de tener un lenguaje estándar para la representación de modelos en diagramas, encontró bastante aceptación en la industria dedicada al software, pues con el apoyo de varias empresas se mejoró el UML y se presentó la versión 1.1 en Septiembre de 1997 al OMG[7] para su adopción como lenguaje estándar, lo cual ocurrió en Noviembre de 1998.

2. Metas del UML

La meta principal del UML es facilitar a los desarrolladores la visualización de sus productos de trabajo en diagramas y planos estandarizados. De esta forma mejoraría la comunicación entre todos los involucrados en el desarrollo de sistemas de software.

El UML es un lenguaje de modelación de propósito general.

Se pretende que sea útil en los tópicos más importantes de la actualidad en el desarrollo de software, como desarrollo en equipo, distribución, concurrencia, patrones y sistemas de larga escala.

---

[5] Vease [Jacobson1992]

[6] *Unified Modeling Language* en inglés.

[7] *Object managment Group*, una organización dedicada al estudio y estandarización de la tecnología basada en el paradigma de la orientación a objetos. Véase [WebOMG]

El UML debe ser tan simple como sea posible, mientras sea capaz de modelar un amplió rango de sistemas prácticos que necesiten ser construidos.

3. Definición del Lenguaje Unificado de Modelación

El Lenguaje Unificado de Modelación es una herramienta que ayuda a describir, entender, documentar, visualizar y documentar todos los productos que permiten y conforman el desarrollo de los sistemas de software. Está formado de símbolos o iconos, donde los iconos son notaciones gráficas que conforman una sintaxis; y detrás de ésta notación gráfica, se especifica un significado, es decir, una semántica.

4. La utilidad de los modelos.

Un modelo es una representación en un cierto medio de algo que está en otro medio. El modelo captura los aspectos importantes de las cosas que están siendo modeladas desde un cierto punto de vista que simplifica u omite el resto.

Los modelos tienen diferentes propósitos:

- Para capturar los requerimientos y precisar el dominio conocido, para que todos los involucrados estén de acuerdo con dicho modelo.
- Para pensar acerca del diseño del sistema.
- Para capturar las decisiones de diseño en una forma que estén separadas de los requerimientos.
- Para generar productos de trabajo re usables[8].
- Para organizar, encontrar filtrar, recuperar, examinar y editar información en sistemas a gran escala.
- Para explorar múltiples soluciones económicas.
- Para entender sistemas complejos.

Dentro de un modelo existe una presentación y una semántica. Para el caso de los modelos en el desarrollo de sistemas, se tienen dos aspectos principales: la información semántica y la representación visual.

[8] significa que se puedan volver a usar en otros sistemas o en otros proyectos.

El aspecto semántico captura el significado de una aplicación como una red de elementos lógicos, tales como clases, asociaciones, estados, casos de uso y mensajes. A veces a la representación visual se le llama *el modelo*.

Un modelo puede decir *que hacer* al presentar especificaciones, y también puede decir *cómo* se cumplen las funciones, al presentar alguna implementación. Estos aspectos deben ser modelados separadamente. Siempre es importante saber el *qué* antes de usar tiempo en el *cómo*.

Los modelos son generalmente descripciones de instancias. La mayoría de las instancias existen solo como parte de una ejecución en tiempo. Pero a veces las instancias en tiempo de ejecución son descripciones de otras cosas.

Existen muchas interpretaciones posibles de los modelos en un lenguaje de modelación. Se pueden definir ciertos puntos de variación semántica —lugares donde son posibles diferentes interpretaciones— y asignar a cada interpretación un nombre.

En una sola palabra, se diría que un modelo es una abstracción semánticamente cerrada de un sistema.

5. El enfoque de modelación por vistas.

Supóngase que nos disponemos a construir una casa. Para ello será necesario contar con varios datos. ¿Qué información sería útil tener? Para comenzar necesitamos saber las medidas de cada elemento que conformará la construcción, además requerimos tener cálculos de los materiales necesarios, cuántas personas se demandan, que habilidades deben tener, en qué tiempo estimamos terminar, cuánto dinero es necesario, el tipo de terreno, los tipos de acabados, el estilo de diseño, etc.

Ante la abrumadora cantidad de información utilizable, la primer idea sería separar y organizar los datos necesarios. Para esto sería recomendable tener, por ejemplo, un plano que indique las medidas de muros y trabes. En otro plano podríamos tener la información relativa a la instalación eléctrica. En otro podríamos tener la instalación de tuberías. En otro plano una descripción del personal deseado. Y así seguir con un plano por cada punto de vista útil para la construcción.

En resumen, se nos facilitaría mucho el trabajo si contáramos con planos separados de cada aspecto que nos interese de la construcción.

Esta es la idea que se trata de seguir al hacer una primera distinción de los tipos de diagramas en UML. Separarlos por vistas. Por los aspectos en que se enfocan cada una de estas vistas.

6. Conceptos básicos del UML

Las vistas en UML

Una vista es un subconjunto de elementos de modelación que representan un aspecto del sistema. A *grosso modo*, las vistas pueden ser divididas en tres áreas.

- Clasificación estructural
- Comportamiento dinámico
- Administración del modelo.

La *clasificación estructural* describe las cosas en el sistema y sus relaciones con otras cosas. A los casos de uso, clases, componentes y nodos se les llama clasificadores. Los clasificadores son la base para construir el comportamiento dinámico. Las vistas de clasificación incluyen la vista estática, la vista de caso de uso y la vista de instalación.

El *comportamiento dinámico* describe el comportamiento de un sistema a través del tiempo. El comportamiento puede ser descrito como una serie de cambios en un cierto momento del sistema percibido desde la vista estática. Las vistas de comportamiento dinámico incluyen la vista de máquinas de estado, la vista de actividades y la vista de interacción,

La *administración del modelo* describe la organización de los modelos mismos en unidades jerárquicas. El paquete es la unidad genérica organizacional para los modelos. La vista de administración de modelos se cruza con las otras vistas y las organiza para el trabajo de desarrollo y el control de la configuración.

El UML también tiene varios elementos para extender el lenguaje. Estos son los estereotipos, las restricciones y los valores etiquetados.

La siguiente tabla muestra las vistas del UML y los diagramas que despliega, así como los conceptos más relevantes en cada vista.

Área principal	Vista	Diagramas	Conceptos principales
Estructural	Vista estática	Diagrama de clases	Clase, asociación, generalización, dependencia, realización, interfase.
	Vista de caso de uso	Diagrama de caso de uso	Caso de uso, actor, asociación, extender, incluir, generalización de caso de uso.
	Vista de implementación	Diagrama de componentes	Componente, interfase, dependencia, realización
	Vista de *instalación*	Diagrama de *instalación*	Nodo, componente, dependencia, localización
Dinámica	Vista de máquina de estado	Diagrama de estados	Estado, evento, transición, acción
	Vista de actividad	Diagrama de actividad	Estado, actividad, transición, separación, unión
	Vista de interacción	Diagrama de secuencias	Interacción, objeto, mensaje, activación
		Diagrama de Colaboración	Colaboración, interacción, rol de colaboración, mensaje
Administración del modelo	Vista de administración del modelo		Paquete, subsistema, modelo.
Extensibilidad	Todas	Todos	Restricciones, estereotipos, valores etiquetados.

La Vista Estática

La vista estática modela los conceptos en el dominio de la aplicación. Esta vista es estática porque no describe el comportamiento del sistema dependiendo del tiempo. Los elementos principales de esta vista son las clases y sus relaciones: asociación, generalización, y varios tipos de dependencia.

Una **clase** es una descripción de un conjunto de objetos que comparten los mismo atributos, operaciones, relaciones y semánticas; en otras palabras, una clase es la descripción de un concepto desde el dominio de la aplicación o la solución de la solución.

Las clases se dibujan como rectángulos con tres zonas separadas siendo la primera la que contiene el nombre de la clase, la segunda tiene la lista de atributos y la tercera tiene las operaciones.

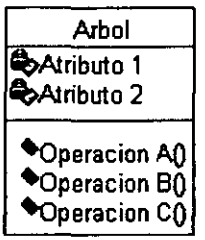


Fig 1. Representación de una clase en UML

Cuando se quiere hacer una abstracción de las operaciones que una clase ofrece, se usa el concepto de **interfase**. La interfase es una colección de operaciones que son usadas para especificar un servicio de una clase o un componente. Su representación es similar a la de una clase.

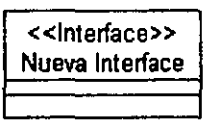


Fig 2. Representación de una interfase en UML

Las relaciones entre las clases son dibujadas como líneas conectando los rectángulos de las clases. La vista estática se modela con un diagrama de clases. Un **diagrama de clases** es aquel que muestra un conjunto de clases, interfaces y colaboraciones y sus relaciones; también puede entenderse como un diagrama que muestra una colección de elementos declarativos (estáticos).

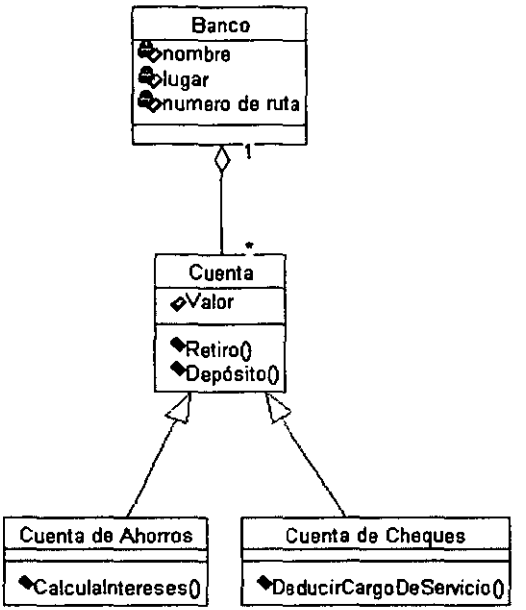


Fig 3. Diagrama de Clases en UML

Otro diagrama que muestra la vista estática de un sistema es el diagrama de objetos. Cuando una clase es *instanciada*[9], se dice que es un **objeto**, así que los **diagramas de objetos** son los que muestran un conjunto de objetos y sus relaciones en un cierto momento.

La vista de caso de uso

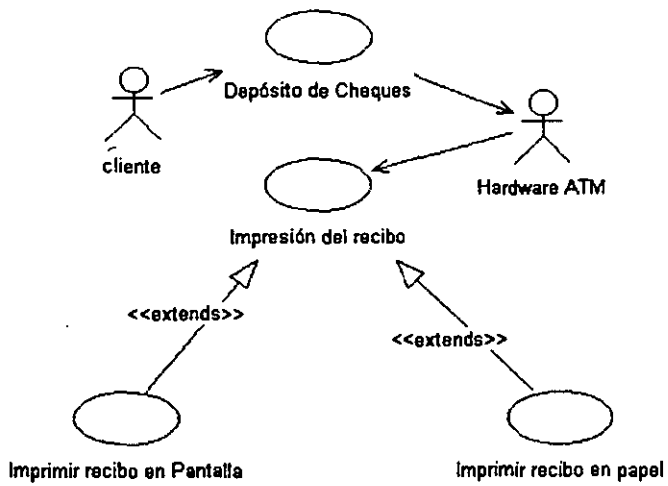
Un **Caso de uso** es una descripción de un conjunto de secuencias de acciones, incluyendo las variantes, que un sistema ejecuta y que tiene un resultado observable de un valor para un actor particular. En otras palabras, un caso de uso es una unidad coherente de funcionalidad expresada como una transacción entre los actores y el sistema. Un actor es un usuario externo al sistema. Puede ser una persona, una organización o un sistema de software. Se representa un caso de uso como un ovalo con un nombre de identificación.

---

[9] Un elemento representativo de una clase. Un ejemplo de una clase ya no como abstracción sino como ente real. Por ejemplo tener la clase "auto" y la instancia sería un "volkswagen sedan"

Fig 4. Representación de un Caso de Uso

Un **diagrama de caso de uso** muestra un conjunto de casos de uso y actores y sus relaciones; los diagramas de casos de uso, muestran una vista estática de un caso de uso de un sistema.



El propósito de la vista de caso de uso es mostrar cómo participan los actores en cada caso de uso.

La vista de interacción.

La vista de interacción describe las secuencias de mensajes intercambiados entre los roles que implementa el comportamiento de un sistema. Esta vista muestra el comportamiento dinámico en un sistema.

Una **interacción** es un comportamiento que abarca un conjunto de mensajes intercambiados entre un conjunto de objetos dentro de un contexto particular para cumplir con un propósito específico.

La vista de interacción es mostrada por dos diagramas, cada uno enfocado en diferentes aspectos: el diagrama de secuencias y el diagrama de colaboración. Ambos son isomórficos.

El diagrama de secuencia

Un diagrama de secuencia muestra un conjunto de mensajes arreglados en una secuencia de tiempo. Una secuencia muestra un escenario, esto es, la historia individual de una transacción. Uno de los usos del diagrama de secuencias es mostrar la secuencia de comportamiento de un caso de uso. Cuando el comportamiento es implementado, cada mensaje en un diagrama de secuencia, corresponde a una operación en una clase o un evento disparado en una transición en una máquina de estado.

En pocas palabras, un diagrama de secuencia es un diagrama de interacción que enfatiza el orden de ejecución de los mensajes.

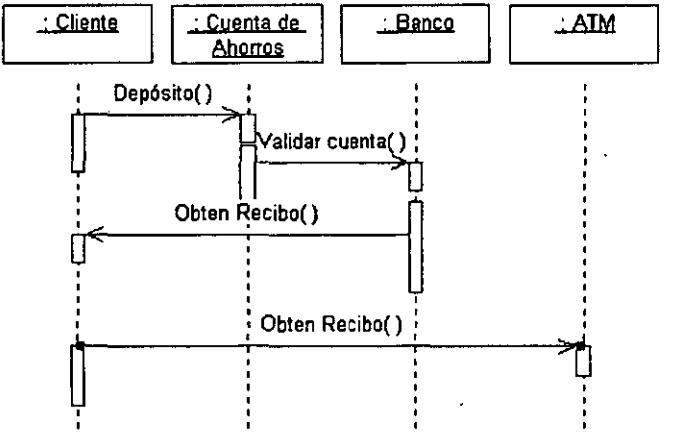


Fig 6. Diagrama de secuencia en UML

El diagrama de colaboración

Una colaboración modela los objetos y enlaces que son significativos dentro de una interacción. Un diagrama de colaboración muestra los roles en la interacción en un arreglo geométrico. Los mensajes son mostrados sobre las líneas que conectan los clasificadores de roles. Los diagramas de colaboración muestran la implementación de una operación. Las colaboraciones muestran los parámetros y las variables locales de la operación así como las asociaciones más permanentes.

Hay que observar que un diagrama de secuencia muestra los momentos en secuencia en una dimensión geométrica, pero las relaciones entre los roles están implícitas.

Un diagrama de colaboración muestra las relaciones entre los roles geométricamente así como los mensajes relacionados a las relaciones, pero el tiempo de las secuencias están menos claros porque están implicados por una secuencia de números. Aquí se enfatiza la organización estructural de los objetos para enviar o recibir mensajes.

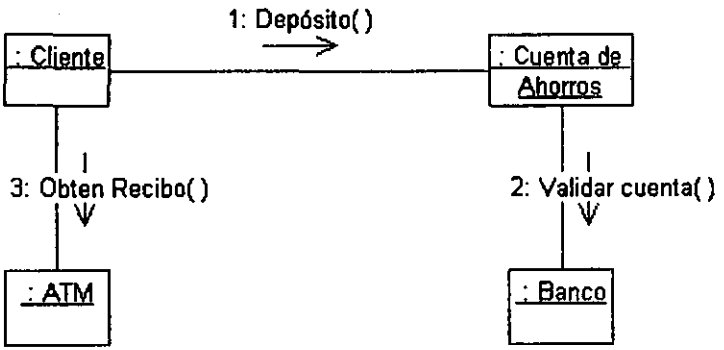


Fig 7. Diagrama de Colaboración

La vista de máquina de estado.

Una máquina de estado modela las posibles estados que pasa un objeto durante su vida en respuesta a eventos. Una máquina de estado contiene estados conectados por transiciones. Cada estado modela un periodo de tiempo durante la vida de un objeto durante la cual se satisfacen ciertas condiciones. Cuando un evento ocurre, puede provocar que el objeto pase a otro estado por una transición.

Cuando una transición se ejecuta, puede ser ejecutada una acción asociada a dicha transición. Las máquinas de estado son mostradas como diagramas de estados.

Las máquinas de estados pueden ser usadas para describir interfases de usuario, los dispositivos controladores u otros sistemas reactivos. También pueden ser usados para describir objetos pasivos que pasan por varias pases cualitativamente distintas durante su vida, cada una de las cuales tiene un comportamiento especial.

Esta vista se modela con **diagramas de estados** que muestran una máquina de estados; los diagramas de estados son parte de la vista dinámica de un sistema.

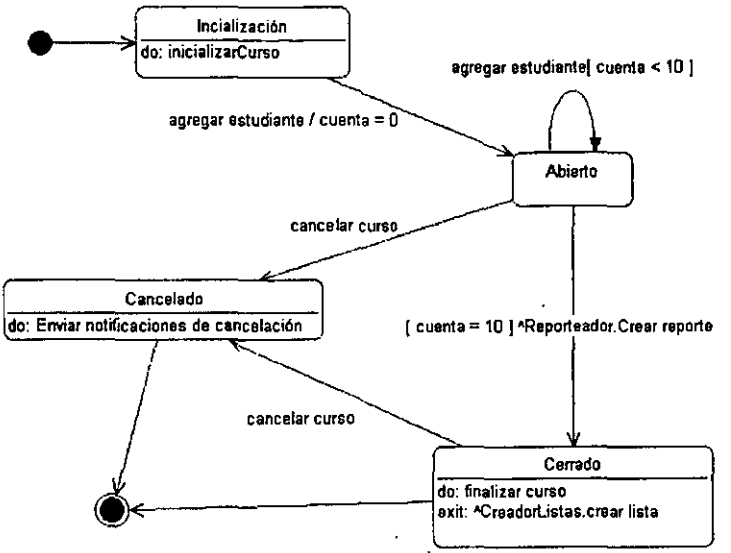


Fig 8. Diagrama de Estados

La vista de actividades.

Una gráfica de actividades es una variante de una máquina de estados que muestra las actividades computacionales involucradas en la ejecución de un cálculo. Un estado de actividad representa una actividad que se define como un paso en el flujo de la ejecución de una operación.

Una gráfica de actividad describe las actividades de secuencias grupales y concurrentes.

Las gráficas de actividades son mostradas en **diagramas de actividades**. Las flechas muestran las dependencias secuenciales. Las barras gruesas muestran controles de separaciones o de uniones.

El propósito principal de los diagramas de actividad es modela el flujo del mundo real de la organización humana, pero también pueden ser usados para modelar actividades del software.

Un diagrama de actividades ayuda a comprender la ejecución del comportamiento del sistema, sin tener que ver los detalles internos del paso de mensajes requerido en un diagrama de colaboración.

Las vistas físicas

Las vistas físicas modelan la estructura de instalación de la aplicación misma, tales como la organización en componentes y su instalación en nodos en tiempo de ejecución. Estas vistas proveen una oportunidad para mapear las clases en componentes de instalación y nodos.

Existen dos vistas físicas: la vista de implementación y la vista de instalación.

La **vista de implementación** modela los componentes en un sistema, así como las dependencias entre los componentes. También modela la asignación de las clases y otros elementos del modelo en componentes. La vista de instalación es mostrada en diagramas de componentes.

Un **componente** es una parte física y reemplazable de un sistema que conforma la realización de un conjunto de interfaces. Se representa como un rectángulo con dos barras horizontales saliendo de uno de sus lados.

Fig 9. Representación de un Componente

Los componentes pueden mostrar pequeñas líneas salientes con un círculo en el extremo, estas son llamadas interfases —un conjunto coherente de servicios. Una flecha punteada desde un componente a una interfase indica que un componente requiere el servicio que ofrece la interfase.

Los diagrama de componentes muestran un conjunto de componentes y sus relaciones; los diagramas de componentes muestran la vista estática de componentes de un sistema.

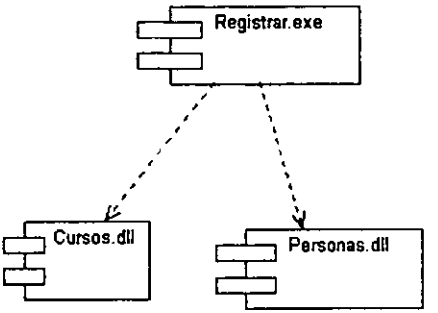


Fig 10. Diagrama de Componentes

La **vista de instalación** representa el arreglo de las instancias de los componentes en tiempo de ejecución en instancias de nodos. Un nodo es un recurso en tiempo de ejecución, tal como una computadora, un dispositivo o memoria. La vista de instalación es mostrada en diagramas de instalación. Este diagrama muestra los tipos de nodos en el sistema y los tipos de componentes que tienen. El nodo se presenta con el símbolo de un cubo.

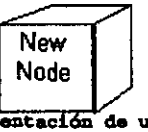


Fig 11. Representación de un Nodo en UML

El **diagrama de instalación** muestra un conjunto de nodos y sus relaciones; muestra la vista estática de la instalación de un sistema.

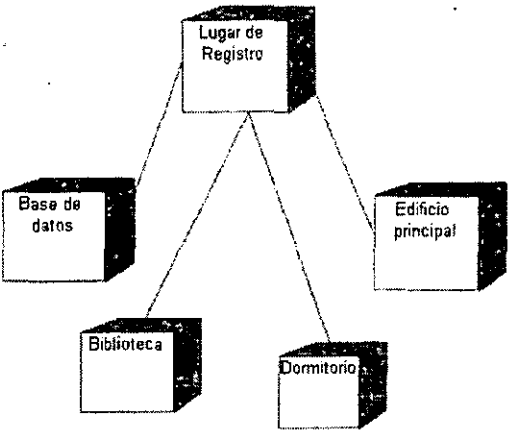


Fig 12. Diagrama de Instalación

La vista de la administración del modelo

La vista de administración del modelo, modela la organización del modelo en sí. Un modelo comprende un conjunto de paquetes que tienen elementos de modelo, tales como clases, máquinas de estado y casos de uso. Los paquetes pueden contener otros paquetes; de ahí que en un modelo se designe un paquete raíz que indirectamente contiene todos los elementos del modelo.

Los **Paquetes** se representa como fólderes, y son el mecanismo de propósito general para organizar elementos en grupos. Los paquetes son unidades para manipular los contenidos de un modelo. Cada elemento del modelo esta apropiado por un paquete u otro elemento.

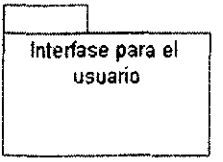


Fig 13. Representación de un paquete en UML

Se define **subsistema** como una agrupación de elementos, algunos de los cuales constituyen una especificación del comportamiento ofrecidos por los otros elementos contenidos.

La información de la administración del modelo es mostrada usualmente en diagramas de clases.

Elementos de extensibilidad.

El UML incluye tres elementos principales de extensibilidad:

- Restricciones
- Estereotipos
- Valores etiquetados.

Una **restricción** es una sentencia textual de una relación semántica expresada en algún lenguaje formal o en lenguaje natural. Las restricciones permiten agregar reglas nuevas o modificar las ya existentes. Usualmente las restricciones se colocan adyacentes a un elemento y encerradas entre símbolos de llaves. También se puede mostrar una restricción colocándola en una nota conectada al elemento con una dependencia. La siguiente tabla muestra las palabras de las restricciones que son elementos estándar en la notación del UML.

Restricción	**Aplicable al símbolo**	**Significado**
completa	Generalización	Especifica que todos los hijos en la generalización han sido especificados en el modelo y que no se permiten hijos adicionales
destruido	Instancia Liga	Especifica que la instancia o liga es destruida previamente a la terminación de la ejecución de la interacción encerrada.
disjunto	Generalización	Especifica que el objeto del padre dado no puede tener mas de uno de los hijos dados como un tipo.
implícito	Asociación	Especifica que la relación no es manifiesta, sólo conceptual.

incompleto	Generalización	Especifica que no han sido especificados todos los hijos en la generalización y que se permiten hijos adicionales
nueva	Instancia Enlace o liga	Especifica que la instancia o liga se crea en tiempo de ejecución de la interacción encerrada
O (letra o)	Asociación	Especifica que de un conjunto de asociaciones, exactamente una es manifiesta para cada objeto asociado.
Traslape	generalización	Especifica que los objetos de un padre dado pueden tener más de un hijo dado como tipo
Transitorio .	Instancia Enlace o liga	Especifica que la instancia o liga es creada durante la ejecución de la interacción encerrada pero es destruida después de completar la ejecución.

Un **estereotipo** es una extensión del vocabulario del UML que permite crear nuevos tipos de bloques de construcción que son derivados de uno ya existente pero que es necesario por ser específicos para un problema particular. Usualmente los estereotipos se ponen entre símbolos dobles de paréntesis angulados, por ejemplo <<Entidad>>, como en la figura siguiente:

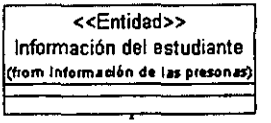


Fig. 14. Utilización de estereotipos en UML

Un **valor etiquetado** es una extensión de las propiedades de un elemento del UML, que permite crear nueva información en la especificación de ese elemento. Usualmente estos valores se colocan entre símbolos de llaves ("{" y "}"), cerca de un elemento que se quiere etiquetar. Por ejemplo {localización = cliente}.

La siguiente tabla muestra los valores que son elementos estándar en el UML.

Valor etiquetado	*Se aplica al símbolo*	*Significado*
Documentación	Todos los elementos	Especifica un comentario, descripción o explicación del elemento al cual esta adherido
Localización	La mayoría de los elementos	Especifica el nodo o componente en el cual reside el elemento
Persistencia	Clase Asociación Atributo	Especifica si el estado de la instancia es preservado después de que terminó el proceso que creó la instancia; los valores son persistentes (preservan el valor) y transitorios (que no preservan el valor)
Semántica	Clase Operación	Especifica el significado de la clase u operación.

Los tres elementos de extensibilidad permiten varios tipos de extensiones al UML sin que sea necesario cambiar el metamodelo básico del UML.

Las conexiones entre las vistas

Todas las vistas coexisten dentro de un solo modelo y sus elementos tienen varias conexiones, algunas se muestran en la siguiente tabla que, aunque no está completa, muestra las relaciones principales entre los elementos de las diferentes vistas.

Elemento	**Elemento**	**Relación**
Clase	Máquina de estado	Pertenencia
operación	Interacción	Realización

Caso de uso	Colaboración	Realización
Caso de uso	Instancia de interacción	Escenario ejemplo
Instancia de componente	Instancia de nodo	Localización
Acción	Operación	Llamada
Acción	Señal	Envío
Actividad	Operación	Llamada
Mensaje	Acción	Invocación
Paquete	Clase	Pertenencia
Rol	Clase	Clasificación

7. Organización de los conceptos del UML

Para entender la forma en que está planteado el UML, los conceptos y modelos pueden agruparse en las siguientes áreas:

Estructura estática. Cualquier modelo debe definir primero el universo de su discurso, esto es, los conceptos claves de la aplicación, sus propiedades internas y las relaciones entre ellos. Este conjunto es la estructura estática.

Los conceptos de la aplicación son modelados como clases que describen conjuntos de objetos discretos que mantienen información y se comunican para implementar un comportamiento. La información que mantienen se modela como atributos; el comportamiento se modela como operaciones.

Comportamiento dinámico. Existen dos formas de modelar el comportamiento: uno es la historia de la vida de un objeto tal como interactúa con el resto del mundo; la otra es con los patrones de comunicación de un conjunto de objetos conectados interactuando para implementar un comportamiento.

La vista de un objeto aislado es con una máquina de estados. La vista de un sistema de objetos interactuantes es una colaboración, una vista dependiente del contexto de los objetos y sus enlaces entre ellos, junto con el flujo de mensajes entre los objetos a través de las ligas de datos.

Las colaboraciones y las interacciones son mostradas en diagramas de secuencias y diagramas de colaboración, los cuales son isomórficos.

Elementos de instalación. Los modelos del UML pueden servir tanto para análisis lógicos como para implementaciones físicas. Algunos de los elementos representan a los de instalación.

Por ejemplo, un componente es una parte física reemplazable del sistema, que conforma y provee un conjunto de interfases.

Un nodo es un recurso computacional en tiempo de ejecución que está en un lugar dado. El nodo puede tener componentes u objetos. La vista de instalación describe la configuración de los nodos en un sistema en ejecución y también el arreglo de los componentes y objetos en él.

Organización de los modelos. En los sistemas grandes, la información de la modelación debe ser dividida en piezas coherentes para que los equipos puedan trabajar sobre diferentes partes en forma concurrente. Incluso para sistemas pequeños, la comprensión humana requiere la organización de modelos en paquetes de tamaño modesto. Para esto se usan los paquetes. Los paquetes son unidades de organización jerárquica de propósito general en UML. Se pueden usar para almacenamiento, control de acceso, administración de configuraciones y bibliotecas de construcción que contengan fragmentos de modelos reusables.

Mecanismos de extensibilidad. Sin importar cuan fácil o completo sea un leguaje, de todos modos la gente querrá hacer extensiones. En UML se prevé la capacidad de hacerlas. Para ello se utilizan estereotipos, restricciones y valores etiquetados para hacerlo.

8. Diagrama de la organización del vocabulario del UML.

EL UML da a los programadores un vocabulario que tiene tres categorías: cosas, relaciones y diagramas. En el diagrama siguiente podemos los tipos de elementos que forman cada categoría.

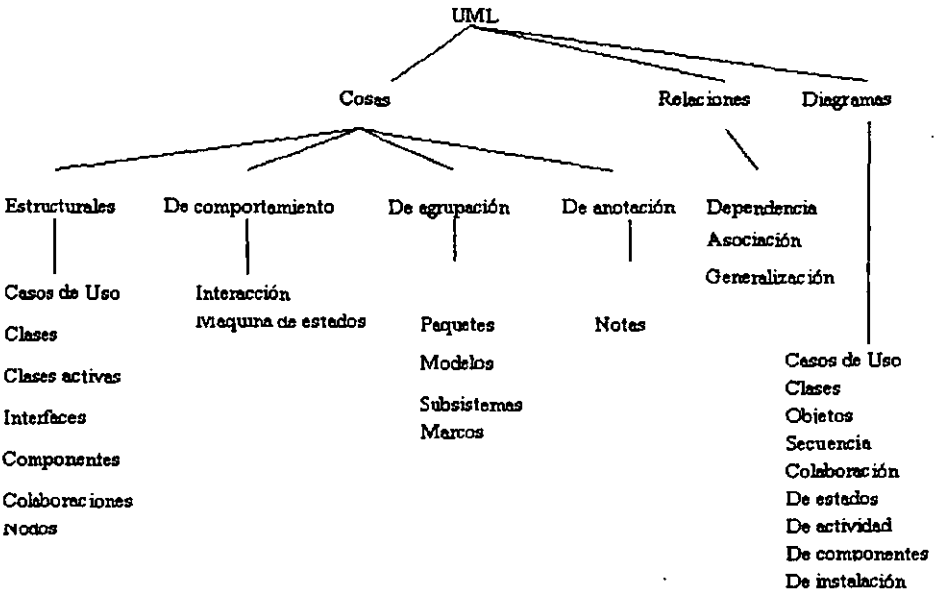


Fig. 15. Vocabulario de UML

El elemento de anotación, la **nota**, es un comentario adjuntado a un elemento o una colección de elementos.

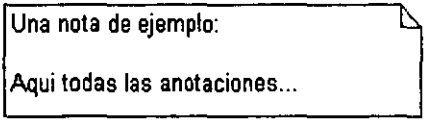


Fig 16. Representación de una nota en UML

Una **relación de dependencia** es una relación semántica entre dos cosas, en la cual el cambio en una cosa (la cosa independiente) puede afectar la semántica de la otra cosa (la dependiente)

Fig 17. Representación de una relación de dependencia en UML

Una **relación de asociación** es una relación estructural que describe un conjunto de enlaces, donde un enlace es una conexión entre objetos; la relación semántica entre dos o más clasificadores que involucran las conexiones entre sus instancias.

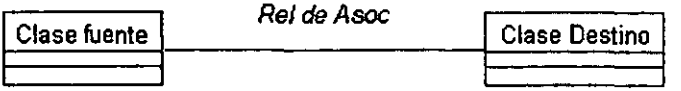


Fig 18. Representación de una relación de asociación

Una **relación de generalización** es una relación de especialización o generalización, tal que los objetos del elemento especializado (el subtipo) son sustituíbles por objetos del elemento generalizado (el supertipo). Es la forma de representar la herencia.

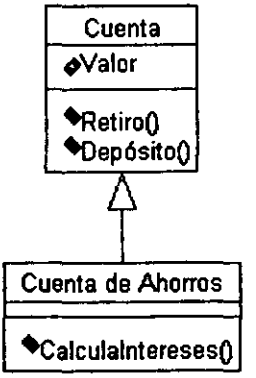


Fig 19. Representación de la relación de generalización

9. Resumen y algunos comentarios finales del UML.

El UML es un lenguaje para modelar los sistemas de computo y es el siguiente paso en la evolución de las tres metodologías más usadas en el desarrollo orientado a objetos: método de Booch, OMT y Objectory.

Está organizado en torno a vistas, que son diferentes formas de mirar a un misma sistema pero abstrayendo sólo ciertos aspectos en cada vista. El UML tiene un conjunto de elementos gráficos que forman su vocabulario, y cuenta con una estructura de interpretación que forma la semántica del lenguaje.

El UML es una herramienta que hoy en día es un estándar. Desde 1997 fue adoptado por el *Object Management Group*, que es el organismo encargado de regular los usos en el área del desarrollo de software con el enfoque la programación orientada a objetos. Algunas de las compañías que usan UML para el desarrollo o mantenimiento de sistemas de cómputo, son:

Hewlett-Packard, i-Logix, IBM, IntelliCorp, MCI Systemhouse, Microsoft, Oracle, Rational Software, DAC, Reich Technologies, SAP, Unisys y Rational Software.

III. Metodología del Proceso Unificado

1. Características principales.

Manejado por casos de uso, centrado en arquitectura, iterativo e incremental.
Actualmente se necesitan sistemas de software más grandes y más complejos, debido al aumento del poder de todos los equipos de sistemas computacionales.

Se necesita software más veloz y más eficiente, y que sea generado de la manera más rápida posible, que se adapte a nuestras necesidades cada vez más demandantes.

Por esto es necesario actualizar los métodos de construcción de software, para ser capaces de cumplir con las metas del desarrollo de software complejo que se necesita hoy en día.

La comunidad dedicada al desarrollo de software requiere entonces una forma controlada de trabajo. Un proceso que:

- Sirva de guía para ordenar las actividades del equipo.
- Dirija las tareas de los desarrolladores individualmente y de todo el equipo en conjunto.
- Especifique los productos que deben ser desarrollados
- Ofrezca criterios de monitoreo y medición para actividades y productos de proyecto

El Proceso Unificado busca ser la solución a éste problema

2. Definición del Proceso Unificado

El Proceso Unificado es un proceso para el desarrollo de software, entendiendo esto como el conjunto de actividades necesarias para transformar los requerimientos de un usuario en un sistema de software.

El proceso unificado está "basado en componentes", lo cual significa que el sistema se construye con componentes[10] de software interconectados por interfaces[11] bien definidas.

[10] Una parte física reubicable del sistema que conforme y provea la realización de un conjunto de interfaces.

[11] Una colección de operaciones que son usadas para especificar un servicio de una clase o un componente.

El Proceso Unificado usa el *Lenguaje Unificado de Modelación* (UML), para realizar todos los planos y diagramas del sistema de software.

Los aspectos más importantes del Proceso Unificado son tres conceptos claves: **manejado por casos de uso, centrado en arquitectura** y ser **iterativo e incremental.**

3. Los Casos de Uso en el Proceso Unificado.

Cuando se describen las acciones que ejecuta un sistema para responder a las acciones de un usuario, se está construyendo un caso de uso. Un caso de uso es una pieza de funcionalidad en el sistema que le da al usuario un valor como resultado. El caso de uso captura los requerimientos funcionales.

Todos los casos de uso juntos, conforman el modelo de casos de uso, que describe la funcionalidad completa del sistema. Todo el proceso de desarrollo está dirigido por los casos de uso. Esto significa que el proceso sigue un flujo ordenado y basado en los requerimientos del usuario.

Los casos de uso manejan la arquitectura del sistema y la arquitectura del sistema afecta la selección de los casos de uso. Ambos maduran conforme el ciclo de vida del sistema continua.

4. La característica de estar centrado en arquitectura.

El rol de la arquitectura de software es similar al rol que juega la arquitectura en la construcción. La arquitectura en un sistema de software es descrita desde diferentes puntos de vista, con lo cual se busca cubrir todos los detalles útiles.

El concepto de arquitectura de software contiene los aspectos estáticos y dinámicos que son significativos en el sistema.

La arquitectura es una visión del diseño completo con las características más importantes hechas visibles, dejando los detalles de lado.

¿Cómo se relacionan los casos de uso y la arquitectura? Cada producto tiene función y forma. Uno no está completo sin el otro y ambas deben estar balanceadas. Los casos de uso deben encajar en la arquitectura. Así también la arquitectura debe permitir sitios donde se realicen los casos de uso, tanto en el presente como en el futuro. Tanto la arquitectura como los casos de uso deben evolucionar en paralelo.

La arquitectura debe ser diseñada para permitir que el sistema evolucione, no únicamente en el desarrollo inicial sino a través de las futuras versiones. Para lograr esto, los arquitectos deben trabajar con los casos de uso claves del sistema. Estos son usualmente el 5 o 10% del total de los casos de uso. Son siempre los más significativos, los que constituyen el núcleo de las funciones del sistema.

5. Iterativo e Incremental

El desarrollo de algunos productos de software es labor que puede tomar desde días hasta años. Por esto, siempre es práctico dividir el trabajo en pequeñas piezas o mini proyectos.

Cada mini proyecto es una iteración que resulta en un incremento. La iteración se refiere al hecho de repetir los pasos en los flujos de trabajo[12], e incrementos se refiere al hecho de hacer crecer el producto. Todas las iteraciones deben ser controladas.

Los desarrolladores deben basar la selección de lo que debe ser implementado en una iteración, en dos factores:

- Primero, la iteración es sobre un grupo de casos de uso que en conjunto extienden la utilidad del producto.
- Segundo, la iteración trabaja sobre los riesgos más importantes.

Cada iteración trabaja sobre los artefactos o productos de la iteración previa.

En cada iteración, los desarrolladores identifican y especifican los casos de uso más relevantes, crean un diseño usando la arquitectura elegida como guía, implementan el diseño en componentes y verifican que estos satisfagan los casos de uso.

Cada iteración se conduce siempre por los cinco flujos de trabajo:

Requerimientos, análisis, diseño, implementación y prueba

Si una iteración cumple sus metas, el desarrollo procede a una nueva iteración.

Hay varios beneficios de llevar un proceso de iteraciones controladas:

- Reduce los riesgos de costo.
- Reduce los riesgos de no tener el producto listo según el calendario planeado.
- Agiliza el tiempo de desarrollo.

---

12 *Workflow* en inglés. Un flujo de trabajo puede entenderse como las tareas a realizar para llevar a cabo un trabajo.

- Se da cuenta de una realidad a veces ignorada.

Los tres conceptos (manejado por casos de uso, centrado en arquitectura e iterativo e incremental) son igualmente importantes.

La arquitectura provee la estructura en la cual se guiará el trabajo de cada iteración, mientras que los casos de uso definen las metas y dirigen el trabajo de cada iteración. No dar valor a cualquier de las tres ideas claves, reducirá el valor del proceso unificado.

6. La estructura del Proceso Unificado

El proceso unificado se repite sobre una serie de ciclos que conforman la vida del sistema. Cada ciclo termina con una versión[13] del producto.

Cada ciclo consiste de cuatro fases:

- Incepción.
- Elaboración.
- Construcción.
- Transición

Cada fase se subdivide en una o varias iteraciones.

7. El producto

Cada ciclo produce una nueva versión del sistema, y cada versión es un producto listo para ser entregado. Éste consiste en los componentes, y su código fuente, que pueden ser compilados y ejecutados, además de manuales y otros productos entregables asociados.

Para llevar a cabo el nuevo ciclo de forma eficiente, el desarrollador necesita todas las representaciones disponibles del producto de software:

---

[13] *Release*, en inglés. Se define como un conjunto, relativamente consistente y completo, de artefactos o productos entregados a usuarios internos o externos.

- Un modelo de caso de uso, con todos los casos de uso y las relaciones con los usuarios.

- Un modelo de análisis, que tiene dos propósitos: refinar el caso de uso con más detalle, y hacer una distribución inicial del comportamiento del sistema como un conjunto de objetos que proveen el comportamiento.

- Un modelo de diseño que define: a) la estructura estática del sistema, como subsistemas, clases e interfaces, y b) los casos de uso llevados a cabo como colaboraciones[14] entre los subsistemas, clases e interfaces.

- Un modelo de implementación, el cual incluye componentes (representados por código fuente) y el mapeo de las clases a los componentes.

- Un modelo de instalación[15], que define los nodos físicos de las computadoras y mapea los componentes a esos nodos.

- Un modelo de pruebas, el cual describe los casos de prueba que verifican los casos de uso.

- Y una representación de la arquitectura.

El sistema puede tener también un modelo de dominio o un modelo del negocio que describa el contexto del negocio del sistema.

8. Las fases dentro de un ciclo

Cada ciclo se hace en un periodo de tiempo. Cada periodo de tiempo es dividido en cuatro fases. Cada fase termina un hito[16]. Se define un hito como la disponibilidad de un conjunto de artefactos; esto es que ciertos modelos o documentos han sido llevados a un estado prescrito.

Los hitos son elementos que ayudan a la administración del proyecto, como la decisión de pasar a otra fase y el monitoreo del progreso del trabajo en las fases, así también para llevar un registro del tiempo y esfuerzo consumido en cada fase. Estos últimos datos son útiles para poder estimar el tiempo y requerimientos de gente para otros proyectos.

---

[14] Una sociedad de clases, interfaces y otros elementos que trabajan juntos para proporcionar algún comportamiento cooperativo que sea más grande que la suma de sus elementos.

[15] *Deployment* en inglés.

[16] *Milestone* en inglés. Hito es un punto de referencia que marca acontecimientos importantes en un proyecto y que se utiliza para controlar el progreso del mismo.

En la figura siguiente se muestran los flujos de trabajo en la columna izquierda. Las curvas representan aproximadamente la cantidad de esfuerzo que en cada fase deben llevar a cabo los flujos de trabajo.

Recuérdese que cada fase está usualmente dividida en iteraciones o mini proyectos.

El siguiente diagrama muestra una hipotética cantidad de esfuerzo a llevar a cabo en cada uno de los flujos de trabajo según las fases en que se encuentre el proyecto.

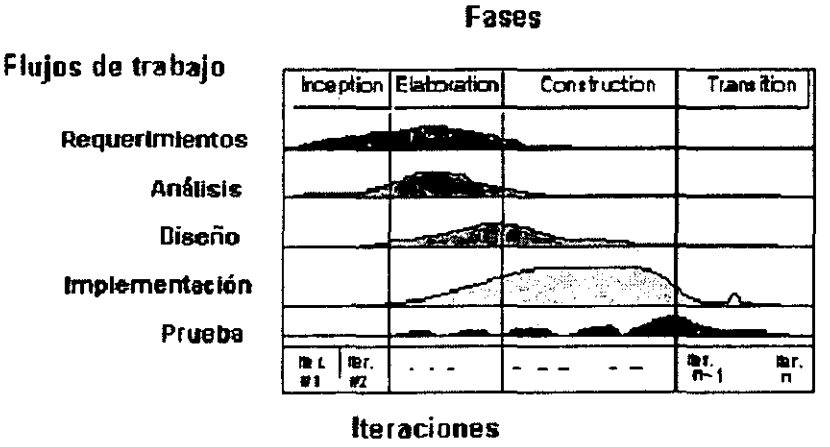


Fig 20. Las fases y los flujos de trabajo en el Proceso Unificado

En la **fase de Incepción**, se responde a las siguientes preguntas:

- ¿Qué es lo que principalmente hará el sistema para cada uno de sus usuarios más importantes?
- ¿A qué se parecerá la arquitectura de este sistema?
- ¿Cuál es el plan y cuánto costará desarrollar el producto?

Para la primera pregunta se obtendrá el modelo de casos de uso que contenga los casos de uso más críticos. En esta etapa la arquitectura aún es tentativa.

Los riesgos más importantes serán identificados y clasificados por prioridad.

Durante la **fase de Elaboración**, la mayoría de los casos de uso son especificados en detalle y se diseña la arquitectura del sistema. La relación entre la arquitectura del sistema y el sistema mismo es muy importante.

La arquitectura es análoga a un esqueleto cubierto con piel pero con un mínimo de músculo (el software) entre el hueso y la piel. Apenas el suficiente músculo como para permitir los movimientos básicos. El sistema será como un cuerpo completo, con esqueleto, piel y músculos.

Así entonces, la arquitectura está expresada como vistas de todos los modelos del sistema, los cuales en su conjunto representan a todo el sistema. Esto implica que hay vistas arquitectónicas del modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación y modelo de instalación.

La vista del modelo de implementación incluye componentes para probar que la arquitectura es ejecutable. El resultado de esta fase es una línea base arquitectónica[17]. Al final de la fase de elaboración, el administrador del proyecto estará en posición de planear las actividades y estimar los recursos requeridos para completar el proyecto.

Aquí la pregunta es, ¿son suficientemente estables los casos de uso, la arquitectura y los planes, y están los riesgos bajo el control suficiente como para cumplir todo el trabajo del desarrollo según el contrato?

Durante la **fase de Construcción**, el producto se construye (el músculo que se agrega al esqueleto-arquitectura). En ésta fase las líneas base arquitectónicas crecen para convertirse en un auténtico sistema.

Durante ésta fase del desarrollo, el monto de recursos requeridos aumenta. La arquitectura del sistema es estable dado que los desarrolladores pueden descubrir mejores formas de estructurar el sistema, o sugerir cambios menores a la arquitectura.

Al final de ésta fase, el producto contiene todos los casos de uso que se acordaron con el usuario para desarrollar ésta versión. Posiblemente se encuentren varios defectos.

Aquí la pregunta hito es: ¿cumple el producto suficientemente las necesidades del usuario como para hacer una entrega temprana del mismo?

---

[17] Un conjunto de artefactos o productos revisados y aprobados que (1) representen bases acordadas para evolución y desarrollo, y (2) puedan ser cambiadas solo a través de un procedimiento formal tal como la administración de configuración y de cambios.

La **fase de Transición** cubre el periodo durante el cual el producto se mueve a su versión beta. En la versión beta, un número reducido de usuarios experimentados trabaja con el producto y reporta sus defectos y deficiencias. Los desarrolladores corrigen los problemas reportados e incorporan algunas mejoras sugeridas para la versión general que será usada por una comunidad más grande.

La fase de transición involucra actividades como manufacturación, capacitación del personal del cliente, proveer asistencia de ayuda, y corrección de defectos encontrados después de la entrega.

9. Los flujos de trabajo.

En cada iteración de cada fase, las actividades se conducen por los llamados flujos de trabajo.

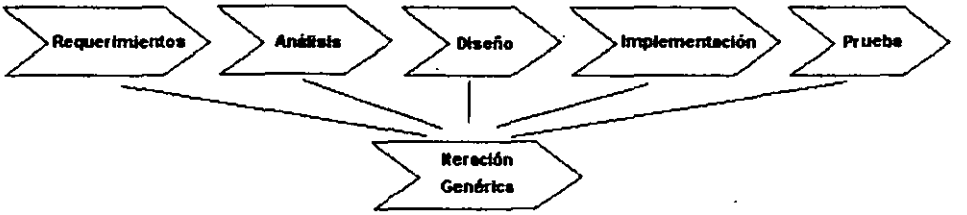


Fig 21. Los Flujos de Trabajo

El **flujo de trabajo de los requerimientos**, señala que se deben capturar estos a través de la correcta construcción de los casos de uso. Para cada caso de uso del modelo de casos de uso, se identificará un caso de uso realizado en las fases de análisis y diseño y un conjunto de casos de prueba en la fase de pruebas. De esta manera los casos de uso servirán como enlace para todos los demás flujos de trabajo.

En **el flujo de trabajo del análisis** el propósito principal es analizar los requerimientos tal como se capturaron en el flujo anterior. Se refinan y estructuran. Esto se hace para lograr una comprensión más precisa de los requerimientos y para construir una descripción de los requerimientos que sea fácilmente mantenible y que sea de ayuda para dar una estructura de todo el sistema, incluida su arquitectura. El resultado del flujo de trabajo del análisis es el modelo de análisis.

El modelo de análisis comprende entre otras cosas los siguientes productos: análisis de los paquetes (sus dependencias y contenidos), análisis de las clases (sus responsabilidades, atributos, relaciones y requerimientos especiales), realizaciones de casos de uso (se describen los casos de uso en términos de colaboraciones) y la vista arquitectónica del análisis del modelo.

En **el flujo de trabajo de diseño** el propósito principal es construir los modelos de los requerimientos no funcionales así como la solución del dominio. El resultado es el modelo de diseño, que es un bosquejo de la implementación. El modelo de diseño incluye: diseños de los subsistemas y sus dependencias, interfases y contenidos; diseño de todas las clases; y la vista arquitectónica del modelo de diseño

En **el flujo de trabajo de la implementación** el propósito esencial es la implementación del sistema en términos de componentes, es decir código fuente, archivos binarios o ejecutables y otros. El resultado es el modelo de implementación. El modelo de implementación es la entrada primaria de todas las actividades subsecuentes de pruebas.

Finalmente, en **el flujo de trabajo de pruebas**, el propósito es verificar el resultado de la implementación. El resultado en un plan de pruebas, es el modelo de pruebas, que describe como debe ser probado un sistema. El modelo incluye los casos de prueba, que especifican que es lo que se debe probar; los procedimientos de prueba, que dicen como hacer las pruebas; los componentes de prueba, que automatizan los procedimientos de prueba.

10. Más sobre la Fase de Transición

En la fase de transición ya tenemos una versión operacional del sistema, es decir ya están ejecutándose algunas de las funcionalidades principales. El administrador del proyecto ha estimado que ya puede operar en el ambiente de trabajo del usuario, sin embargo no está aún exento de errores.

Para esta fase se deja el sistema al usuario para que haga las pruebas que considere necesarias. Es factible que se detecte que faltó por mencionar alguna de las necesidades o restricciones, es decir que el usuario detecta estas faltas en forma retrasada. Una vez que se detectan errores se hace una lista de modificaciones. Entonces se ponderan y se deciden las acciones a seguir.

Sin son modificaciones que puedan considerarse pequeñas, es decir que no afecten demasiado el plan general del desarrollo, entonces se trabaja en resolverlas; pero en caso de que sean modificaciones muy grandes, se dejan pendientes para una próxima versión trabajando un ciclo completo de nuevo.

La intención de ésta fase es cumplir con los requerimientos a satisfacción de todos los involucrados con el sistema, así como asegurarse de que el sistema funcione bien en el ambiente de operación, incluyendo la corrección de defectos que reportaron los usuarios que prueban el sistema.

Esta fase termina con una versión del sistema, funcionando según lo convenido con el cliente, quedando satisfecha la mayor parte de las necesidades planteadas al inicio del desarrollo.

Debe tenerse siempre en cuenta que ningún producto de software es perfecto, así que seguramente habrá que hacer algo de retrabajo, el cual no debe verse como una carga sino como una actividad

normal. Al igual que las demás fases, la fase de transición debe ser planeada tomando en consideración lo anteriormente dicho.

Normalmente la cantidad de personal necesario para cubrir esta fase es tanta como la que se utilizó en la etapa de construcción.

Pruebas en la fase de transición

Se deben concentrar las pruebas en cinco aspectos:

1. Los "*probadores*" deben cubrir las funciones claves de todos los casos de uso.
2. El producto debe pasar satisfactoriamente las pruebas hechas por el cliente.
3. El material del usuario debe ser de calidad aceptable.
4. Disponibilidad del material para usar el sistema.
5. Verificar que tanto el cliente como los usuarios estén satisfechos con el producto.

La actividad en los flujos de trabajo es menor en esta fase, pero esto no significa que es menos trabajo. La atención principal está en arreglar los defectos y eliminar las fallas encontradas en la prueba del ambiente usado, así como asegurarse que son correctas las reparaciones y hacer pruebas de regresión para asegurarse que las reparaciones no introdujeron nuevos errores.

Una vez más hay actividades en los cinco flujos de trabajo principales: requerimientos, análisis, diseño, implementación y prueba. Y los más importantes en la fase son los de implementación y prueba, porque es aquí es donde se detectan, arreglan y prueban los defectos. Pocas veces hay que rediseñar para arreglar los defectos, así que el trabajo en el flujo de diseño es poco. Los requerimientos son afectados muy mínimamente.

Se deben llevar a cabo cuatro actividades en forma paralela:

- Los cinco flujos de trabajo de la iteración en turno.
- La planeación de las iteraciones.
- Hacer mas consideraciones del caso del negocio
- Medir los resultados de la operación

¿Qué se hace exactamente en la fase de transición?

Las actividades a llevar a cabo en la fase de transición son:

- Preparar la versión beta a partir de la versión de la fase de construcción.

- Instalar la versión en el lugar apropiado, junto con actividades posiblemente de migración de datos de sistemas anteriores.

- Responder a los resultados de las pruebas que se vayan haciendo.

- Completar los productos del proyecto.

- Determinar cuando termina un proyecto.

Para obtener la versión beta del sistema, se eligen a los usuarios que harán de "*probadores*", y se les dan instrucciones detalladas de qué probar y como reportar los resultados y observaciones.

Para cuando se instale la versión beta, debe haber instrucciones especificas para hacerlo

Todos los defectos y errores y sugerencias se deberán recolectar a fin de dar una acción de respuesta. Hay dos clases de respuestas: las sencillas que sólo requieren correcciones a fallas menores en el código, o las que provocan cambios sustánciales en la arquitectura del sistema.

Existen dos tipos de relaciones entre el mercado y la organización que hace el software, según los productos que genere: los productos para el mercado (relación uno a muchos) y los productos para un cliente específico (relación uno a uno).

En el caso de tratarse de productos para el mercado, se procurará poner más documentación para las pruebas beta porque se supone que tienen menos experiencia que los "*probadores*" experimentados. Así también se harán diferentes versiones para diversos sistemas operativos, y algunas variaciones como lenguaje, unidades de medida, país, etc.

En el caso de productos para un cliente en específico las diferencias estarían principalmente en:

- La representación del cliente habrá estado participando desde las primeras fases, dando la retroalimentación necesaria.

- Habrán estado observando las pruebas bajo las premisas establecidas en el contrato.

- Habrán estado participando en las sesiones de medición de progreso así como la construcción de los *hitos*.

- La organización habrá estado ayudando para instalar el sistema en un sitio del cliente.

- Las pruebas de aceptación terminan cuando el cliente y la organización llegan a un acuerdo para sobre el cumplimiento de los requerimientos que habían acordado previamente.

Hay ocasiones en que es necesario hacer migraciones o conversiones de datos desde un sistema anterior; en este caso se deberán tomar las siguiente responsabilidades:

- Garantizar al cliente que todas las operaciones del sistema anterior, quedaron cubiertas por el nuevo, incluso haciendo que por un periodo corran la mismo tiempos ambos sistemas.

- Transferir los datos del sistema anterior al nuevo, incluyendo posiblemente cambio de formatos.

- Además de dar las instrucciones para hacer la transferencia, los documentos deben tener pruebas para verificar que la instalación fue exitosa.

¿Cuándo se termina la fase de transición?

La fase de transición se termina hasta que todos los productos del proyecto se han terminado. Finalmente se verificará que todos los productos son consistentes unos con otros. Pero el criterio principal es cuando el cliente esta satisfecho.

Ya terminada la fase de transición, se pasa la responsabilidad del mantenimiento a otra organización de apoyo.

¿Cómo se determina si el proyecto fue exitoso en términos monetarios?

Si se hizo el producto para un cliente, se verifica si el precio del contrato cubrió totalmente los gastos del desarrollo del proyecto.

En el caso de desarrollar un sistema para el mercado, el éxito estará dado por las metas que se haya marcado la compañía según el capital invertido en el desarrollo.

Midiendo la fase de transición

Dado que ésta es la última fase del ciclo, la medición de resultados es diferente; en primera porque ya no hay otra fase a la cual se le pase algún producto de trabajo y porque es posible que después haya otros ciclos, se deberá recolectar la información que sea útil para ello.

Las búsquedas deberán ser de dos tipos:

- Medición de iteraciones y fases.

- Mediciones posteriores al proyecto. (*postmortem*)

Para el primer caso, si las tres fases anteriores fueron bien hechas, ésta última fase será relativamente sencilla y sólo se corregirán algunos defectos fácilmente corregidos.

Pero si por otro lado no hubo un a correcta identificación de riesgos, o se falló en el diseño de la arquitectura, entonces ésta fase será significativamente difícil. Entonces el administrador del proyecto extenderá la fase de transición para tener un sistema mínimamente satisfactorio.

El proceso de tomar mediciones es para recolectar información que ayude a hacer mejor las cosas en una próxima versión. Las mediciones no deben ser la base para acciones negativas personales.

Para el segundo caso, se analiza que hizo bien y que hizo mal la organización que llevó el proyecto. Lo que se busca es registrar información útil para el desarrollo de otros proyectos y que sirva para hacer procesos de desarrollo más efectivos y exitosos. Por ejemplo se pueden registrar las razones para tomar un diseño en lugar de otro, así como mencionar puntos que ayuden al proceso mismo.

III. El Modelo de Madurez de Capacidades

1. Los procesos de desarrollo del software

La capacidad de desarrollar y entregar software confiable dentro de los tiempos y costos planeados, sigue siendo un dolor de cabeza para muchas organizaciones. La mayoría de las veces, los proyectos son excesivamente retardados y sobre presupuestados, y los beneficios de las mejores herramientas y métodos no pueden ser llevados a cabo dentro del remolino de un proyecto caótico e indisciplinado.

Algunos administradores de la industria del software toman la siguiente actitud: "Prefiero que el proyecto esté mal a que este retardado, ya lo arreglaremos después", lo cual origina muchos problemas para el cliente.

El Modelo de Madurez de Capacidades (*Capability Maturity Model*, que llamaremos en adelante sólo CMM) del software, es una estructura desarrollada por el *Software Engineering Institute*[18], que describe los elementos clave de un proceso de software efectivo. Describe un camino evolutivo de mejoramiento desde un proceso deficiente (o *ad hoc*), hacia uno maduro y disciplinado.

Este camino se transita por cinco niveles de madurez.

El CMM cubre prácticas para planeación, ingeniería, administración de desarrollo de software y mantenimiento. Cuando se siguen estas prácticas, se mejora la habilidad de las organizaciones para cumplir las metas de costo, calendarización, funcionalidad y calidad del producto.

El CMM guía a las organizaciones de software que quieren tener más control de sus procesos para desarrollar o dar mantenimiento al software y evolucionar hacia una cultura de excelencia de administración e ingeniería de software.

El CMM tiene las siguientes características generales que motivan a su estudio:

- Está basado en las prácticas reales.
- Refleja lo mejor de las prácticas de desarrollo.
- Refleja las necesidades de mejorar los procesos de desarrollo de software.
- Está documentado, y
- Está disponible públicamente.

---

[18] Véase [WebSEI]

2. Los tipos de organizaciones dedicadas al software.

Las organizaciones inmaduras generalmente realizan procesos de software improvisados durante el curso del proyecto, o si tienen un proceso establecido, no lo siguen rigurosamente. Este tipo de organizaciones es "reaccionario" (en el sentido de que *reaccionan* según lo que suceda en el momento), sus administradores se concentran en resolver crisis inmediatas (en cuyo caso se les conoce como *bomberos*), además están siempre apretadas de tiempo y excedidas de presupuesto ya que los planes no se basan en estimaciones realistas.

En contraste una organización madura, tiene la organización suficiente para administrar el desarrollo de software y los procesos de mantenimiento. Comunica con exactitud los procesos del software tanto para el personal actual como para nuevos empleados, y lleva a cabo actividades de trabajo de acuerdo a procesos planeados.

3. Conceptos fundamentales del proceso de madurez.

Un **proceso** es una secuencia de pasos ejecutados para un propósito dado. El proceso es lo que hacemos, usando procedimientos, métodos, herramientas y equipo, para transformar material en bruto (entradas) en un producto (salida) valioso para los clientes. Solo cuando las actividades son ejecutadas o los métodos son usados, se puede hablar con exactitud de un proceso, es decir, una descripción del proceso no es un proceso.

Un **proceso de software** puede ser definido como un conjunto de actividades, métodos, prácticas y transformaciones que la gente emplea para desarrollar y mantener[19] software o productos asociados.

La capacidad del proceso de software describe el rango de los resultados esperados que pueden ser logrados al seguir el proceso de software.

El **desempeño del proceso de software** representa los resultados reales alcanzados al seguir un proceso de software, así que se enfoca en los resultados logrados, mientras que la capacidad del proceso de software se enfoca en los resultados esperados.

La madurez del proceso de software es la explicación de un proceso específico, explícitamente definido, administrado, medido, controlado y efectivo.

La institucionalización es la construcción de infraestructura y cultura que apoya los métodos, prácticas y procedimientos que son la forma continua de hacer negocios.

---

[19] cuando se habla de "mantener" un sistema de software se refiere al hecho de hacer el mantenimiento necesario, con las modificaciones que agregan funcionalidades o corrigen defectos.

4. La Calidad en el CMM

Para lograr consenso entre la administración y el equipo profesional sobre cuales actividades de mejoramiento emprender primero, se debe tener una estrategia organizada para mejora.

El CMM ordena las etapas para que cada una ofrezca un fundamento sobre el cual construir mejoras en la próxima etapa. Así en una estrategia de mejora vista desde una estructura[20] de proceso de madurez de software, provee un camino a seguir para el proceso de mejora continua.

El CMM es una aplicación de los conceptos de la administración de la calidad total al software. La administración de calidad total[21] se define como la aplicación de métodos cuantitativos y recursos humanos que mejoran los materiales y servicios provistos como entradas para una organización y para mejorar todos los procesos dentro de la organización; la meta es cumplir con todas las necesidades del cliente, ahora y en el futuro.

5. Aclaraciones útiles sobre el uso del CMM

El CMM no es una descripción exhaustiva del proceso de software, ya que sólo proporciona una estructura conceptual para mejorar la administración y el desarrollo de productos de software en una forma disciplinada y consistente. No garantiza que los productos de software serán construidos exitosamente, ni que los problemas de Ingeniería de Software serán resueltos adecuadamente.

El CMM sólo identifica prácticas para un proceso maduro de software y proporciona ejemplos del estado del arte[22], pero no significa que sea exhaustivo o dictatorial.

El CMM es sólo una herramienta para ayudar a las organizaciones para mejorar sus procesos de software. La inteligencia, la experiencia y el conocimiento deben dar forma a una apropiada interpretación del CMM en un ambiente específico.

Si se aplica memorísticamente o como chequeo de lista, puede resultar perjudicial para cualquier organización en lugar de ayudarla. Se debe tener compromiso a largo plazo en los procesos de mejora continua, a fin de alcanzar niveles más altos de madurez en el proceso de software.

---

[20] *Framework* en inglés.

[21] *Total Quality Management* en inglés.

[22] Estado del arte es un tópico que se refiere a los conocimientos o estándares más actuales en la época en que se escribe.

6. Los niveles del CMM

Un **nivel de madurez** es una plataforma bien definida para evolucionar hacia la consecución de un proceso de software maduro. Cada nivel de madurez tiene un conjunto de metas de procesos que, cuando son conseguidos, estabilizan un componente importante del proceso de software.

La organización del CMM en cinco niveles, clasifica la prioridad de las acciones de mejora para incrementar la madurez del proceso de software.

Los **cinco niveles** pueden ser descritos brevemente así:

Nivel 1: El nivel inicial

El proceso de software es caracterizado por ser *ad hoc* (adaptado únicamente para servir en el momento), y usualmente caótico. Pocos procesos están definidos y el éxito depende de los esfuerzos individuales y heroicos.

En el nivel inicial, la organización no cuenta con un ambiente estable de desarrollo y mantenimiento de software. Durante las crisis, los proyectos abandonan los procedimientos planeados y se regresan a codificar y probar. Ocasionalmente la fuerza y capacidad de los administradores puede resistir a las presiones para tomar atajos en el proceso de software, pero cuando dejan el proyecto, su influencia de estabilización se va también con ellos.

El éxito en las organizaciones en el nivel 1, depende de la eficiencia y heroísmo de las personas en la organización y no puede ser repetido a menos que los mismos individuos eficientes sean asignados al próximo proyecto. Este nivel se caracteriza por ser **inestable**.

Nivel 2: El nivel repetible

Los procesos básicos de administración del proyecto son establecidos para rastrear costos, calendarización y funcionalidad. La disciplina del proceso es tal que se puede repetir el éxito en proyectos similares.

En el nivel repetible, se establecen políticas para administración del proyecto de software, así como procedimientos para hacerlo. La capacidad del proceso esta sustentada al establecer procedimientos básicos de disciplina en la administración sobre el proyecto. Los proyectos usan procesos efectivos que son definidos, documentados, practicados, capacitados, medidos, reforzados y mejorables.

Los proyectos, en organizaciones que están en el nivel 2, han instalado controles básicos de administración de software. Son definidos los estándares de proyectos de software, y las organizaciones los siguen con exactitud.

La capacidad del proceso de software en las organizaciones en el nivel 2, puede ser resumida como **disciplinada,** porque la planeación y el rastreo del proyecto de software son estables, y los éxitos recientes pueden ser repetidos. El proceso del proyecto está bajo control efectivo de un sistema de administración de proyecto, basándose en planes realistas según el desempeño de proyectos previos.

Nivel 3: El nivel definido

El proceso de software tanto para las actividades de administración como las de ingeniería, está documentado, estandarizadas e integradas en un proceso de software modelo para la organización. Todos los proyectos utilizan la versión aprobada y hecha a la medida del proceso de software de la organización para el desarrollo y mantenimiento del software.

En el nivel definido, está documentado un proceso (o procesos) estándar para desarrollo y mantenimiento de software, y es usado en toda la organización, que explota las prácticas efectivas de Ingeniería de Software resultado de la estandarización de sus procesos de software.

La capacidad del proceso de software en organizaciones en el nivel 3, puede ser resumida como **estándar y consistente**, porque las actividades de ingeniería de software y de administración, son estables y repetibles. Dentro de las líneas del producto establecidas, el costo, la calendarización y la funcionalidad están bajo control, y la calidad del software es rastreada.

Ésta capacidad del proceso está basada en una comprensión común, de amplio uso en la empresa, de las actividades, roles y responsabilidades en el proceso de software definido.

Nivel 4: El nivel administrado

Se hacen mediciones detalladas del proceso de software y la calidad del producto. Tanto el proceso de software como los productos son controlados y entendidos cuantitativamente.

En el nivel administrado, la organización establece metas cuantitativas de calidad para los proceso y para los productos. Como parte de un programa organizacional de medición, la productividad y la calidad son medidas en las actividades importantes del proceso de software en todos los proyectos. Se usa una base de datos al nivel de toda la organización del proceso de software para recoger y analizar los datos disponibles de los procesos de software definidos en los proyectos.

La capacidad del proceso de software en las organizaciones en nivel 4, se puede resumir como **capacidad que está siendo cuantificada y es predecible** porque los procesos son medidos y operan dentro de límites cuantitativos.

Este nivel de capacidad del proceso permite a una organización predecir hacia donde va en el proceso y la calidad del producto dentro de ciertos límites cuantitativos. Se puede predecir que los productos de software serán de alta calidad.

Nivel 5: El nivel de optimización

Se habilita un proceso continuo de mejoramiento con retroalimentación desde los procesos y desde nuevas ideas en planes piloto y de otras tecnologías.

En el nivel de optimización, toda la organización está enfocada en un proceso continuo de mejoramiento. La organización tiene la capacidad de identificar debilidades y fortalezas del proceso, con el objetivo de prevenir la ocurrencia de defectos.

La capacidad del proceso de software en las organizaciones en nivel 5 puede ser resumida como **mejora continua**, porque están haciendo lo posible por mejorar el rango de sus capacidades de proceso, y así mejorar el desempeño del mismo. Las mejoras ocurren por los avances por incrementos en el proceso existente y por las innovaciones usando las nuevas tecnologías y métodos.

A través de los cinco niveles, la capacidad del proceso interactúa con las personas, la tecnología y las mediciones como se ve en la siguiente tabla.

	Nivel 1	Nivel 2	Nivel 3	Nivel 4	Nivel 5
Procesos	Existen o son usados procesos poco estables. "Solo hazlo"	La estimación es estable y documentada y los procesos comprometidos están al nivel del proyecto. Los problemas son reconocidos y corregidos cuando ocurren.	Se usan administración integrada y procesos de ingeniería en la organización. Los problemas son anticipados y prevenidos o sus impactos son minimizados	Los procesos son cuantitativamente comprendidos y estabilizados. Las fuentes de los problemas individuales son comprendidas y eliminadas.	Los procesos son mejorados continua y sistemáticamente. Las fuentes de problemas comunes son comprendidas y eliminadas.
Personal	El éxito depende de individuos heroicos. "Los	El éxito depende de individuos. Hay apoyo de sistemas de administración.	Los grupos de proyecto trabajan juntos, quizá como un producto de equipo integrado	Dentro de cada proyecto existe un fuerte sentido de trabajo en equipo.	En toda la organización existe un fuerte sentido de trabajo en equipo.

nas	bomberazos" son usuales. Las relaciones entre disciplinas no están coordinadas.	. Los compromisos son entendidos y administrados. El personal está capacitado.	integrado La capacitación está planeada y provista acorde a los roles.		Cada uno está involucrado en la mejora de procesos.
Tecnología	La introducción de nueva tecnología es riesgosa.	La tecnología apoya las actividades establecidas y estables.	Las nuevas tecnologías son evaluadas sobre bases cuantitativas.	Las nuevas tecnologías son evaluadas sobre bases cuantitativas.	Las nuevas tecnologías son usadas e implantadas.
Medicciones	La recolección de datos y análisis son *ad hoc*	Para proyectos individuales son usados datos de planeación y administración.	Los datos son recogidos y usados en todos los procesos definidos. Los datos son compartidos sistemáticamente compartidos en todos los proyectos.	La definición de datos y los conjuntos son estándares en toda la organización Los datos son usados para comprender el proceso cualitativamente y estabilizarlo.	Los datos son usados para evaluar y seleccionar procesos de mejora.

7. La estructura interna de los niveles de madurez

Cada nivel de madurez ha sido descompuesto en partes constituyentes.

Cada uno de los niveles está compuesto de varias **áreas claves del proceso** y cada área clave del proceso está organizada en cinco secciones llamadas **características comunes**.

Las características comunes contienen las **prácticas clave** que, cuando son aplicadas conjuntamente, cumplen con las **metas** del área clave del proceso.

Un nivel de madurez es un a plataforma que puede evolucionar hacia la consecución de un maduro proceso de software. Cada nivel de madurez indica un nivel de capacidad.

La siguiente figura muestra la estructura del CMM:

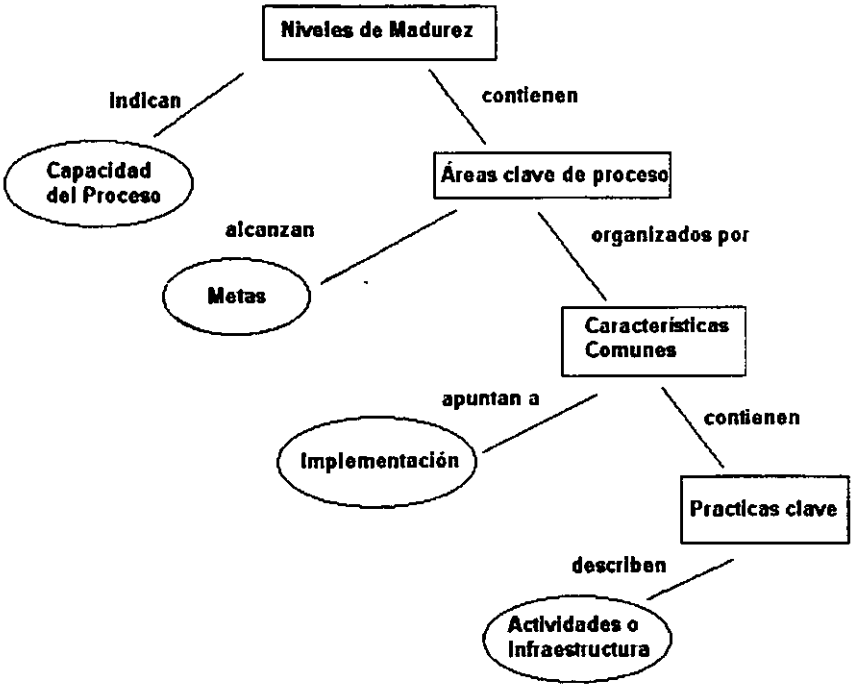


Fig 22. Estrucura del CMM

Las áreas claves del proceso

Las áreas claves del proceso identifican los temas que deben ser atendidos para lograr un nivel de madurez.

Cada área clave del proceso identifica un conjunto ordenado de actividades relacionadas que, cuando son ejecutadas colectivamente, logran una serie de metas consideradas importantes para acrecentar la capacidad del software.

Se describen en el CMM sólo aquellas áreas que se han identificado como definitivamente claves en la capacidad del proceso.

Las áreas claves del proceso pueden ser consideradas los requerimientos para alcanzar un nivel de madurez. Para alcanzar un nivel de madurez, las áreas claves del proceso para dicho nivel (y los niveles más bajos), deben ser satisfechos y los procesos deben ser institucionalizados.

Las **metas** de cada área clave del proceso resumen sus prácticas claves y pueden ser usadas para determinar si se ha implementado ya efectivamente en una organización, la correspondiente área clave del proceso.

Las Áreas Clave del Proceso en el Nivel 2

Las áreas claves del proceso en el nivel 2 se enfocan en los asuntos del proyecto relativos a establecer controles básicos de administración de proyecto. Son las siguientes:

- Administración de requerimientos
- Planeación del proyecto de software
- Seguimiento y supervisión del proyecto de software
- Administración de subcontratistas de software
- Aseguramiento de la calidad del software
- Administración de la configuración

El propósito de **la Administración de Requerimientos** es establecer una comprensión común entre el cliente y los requerimientos del mismo que serán satisfechos por el proyecto de software. Este acuerdo con el cliente es la base para planear y administrar el proyecto de software.

La revisión los requerimientos asignados al software e interactuar con el cliente (sea interno o externo) es parte del establecimiento de tal comprensión. La documentación y el control de los requerimientos del cliente es un prerrequisito para basar la estimación, planeación, ejecución y seguimiento de las actividades del proyecto a través del ciclo de vida del software.

El propósito de **la Planeación del Proyecto de Software** es establecer planes razonables para realizar la ingeniería de software y para administrar el proyecto de software. Los planes empiezan con una frase del trabajo y las restricciones y metas que definen los alcances del proyecto de software.

Los procesos de la planeación del software incluyen los pasos para estimar el tamaño de los productos de trabajo y los recursos necesarios, producir un calendario, medir los riesgos del software, y negociar los compromisos. El plan es documentado y mantenido como una herramienta necesaria para administrar el proyecto de software.

El propósito de la **Supervisión y Seguimiento del Proyecto de Software** es establecer la visibilidad adecuada en el progreso real para que la administración tome las acciones efectivas cuando la ejecución del proyecto de software se desvíe significativamente de los planes del software.

La administración del proyecto de software debe estar basada en el plan de desarrollo del software. La administración involucra el seguimiento y la revisión de los resultados del software contra el plan y toma las acciones correctivas necesarias según los logros y resultados reales.

Estas acciones pueden incluir la revisión del plan de desarrollo del software para reflejar los logros reales, re planeación del trabajo que se está haciendo, y/o tomar las acciones para mejorar el desempeño.

El propósito de la **Administración de Subcontratos del Software** es seleccionar a los subcontratistas calificados y manejarlos efectivamente. Los subcontratistas deben ser seleccionados basándose en alianzas estratégicas de negocios.

El propósito del **Aseguramiento de la Calidad del Software** es proveer la administración con la visibilidad apropiada dentro del proceso que se esté usando en el proyecto de software y los productos que se están construyendo.

Esta visibilidad es lograda al revisar y auditar los productos de software así como las actividades para verificar que se cumple con los estándares y procedimientos aplicables. Cuando existan problemas se deberán resolver si es posible dentro del mismo proyecto del software; si no es así se deberá "escalar" el problema a donde sea apropiado para encontrar su solución.

El propósito de la **Administración de la Configuración del Software** es establecer y mantener la integridad de los productos del proyecto de software a través del ciclo de vida del proyecto.

La integridad de los productos de trabajo se logra al identificar la configuración del software (i.e. los productos de trabajo seleccionados) en ciertos puntos del tiempo, controlando sistemáticamente los cambios de la configuración, y manteniendo la integridad y seguimiento de la configuración a través del ciclo de vida del software.

Prácticas claves

Las prácticas claves describen las actividades e infraestructura que más contribuye a una implementación efectiva e institucionalización del área clave del proceso. Las prácticas especificas a ser ejecutadas en cada área clave del proceso se desarrollarán conforme la organización alcance niveles más altos de madurez.

Las prácticas claves describen "que" debe ser hecho, pero no deben tomarse como dictados de "cómo" debe ser implementado el proceso. Las prácticas claves deben ser interpretadas racionalmente para juzgar si las metas del área clave del proceso son logradas efectivamente.

Características comunes

Las prácticas que describen las áreas claves del proceso están organizadas en características comunes. Las características comunes son atributos que indican si la implementación o institucionalización de un área clave del proceso es efectiva, repetible y durable.

Existen cinco características comunes:

Compromisos a ejecutar.

Describe las acciones que la organización debe tomar para asegurar que el proceso está establecido y que va a durar. Usualmente involucra el establecimiento de políticas organizacionales y liderazgo.

Habilidad a desempeñar.

Describe las precondiciones que deben existir en el proyecto u organización para implementar competentemente el proceso de software. Involucra recursos, estructuras organizacionales y capacitación.

Actividades ejecutadas.

Describe las actividades, roles y procedimientos necesarios para implementar un área clave del proceso. Normalmente involucra establecimiento de planes y procedimientos, ejecución del trabajo, su seguimiento y la toma de acciones correctivas cuando sea necesario.

Las prácticas en la característica común de *Actividades Ejecutadas* describe lo que debe ser implementado para establecer una capacidad del proceso.

Medición y análisis.

Describe las prácticas básicas de medición que son necesarias para determinar el estado del proyecto. Estas mediciones son usadas para controlar y mejorar el proceso.

Verificación e implementación.

Describe los pasos para asegurar que las actividades son ejecutadas de acuerdo con el proceso que ha sido establecido. Típicamente implica revisar y auditar por medio de la administración y el aseguramiento de la calidad del software.

En resumen, el CMM explica con detalle las características de los niveles de madurez en que se encuentra o puede encontrarse una organización dedicada al desarrollo de software; detalla las prácticas recomendadas para mejorar el proceso. Las prácticas deben cambiar para reflejar las nuevas experiencias y metodologías.

Más que una guía para mejorar procesos, el CMM provee una base para crecer. El proceso de software debe ser pragmático y ajustable, o no será usado. Es por eso que el **CMM es una guía y no reglas**. Su objetivo es establecer metas para cada nivel de madurez y cada área clave, así como también proveer ejemplos de cada práctica.

Ninguna organización debe intentar hacer todo lo que se dice. Hacerlo resolvería algunos problemas y haría fracasar otros. Debe usarse lo que satisfaga las necesidades, pero poniendo más atención en las metas.

CAPÍTULO 3

DESARROLLO DE LA PRIMER VERSIÓN DEL SISTEMA

1. Determinación de equipos de trabajo

Facilitando la conformación de los equipos mediante un ejercicio

Desde la primer sesión del curso, se plantearon a grandes rasgos los objetivos del sistema que se quería construir. En la segunda sesión se realizó un ejercicio al aire libre de tipo lúdico y demostrativo, con el fin de resaltar las actitudes y características necesarias para un trabajo en equipo.

Las actividades realizadas fueron básicamente dos. Entre todos los asistentes y con los ojos vendados, se construyó un cuadrado con una cuerda. Después se conformaron dos equipos y se hizo pasar horizontalmente a cada miembro del equipo a través de un espacio pequeño, acotado por la misma cuerda, y sin tocar los bordes.

Una vez terminadas las tareas, se comentaron las actitudes tomadas por cada persona sin mencionarse los nombres, como arquetipos de conducta que puede encontrarse en cualquier integrante de un equipo de trabajo, señalando causas, virtudes, utilidades, negligencias, etcétera.

Se mencionaron en general todas las posturas, tanto adecuadas como evitables, que puede asumir cualquier persona que está involucrada en un trabajo en equipo.

El propósito principal del ejercicio, era hacer ver a cada cual, que actitud había tomado al intentar trabajar en equipo para lograr un objetivo común, poniendo énfasis en lo que cada cual identificaba de las personalidades arquetípicas: el indiferente, el propositivo, el que nada opina, el que prefiere esperar que otros tomen la iniciativa, el analítico, el tibiamente comprometido, el líder nato, etc.

Para cada actitud se comentaron aspectos positivos y negativos.

Desde mi punto de vista el ejercicio fue útil pero limitado, pues como se observó en el ejercicio, varias personas demostraron una conducta diferente a sus intereses o motivaciones académicas. De esta forma hubo quienes destacaron con propuestas y liderazgo organizacional, pero que eventualmente tomaron una actitud más discreta y tímida al transcurrir el desarrollo de los proyectos de software del curso.

Un objetivo secundario del ejercicio, era conocer a las demás personas, para de esta forma darse una idea del perfil con que se manejaban al hacer trabajo en equipo, pero considero que este objetivo no pudo ser logrado ya que los que participamos, éramos alumnos que teníamos al menos un año de conocernos entre sí, al menos en un plano superficial.

Los proyectos de sistemas que se plantearon

Como los alumnos inscritos al curso eran casi una veintena, se pensó en hacer dos sistemas: uno que se ocupara del proceso de inscripción y reinscripción; y otro atendiendo el proceso de inscripción al examen de admisión al posgrado.

Estableciendo número de equipos

Tomando en cuenta los sistemas antes descritos, se pensó en un primer momento en dos opciones: Una era hacer dos equipos, de 10 personas aproximadamente, donde cada equipo tomaría un proyecto. La otra opción era que cada equipo hiciera ambos proyectos para que así pudiera hacerse una comparación al final del curso.

La experiencia de las profesoras era que en cursos anteriores donde habían realizado este tipo de experiencias académicas, la mayoría de las veces, los proyectos quedaban en versiones que incluso podrían clasificarse como previas a un prototipo por su mínima funcionalidad. Esto aunado a la opinión mayoritaria de los alumnos, inclinó la elección por la primer opción.

Selección de integrantes

Se esperaba que, basándose en el ejercicio realizado en el bosque de Tlalpan, se fueran conformando los equipos según las percepciones captadas por cada persona. De esta forma hubo un grupo de alumnos que hicieron tempranamente su lista de integrantes, dejando a los restantes en el otro equipo sin capacidad de elección. Finalmente el equipo A, primero en formarse, tuvo a nueve integrantes y el equipo B con igual número

Elección de proyectos.

El equipo que se denominó A, solicitó le fuera asignado el proyecto del manejo de inscripción, mientras que el otro equipo condescendió la petición y tomó para sí, el proyecto de control de alumnos inscritos a examen de admisión.

En éste trabajo se describe el trabajo del equipo A.

2. Lineamientos iniciales para el desarrollo

Entre las objetivos iniciales para el desarrollo del sistema, se plantearon los siguientes en éste orden de importancia:

1. Trabajar con la metodología del *"Proceso Unificado"*, propuesta por Jacobson, Booch y Rumbaugh.

2. Usar el Lenguaje Unificado de Modelación UML.

3. Hacer que el sistema opere por Internet.

4. Implementar usando el lenguaje de programación Java.

5. Aplicar los procesos de administración y revisión entre colegas según el Modelo de Madurez de Capacidades (Capability Maturity Model, CMM).

Como en muchos proyectos que comienzan con cierto temor por el fracaso dada la ignorancia de recursos y actividades, iniciamos con una seria desventaja el proyecto. La metodología básica que debía ser tomada como guía de todo el desarrollo del proyecto, era desconocida por todos los miembros del grupo; así también había cierto grado de expectación por parte de las profesoras pues el Proceso Unificado era un producto relativamente nuevo en el mundo de la Ingeniería de Software.

Cabe mencionar los siguiente puntos:

- No había ningún antecedente de sistema similar realizado en algún proyecto del instituto.

- No se contaba con ningún sistema administrador de bases de datos que estuviera operando en las salas de cómputo del instituto.

- No había certeza de poder usar la infraestructura de las salas de cómputo.

3. Metodología adoptada

La metodología usada fue una resultado de tomar el Proceso Unificado y las recomendaciones del CMM, que ya ha sido presentado en el capítulo 2.

4. Formas de organización

En el Proceso Unificado se hace el trabajo con la asignación de roles para cubrir tareas específicas para cada fase del desarrollo.

Para propósitos del proyecto se tomaron los roles que sugiere el Proceso Unificado.

Equipo de administración:

- Administrador de Proyecto
- Administrador de Configuraciones
- Administrador de Calidad

Equipo de desarrollo:

- Líder de proyecto
- Analista
- Diseñador
- Programador
- Probador

Expertos en tópicos específicos (algún sistema administrador de base de datos, lenguaje Java, interfaces de usuario, redes, etc.)

Eventualmente se añadieron los roles siguientes

- Coordinador de equipo de programación
- Documentador (responsable de llevar al día los documentos de análisis y diseño)

Se determinaron primero los roles que debían existir. Basándose en esto se asignaron a las personas que conformaron el equipo. Por norma hubo una persona asignada a cada rol.

Se pensaba que el trabajo iba a estar distribuido de forma equitativa.

La designación de los roles se hizo en una reunión donde no estaban las profesoras, únicamente los participantes. Se hizo una breve mención de las actividades que debería hacer cada rol y se presupuso que ya eran claras las responsabilidades, actividades y demás compromisos que exigía cada rol.

Donde había demanda de dos o más personas para desempeñar un rol, se hizo una negociación sobre las expectativas de cada quien explicando su interés. Finalmente los roles se asignaron por consenso.

5. Planes de trabajo

Se propuso tener planes de trabajo para cada rol y se eligieron los diagramas de Gantt para presentarlos. Se debían anotar todas las tareas importantes, enmarcadas en la fase en que se encontrara el proyecto.

Se procuró agregar la información estándar para la identificación de documentos, que se describe en la siguiente sección.

6. Documentos usados para el proyecto

Para todos los documentos del proyecto se utilizó un par de rectángulos a modo de mecanismo de estandarización e identificación.

En el encabezado[23] teníamos un rectángulo con cuatro espacios para la siguiente información:

- *Entregable*, con el nombre descriptivo del documento.
- *Referencia*, con el nombre del archivo electrónico.
- *Versión*, que indicaba la versión del documento.
- *Pág*, que mostraba el número de página, seguido de diagonal seguido del total de páginas.

Para el pie de página[24], se tenía otro rectángulo conteniendo lo siguiente:

- *Fase*, indicando la fase en que se encontrara el proyecto según el Proceso Unificado.
- *Sistema*, que invariablemente fue Sistema de Inscripción al Posgrado en Ciencia e Ingeniería de la Computación de la UNAM.

---

[23] Refierase al apéndice A inciso b

[24] Refiérase al apéndice A, inciso c para ver un ejemplo

- *Estado del documento*, representado por tres cuadros que llevarían la señal X según el estado del documento; B borrador, R en revisión y A aprobado.

- *Etapa*, en que se hizo el documento, según el Proceso Unificado.

- *Realizó*, con el nombre de quien realizó el documento.

- *Fecha* y *Nombres de líder de proyecto, administrador de calidad y administrador del proyecto*, con un espacio para poner la firma de cada uno.

Los documentos que se generaron fueron esencialmente de dos tipos: los de análisis, diseño, reportes por un lado, y los planes de trabajo por otro lado. Esto correspondió a los tipos herramienta con que se crearon los documentos. En el primero caso un procesador de palabras y en el otro el *Project*.

El objetivo primario en la adopción de estándares para los documentos, es facilitar el manejo de los mismos, y propiciar una forma útil y sencilla de intercambiar información. Así también auxiliar la búsqueda de información de algún documento necesario en cierto momento.

Tomando lo anterior en consideración, se puede decir que el objetivo fue cubierto a medias, pues hubo momentos en que el manejo de tantos documentos presentó una carga de trabajo más allá de lo esperada para el rol de administración de configuraciones.

7. Historia sucinta del proceso de desarrollo del SIPOCIC
En la fase de incepción.

El principio fue muy caótico dado el desconocimiento de las actividades correspondientes a cada rol, así como de los lineamientos generales del Proceso Unificado. Conforme avanzó el proyecto se fueron aclarando las responsabilidades y obligaciones de cada rol.

En la fase de elaboración.

Dominada por excesivas reuniones y con poco avance en términos generales. Existieron muchas ideas que iban y venían y eran discutidas hasta donde era posible. Hubo momentos de incertidumbre ante las propuestas que podrían ser las mejores.

Se consideraron varias plataformas y herramientas para hacer la parte técnica de la conexión a la base de datos desde Java.

Aun así hubo muchos puntos que quedaron en el aire para ser analizados y diseñados después. Esto se capitalizó como un error de todo el equipo porque las decisiones no tomadas y los aspectos técnicos que no se exploraron a fondo, se convirtieron después en una carga pesada.

En la fase de construcción.

La etapa más pesada del trabajo, con poco tiempo para ser hecha y con graves dificultades técnicas. Faltó mejor coordinación con los encargados de los roles de diseño ya que los documentos básicos para arrancar ésta fase, fueron entregados tiempo después de iniciada, lo que provocó desconcierto y cambios de última hora.

Sin duda fue un momento interesante, visto ya ahora en perspectiva, porque se nos presentó un hecho repetido usualmente en un ambiente de trabajo real. Teníamos ante nosotros el siguiente dilema, ya clásico en el campo del desarrollo de sistemas:

> "¿Empiezo a trabajar con el código, o espero a que termine la parte de diseño?"

Claro está que la respuesta más sensata y aparentemente lógica es hacer lo segundo... pero las presiones de tiempo y necesidad, tanto del cliente como del equipo de desarrollo, de poder ver algo que dé la sensación de que "el trabajo está siendo hecho", provoca un estado de tensión que obliga a cambiar los planes y adaptarlos tan vertiginosamente como sea posible.

Para el caso de éste experimento se tomó la primer opción, a sabiendas que el trabajo de diseño tenía deficiencias que habría que sortear, con documentos o sin ellos.

Esta sensación fue así transmitida porque la mitad del equipo de construcción (tres personas) eran la parte más importante del equipo de diseño (seis).

En la fase de transición

Con el tiempo encima para terminar el semestre, se terminó el sistema con niveles bastante aceptables para ser un prototipo y se entregaron al cliente, en éste caso la coordinación del posgrado y el personal administrativo a cargo.

Sin embargo los documentos finales de cada etapa, quedaron dispersos entre el equipo mismo.

La experiencia fue un elemento de gran importancia. Los detalles se dan en la sección: *Los resultados del experimento de desarrollar el sistema*, que está más adelante.

8. Recomendaciones del CMM

La práctica de la administración de requerimientos se cubrió al trabajar los casos de uso de forma que tanto el cliente como el equipo de desarrollo estuvo de común acuerdo.

La planeación del proyecto de software fue satisfecha al tener los roles ya antes mencionados y determinando que cada uno debería trazarse planes de trabajo indicando fechas de entrega de productos de trabajo. Las actividades en ésta área clave eran representadas por el rol de administrador del proyecto.

Para el área clave de la supervisión y seguimiento del proyecto de software, se creó un plan general de desarrollo que establecía los periodos de las fases del proceso unificado. Cada semana se hacían reuniones para ver el avance individual y por equipo. Estas sesiones servían de fuente para elaborar reportes que se entregaban al "cliente" (en éste caso representado por las profesoras). Además se planearon revisiones con una "auditora externa" que verificaba las actividades de las personas en los roles de administradores.

En el caso de la administración de subcontratos de software, no se hizo nada al respecto dado que el proyecto era labor relativamente fácil de contener entre los miembros del equipo.

El área clave del aseguramiento de la calidad del software, fue cubierta por las revisiones que se hacían entre colegas y por las revisiones de formatos. Todos esto estaba controlado por el rol establecido de administrador de calidad. Los mecanismos de revisión fueron la revisión entre colegas y la revisión de líder de proyecto, administrador de calidad y administrador del proyecto.

Para la revisión entre colegas se dejó al libre criterio del revisor hacer las observaciones pertinentes según su juicio. Después de hacer las observaciones del caso, se regresaban los documentos a quienes los generaron y una vez revisados, se repetía el proceso de revisión hasta que el documento fuera aprobado.

En el otro tipo de revisiones, el esquema era así: se entregaban los documentos al líder de proyecto para revisar el contenido, las soluciones propuestas, los diagramas o lo que fuera que presentara el documento; seguidamente se hacía pasar por la revisión del administrador de calidad cuya revisión se enfocaba en el cumplimiento de los formatos y el llenado correcto de la información de identificación en cabecera y pie de página.

Después pasaban a revisión del administrador de proyecto donde se hacia la revisión general de todo entregable. Luego de estas tres revisiones, el documento, si no tenía que ser sujeto a revisión, formaba parte del acervo de productos disponibles para todo el equipo, siendo esto último labor del rol de administración de las configuraciones.

En la práctica el modelo propuesto supuso varias fallas. En primer lugar el proceso se vio entorpecido por la falta de comunicación rápida y efectiva entre las personas a revisar. Cuando un documento era rechazado, es decir que debía ser revisado, se perdía el registro de lo que había pasado con el. La inexperiencia de sujetarse a éste proceso, condujo a tener documentos que fueron revisados y aprobados prácticamente sin revisión formal de los tres encargados porque el tiempo siempre apremiaba, con lo cual hubo periodos de caos. Además este esquema suponía una carga de trabajo extra para algunos miembros del equipo, dificultando que hubiera una repartición más equitativa entre todos para desarrollar el sistema.

El área clave de la administración de la calidad se cubrió con las labores que hizo el rol de administrador de configuraciones.

9. La reunión posterior
Donde se habló de lo que a juicio de cada integrante del equipo funcionó o no, como debía.

Como una de las prácticas recomendadas por el CMM, se realizó una reunión, al terminar el proyecto y anunciarse las calificaciones. Se tenía el objetivo de dar una evaluación objetiva del proceso, dando foro para expresar todas las experiencias, recomendaciones, quejas, conocimientos nuevos, etc.

Cada persona se preocupó por expresar opiniones que en nada comprometieran sus calificaciones o les conllevaran juicios de lamentación. Durante el desarrollo del proceso hubo diversos problemas por diferencias de opinión o por falta de entrega de productos. Fue difícil coordinar a las personas dada la diferencia de actividades extraacadémicas entre los miembros del equipo. Todos aquellos que habían hecho algo en algún momento, se sentían con el derecho de señalar en forma denostativa a los que por causas, justificables o no, no habían hecho lo que se esperaba para el proyecto. Hubo conatos de disputas por diferencias de opinión con referencias a asuntos que en ocasiones no venían al caso.

Ninguno de los aspectos mencionados en el párrafo anterior salió a la luz de la sesión "post mortem" que se realizó para evaluar el proyecto.

10. Los resultados del experimento de desarrollar el sistema.

- Hubo poco apoyo real en el Proceso Unificado.
- Pocos miembros del equipo, leyeron el Proceso Unificado. Los demás se conducían basándose en lo que aquellos decían que debía seguir.
- Casi nula discusión de los porqué del Proceso Unificado. Sólo había interés en saber qué se debía entregar, pero faltó un análisis del porqué.
- Los casos de uso, parte esencial del Proceso Unificado, fueron explorados pobremente, y esto se reflejó en la fase de construcción que se tornó más difícil de lo esperado.

- Se lograron los objetivos del experimento académico, pues cada uno entendió, a su manera, el funcionamiento básico del Proceso Unificado.

- Cada quien hizo lo que suponía era lo correcto, pero faltó mayor y mejor comunicación para tener una visión de común acuerdo.

- La gran cantidad de miembros del equipo favoreció que algunos desatendieran sus labores por varios periodos de tiempo.

- La documentación base para estandarizar la presentación de productos de trabajo, no se entregó a tiempo, es decir cuando se requería.

- Se arrancó la fase de la construcción sin tener los entregables suficientes como para pasar a la siguiente fase, es decir que el análisis necesario de los *hitos* nunca se dio con la formalidad requerida.

- Los resultados finales, en cuanto a productos de trabajo, quedaron en un conjunto caótico de documentos poco legibles para una persona que intentara hacer mantenimiento en el futuro.

- Las profesoras se percataron de la debilidad del diseño y supusieron, con certeza, que el proceso de construcción fue complicado.

- El nivel de compromiso fue muy variable. Hubo personas que jamás se comprometieron con nada. Hubo otros que tomaron interés sólo al final. Hubo también quienes se mantuvieron dispuestos en todo el periodo de desarrollo.

- El rol de administración del proyecto fue controvertido, pues no procuró un ambiente sano para el trabajo, preocupándose más por señalar culpables de todo.

- En todos los casos del desempeño de rol, hubo faltas que pueden atribuirse tanto al desconocimiento inicial de actividades y responsabilidades relacionadas, como a la indiferencia que por momentos se percibía.

- El prototipo finalmente salió a la luz gracias al esfuerzo individual de cuatro personas en un equipo de nueve.

- Aunque quedó como prototipo, el sistema pudo ser utilizado para reinscripción al periodo lectivo 00-II que abarcó de Enero a Junio del 2000.

- Se trabajó con las sugerencias del CMM para buscar un control de calidad del software.

- Se reforzaron los conceptos de programación Orientada a Objetos, cristalizando en el sistema hecho en Java y documentado con diagramas en UML.

- La sentencia con la que inicia la descripción de los objetivos en el nivel 2 de CMM es "establecimiento de controles básicos de administración de proyectos"[25], quedó satisfecha a medias porque hubo entregas tardías de documentos importantes que formaban parte de hitos; por ejemplo la entrega retrasada del modelo de diseño, para que la fase de construcción fuera más sencilla de realizar.

11. Recomendaciones para proyectos similares

Con resultados que pueden considerarse normales dentro de la actividad del desarrollo de software[26], finalmente se obtuvo un producto que aun y cuando era un prototipo, se puso en operación para realizar las inscripciones al semestre 2000-II (Enero a Julio del 2000) con algunas modificaciones menores que realizó el autor[27].

Después de pasar por cuatro meses de curso donde se llevó a cabo el desarrollo del sistema, se pueden mencionar algunas experiencias importantes:

- Es recomendable que, en caso de no tener ningún manual de actividades y responsabilidades de rol, se estudien y elaboren estos documentos que son de particular importancia al momento del desarrollo del proyecto.

- También es útil contar con manuales, resúmenes, presentaciones, artículos, o cualquier otro documento que favorezca el entendimiento común del Proceso Unificado y del CMM, para todos y cada uno de los miembros del equipo. De esta forma también se logrará tener un conjunto de documentos que faciliten la incorporación de nuevos miembros al equipo de trabajo o de reasignar personas al desempeño de un rol.

- Fomentar los mecanismos de comunicación que sean los suficientes como para canalizar ideas, dudas y propuestas que surjan durante el desarrollo del software. Debe ponerse especial cuidado de no tener mecanismos engorrosos o demasiado rígidos que produzca aversión. Algunos ejemplos de esto último: múltiples correos electrónicos a todo el grupo, siendo que el mensaje sólo le era útil a uno o dos; juntas excesivas con cualquier pretexto; llenado de múltiples formas para entregar, pedir o revisar un documento.

---

[25] [Paulk1996] pág 32

[26] Véase por ejemplo [Aguilar1994]

[27] Como por ejemplo cambiar el orden de presentación en las listas y depurar el acceso para alguna pantalla.

- Siempre se favorecerá el análisis en grupo de las actividades de cada rol, procurando hacer examen de autoconciencia para que cada quien elija, si es que es posible, desempeñar el rol que le parezca más apto a sus capacidades, intereses y experiencias. Es útil que la designación de rol a cada persona, se haga en forma consensuada, de nuevo si es que es posible.

- Promover equipos de trabajo con el tamaño justo para satisfacer las necesidades del proyecto. Es decir que no halla integrantes de más, lo cual puede propiciar falta la de interés, ni integrantes de menos, que sobrecarguen el trabajo del equipo. Para esto es indispensable que se cuente con las sugerencias de gentes con experiencia en proyectos similares. Ellos podrán hacer una buena estimación de los recursos necesarios.

- Habrá que tener una sesión donde se planteé desde el principio, que nivel de disponibilidad o compromiso existe en cada persona. Se supondría que para un ambiente académico de trabajo, la disponibilidad e interés es el máximo, pero la experiencia nos señaló lo contrario.

- Buscar tener como administrador de proyecto a aquel que tenga capacidad suficiente de favorecer un ambiente de trabajo sano y que procure la participación de todo el equipo. Que tenga capacidad de negociación y aptitud de colaboración.

- Para un ejercicio posterior en un ambiente académico, se recomienda discutir primero la parte teórica del proceso unificado, así como establecer los manuales de desempeño de cada rol; además considerar el tamaño del producto de software que se desea construir. Luego de eso se puede pensar en hacer los equipos de trabajo.

CAPÍTULO 4

TRANSFORMANDO EL SISTEMA

El ciclo del desarrollo.
El experimento académico de la construcción del software, terminó al mismo tiempo que el semestre escolar 00-I (de Agosto a Diciembre de 1999), y se tuvo instalado y operando una versión prototipo en un servidor de las instalaciones del posgrado. El proyecto se encontraba con las siguiente situaciones:

- La disponibilidad del sistema a los usuarios, tanto administrativos como alumno, produjo el primer periodo de retroalimentación por los reportes de defectos que los usuarios encontraban en el sistema.

- Cuando se entregaron los documentos producto de todo el desarrollo del proyecto, hubo algunos que eran incongruentes con lo que realmente se tenía en el sistema final.

- No se hizo un seguimiento de los defectos reportados.

Por estos motivos se consideró que el ciclo completo del desarrollo del sistema, aún no estaba cerrado, pues

- El producto transitó hacia su versión beta, donde un grupo de usuarios experimentados trabajó con el producto.

- Los usuarios reportan defectos y deficiencias encontradas.

- Los desarrolladores corrigen los problemas reportados e incorporan mejoras sugeridas para la versión general.

- Se requerían actividades de ayuda al usuario.

- Era necesario capacitar al personal del cliente

Situaciones todas ellas que indicaban una fase de transición.

La lista de sugerencias o errores encontrados es la siguiente:

Lista de deficiencias del sistema

1. No hay un mecanismo de seguridad para evitar el acceso al sistema de gente no autorizada.

2. No existe la posibilidad de usar cíclicamente el sistema al paso de los semestres, pues debería ser "descargada" o eliminada la información de la base de datos para hacer de nuevo el proceso de inscripción.

3. No hay control en el cupo para las materias.

4. No cuenta con todas las validaciones necesarias para la entrada de datos.

5. Las pantallas son demasiado lentas para cargarse.

6. La presentación visual al usuario es pobre.

7. Si hay errores durante la operación de alguna pantalla, no se informa claramente al usuario.

8. Hay ocasiones en que el sistema queda bloqueado, inutilizando toda acción posible en el visualizador.

9. Puede suceder el caso de que una materia pertenezca a dos o más áreas de concentración de estudios.

10. No hay interfase con el sistema de la universidad que asienta en actas el hecho de estar inscrito en un semestre.

11. No hay verificación de la seriación de las materias.

12. No hay ayuda disponible en caso de hallarse sin saber que hacer.

En la Fase de Transición.

Un primer periodo de corrección de defectos, se realizo durante Enero del 2000, donde el autor hizo las siguiente correcciones que no afectaban prácticamente en nada la arquitectura general del sistema.

- Establecer contraseña igual al número de boleta de los alumnos.

- Reparación de un error al insertar las materias solicitadas para inscripción.

- Generación del grupo en la lista de materias ofrecidas por semestre.

- Presentación en orden alfabético de la mayoría de las listas de los elementos (materias, grupos, alumnos, profesores), cada materia podía pertenecer a dos o más áreas de conocimiento.

Todas las observaciones, salvo la última, pudieron ser corregidas en cuestión de minutos. Para la última se debía hacer un cambio desde el esquema de la base de datos así como modificar código sin tener idea a ciencia cierta de cuánto.

Después de éste conjunto inicial de peticiones que fueron resueltas, se tomó la lista de "deficiencias pendientes" y la iteración se asentó en los flujos de trabajo.

Los flujos de trabajo en la fase de transición
En el **flujo de requerimientos** no hice nada más que conseguir los documentos de los modelos y las especificaciones de análisis y diseño. Esto implicó más trabajo del supuesto porque al termino del proyecto experimental, el equipo de trabajo se dispersó y no hubo un control de administración de configuraciones que facilitara la labor de ordenar todos los productos de trabajo.

En el **flujo de análisis**, hice la identificación de las deficiencias y nuevas solicitudes del cliente. Una de las más importantes fue la corrección para ser un sistema que soportara la operación continua entre semestre. Inicialmente se hizo un esquema de la base de datos donde solo se guardaba información en un semestre; en otras palabras no podía almacenarse la historia académica pues al volver a usar el sistema se debían sobre escribir las informaciones, perdiendo todo pasado.

En el **flujo de diseño**, se hicieron esencialmente las siguiente tareas: rediseño e implementación de la base de datos, rediseño total de las interfaces gráficas al usuario, agregado de control de cupo para cada curso ofrecido, agregado de parámetros generales para dar mantenimiento por semestre.

En el **flujo de implementación**, se construyeron todas y cada una de las pantallas que conformaban el sistema.

Para el **flujo de pruebas**, se hicieron las pruebas del nuevo código, las pruebas confrontando casos de uso y las pruebas con el usuario, durante la inscripción al semestre 01-I, que corrió de agosto a diciembre del 2000.

Enmendando las deficiencias.
Se reportará cómo fueron cubiertas las deficiencias listadas previamente:

1. *"No hay un mecanismo de seguridad para evitar el acceso al sistema de gente no autorizada"*

   Para los usuarios alumnos, se optó por asignar aleatoriamente un número clave en el momento que era validada su contraseña y se habilitaba la entrada a la siguiente pantalla que iniciaba el proceso de elección de materias para la inscripción.

Para ello se implementó la siguiente estrategia: el applet[28] que valida nombre y contraseña del alumno, genera un número aleatorio que es almacenado en la base de datos. Seguidamente, si la contraseña es válida, se llama a una página dinámica con los argumentos: número interno de identificación del alumno y el número aleatorio generado. Una vez que la página empieza a desplegarse, la primera actividad que el applet hace, es verificar que el número aleatorio que tiene el visualizador como argumento, sea el mismo que tiene la base de datos, siendo éste el caso, se continúa la carga del applet, sino se despliega un mensaje que avisa la falta de autorización para ver la página.

Para los usuarios administrativos, el manejo de seguridad se hizo apoyándose en el servidor de http. En el apéndice E se detalla las modificaciones útiles para proveer de seguridad con requerimiento de identificación para el acceso a un directorio de archivos en un servidor http del popular sistema Apache[29].

2. *"No existe la posibilidad de usar cídicamente el sistema al paso de los semestres, pues debería ser* descargada *o eliminada la información de la base de datos para que pueda hacerse de nuevo el proceso de inscripción."*

Se planteó otro esquema de base de datos que permitiera guardar la información relativa al semestre, ya que cuando se creó la primer versión, no se tomó en cuenta.

3. *"No hay control en el cupo para las materias".*

A fin de establecer un cupo máximo de alumnos en una materia, se creó un parámetro general llamado *cupo_máximo* que acota la cantidad de registros que es posible almacenar en la base de datos. No hay ningún mecanismo que haga apartado de materias mientras el alumno decide que curso tomar, así que el lugar para inscripción se ocupa hasta el momento en que el alumno pide que su solicitud sea aceptada en el sistema. Luego entonces si dos o más alumnos compiten por un último lugar en una materia, ese será asignado al que primero solicite inscripción.

4. *"El prototipo no cuenta con todas las validaciones necesarias para la entrada de datos".*

Al prototipo se le implementaron algunas validaciones simples pero no hubo rigor ni fue norma, y cada programador se encargó de hacer las validaciones que creyó necesarias. Para la nueva versión se estandarizaron las validaciones de entradas al usar manejo de excepciones en Java.

---

[28] Applet es un programa que se ejecuta en un visualizador como Communicator de *Netscape* o Hot Java de *Sun*.

[29] Véase [WebAPA].

5. *"Las pantallas son demasiado lentas para cargarse".*

Las pantallas eran muy lentas porque primero se cargaba en el visualizador el archivo con las clases que permiten la conexión con la base de datos, el cual medía aproximadamente 360 Kb. Ciertamente en conexiones rápidas la carga era veloz, pero en el caso de conexiones típicas, la carga podía ser en ocasiones muy pesada. Para este problema se hicieron archivos jar[30] que contenían comprimidas las clases que posibilitan la conexión a la base de datos y los archivos de las aplicaciones en sí. El tamaño se redujo la mayoría de las veces en un 60 por ciento del mencionado.

6. *"La presentación visual al usuario es pobre".*

Se prefirió rediseñar la presentación de las pantallas, ocupando ahora el paquete *swing* de Java que tiene posibilidades de presentarse en diferentes diseños además de ser más funcional, más rica y más atractiva a la vista del usuario. Se pueden apreciar las diferencias entre las pantallas en el capítulo siguiente. La única desventaja era que debía de obtenerse del Internet un *plug-in* que habilitara la visualización de los elementos del paquete *swing*, aunque esto se hacía una sola vez, implicaba un tiempo de espera en lo que dicho elemento era copiado desde un sitio remoto en ocasiones muy lento.

7. *"Si hay errores durante la operación de alguna pantalla, no se informa claramente al usuario".*

Cuando se presentaba un error, el sistema prototipo reaccionaba de la siguiente forma: o no informaba de la ocurrencia del problema o bien mostraba un mensaje críptico en la barra de estatus del visualizador. Para ésta nueva versión se añadieron múltiples mensajes que informan o guían al usuario al operar el sistema.

8. *"Hay ocasiones en que el sistema queda bloqueado, inutilizando toda acción posible en el visualizador".*

Ocasionalmente sucedía con el usuario administrativo, que el sistema simplemente se bloqueaba y dejaba de reaccionar ante cualquier estimulo posible. Esto se debía por la gran cantidad de espacios de memoria que al navegador reservaba después de una operación de lectura de datos. Estos espacios de memoria con resultados de una petición a la base de datos, estaban "comiéndose" literalmente el recurso de memoria. Para esta nueva versión se llevó un control estricto de apertura-cerrado de dichos espacios. Al menos hasta la segunda etapa de pruebas nunca se volvió a presentar éste problema.

9. *"Puede suceder el caso de que una materia pertenezca a dos o más áreas de concentración de estudios".*

---

[30] Un archivo que contiene a muchos y que está comprimido.

Nuevamente se modificó la base de datos para permitir que una materia pudiera pertenecer a dos o más áreas de estudio. Originalmente en la recolección de requisitos se habló de una relación uno a uno. A cada materia le correspondía sólo un área

Puntos que se dejaron como proyectos a futuro

10. *No hay interfase con el sistema de la Universidad que asienta en actas el hecho de estar inscrito en un semestre*

11. *No hay verificación de la seriación de las materias*

12. *No hay ayuda disponible en caso de hayarse sin saber qué hacer.*

Tanto la interfase con el sistema institucional, como la revisión de seriación de materias, y la implementación de una ayuda contextual, se vislumbraron como parte de los proyectos a futuro.

CAPÍTULO 5

COMPARANDO LAS VERSIONES

Revisión de los cambios por casos de uso
Se reportarán las apreciaciones de los cambios según los casos de uso[31], tomando las comparaciones con la versión anterior del sistema.

1.- Petición de ingreso al sistema del usuario alumno

Problema.

Anteriormente el ingreso al sistema, tanto para usuarios administrativos como alumnos, era a través de una misma pantalla.

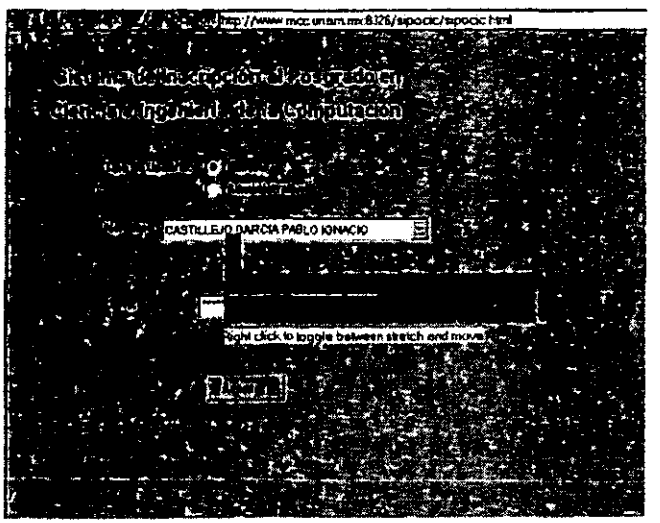


Fig. 23 Pantalla de ingreso en versión prototipo

[31] Vease el apéndice B para los casos de uso.

Esto representaba una debilidad en seguridad pues los alumnos conocían, de inmediato, el nombre[32] de los usuarios administrativos, y con un poco de esfuerzo podrían dar con la contraseña de acceso, asumiendo las capacidades de un usuario administrativo.

El esquema de identificación del usuario para acceder al sistema en la versión anterior, era a grandes rasgos como sigue:

Una primer pantalla verificaba nombre del usuario y contraseña, si era correcta se almacenaba el nombre del usuario en una tabla de la base de datos. Enseguida el sistema mostraba una segunda pantalla cuya primer actividad era tomar el nombre del usuario de la tabla, eliminarlo de ella, y mostrar los datos del usuario identificado.

En caso de que la primer pantalla encontrara que la tabla no estaba vacía, asumía que estaba siendo leída por la otra pantalla, y por tanto no se podría identificar al usuario, con lo cual denegaba el acceso, mostrando un mensaje de "Red saturada. Intente más tarde". Y la segunda pantalla debía de eliminar al usuario que acababa de identificar.

En otras palabras, la primer pantalla metía los nombres de los usuarios en una tabla que hacía de cola de espera[33] de capacidad uno, la segunda pantalla toma de ahí el nombre del usuario correspondiente. Esto implicaba que solo podía acceder un usuario a la vez al sistema[34]. Además si había algún error, y la cola de espera se quedaba con un elemento, implicaba que el sistema quedara permanentemente ocupado, pues la primer pantalla leería un dato y denegaría el acceso.

Solución propuesta.

El acceso al sistema se separó en dos. En una dirección ingresan los alumnos[35], y en otra los usuarios administrativos[36].

---

[32] *Login* en este caso.

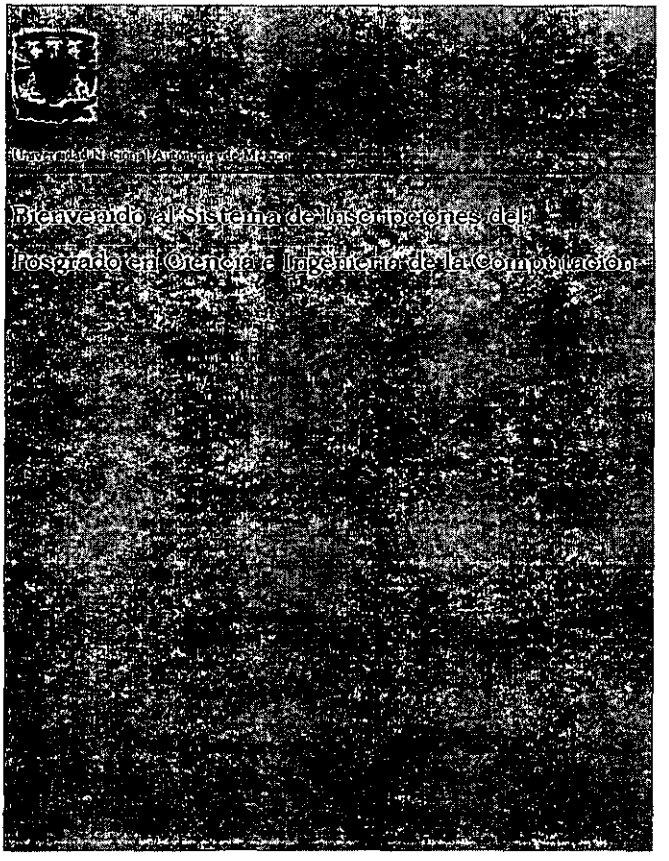
[33] Estructura de datos que almacena elementos en forma lineal uno tras otro, agregando los nuevos en un extremo, similar a una *cola* o formación de personas, que están esperando ser atendidas para un tramite o servicio.

[34] Sistema no concurrente.

[35] *http://[illegible]:8326/inopción.html*

Para los alumnos se implementó el siguiente esquema de identificación de usuario: Una primer pantalla pide los datos esenciales de identificación, nombre de usuario y contraseña.

Fig 24. Primera parte de la pantalla de bienvenida al sistema.



[36] http://www.mcc.unam.mx/inscripciones/adsip.html El nombre de la página html viene de las iniciales de administración del sistema de inscripciones al posgrado.

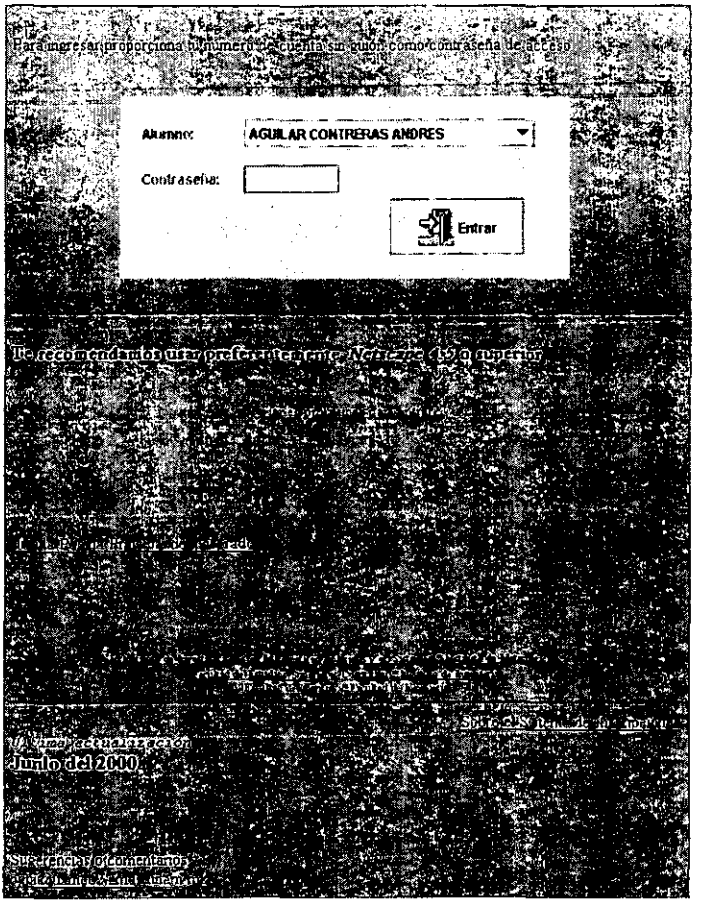


Fig. 25 Segunda parte de la pantalla de bienvenida al sistema

Estos datos se verifican contra lo almacenado en la base de datos del sistema. Si es correcta la identificación, se genera un número aleatorio que se toma como el número de la sesión. Éste dato se almacena en la base de datos. Enseguida se ejecuta un servlet[37] que forma dinámicamente una página html que recibe como parámetros el número de sesión del alumno y el número que identifica al alumno. La página manda a ejecutar un applet que primero verifica la coincidencia de los datos de los parámetros. Así se valida que la sesión es de un usuario legal y se procede a abrir el sistema a las operaciones de solicitud de inscripción.

---

[37] Programa que se ejecuta del lado del servidor.

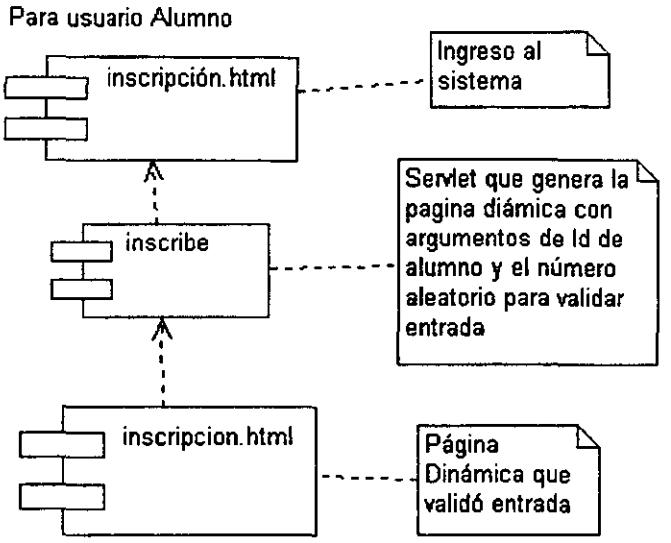


Fig 26. Diagrama del nuevo esquema para ingresar al sistema

Suponiendo el caso de un usuario que desee entrar al sistema sin identificarse, debería tener tres condiciones; uno, conocer la dirección web desde donde se ejecuta el servlet; dos, conocer el número de identificación personal del alumno; y tres, el número aleatorio que se asignó a la sesión, dato éste último que sólo se conoce en tiempo de ejecución.

Si estas condiciones no se cumplen, el sistema no permite el acceso.

Justificación.

Era relativamente sencillo que un usuario alumno ingresara como usuario administrativo, de ahí que se prefiriera separar el acceso según el tipo de usuario.

El mecanismo de seguridad existente era mínimo y deficiente, pues con conocer las direcciones web de cada página se ingresaba como usuario administrativo.

No podía haber accesos concurrentes al sistema.

2.- Validación del usuario

Problema

Se identificaba al usuario a con su nombre y contraseña que se tenía en la base de datos, pero el acceso al sistema no podía ser concurrente, en caso de los usuarios alumnos.

Solución propuesta

Se implantó un mecanismo de seguridad para los alumnos, que permite accesos concurrentes y es más robusto en ejecución. Los detalles se presentan en el caso de uso 1: "Petición de ingreso al sistema del usuario alumno.

Para el usuario administrativo se implantó otro mecanismo de seguridad que se detalla en el caso de uso 5: "Petición de ingreso al sistema del usuario administrativo".

Justificación

Para los usuarios alumnos, se cuenta con un sistema de seguridad basado en un número aleatorio generado en tiempo de ejecución, con lo cual es menos sencillo entrar ilegalmente al sistema.

Para los usuarios administrativos se optó por establecer un mecanismo de identificación a través del propio servidor de http, con lo cual se ve cubierta la necesidad de seguridad.

Cabe preguntarse porque no se hizo esto último para los alumnos. Se decidió éste esquema porque el universo de los alumnos es más grande y movible que el de los usuarios administrativos. Además se elimina la precondición para entrar al sistema de alumnos con cuenta en el servidor http del posgrado.

Así también se mantiene un universo pequeño de usuarios administrativos, cuyo ingreso podría depender en algún momento, del apoyo del administrador del servidor, y dejando el ingreso y mantenimiento de los datos de los alumnos en manos del personal administrativo.

3.- Generación de Solicitud de Inscripción

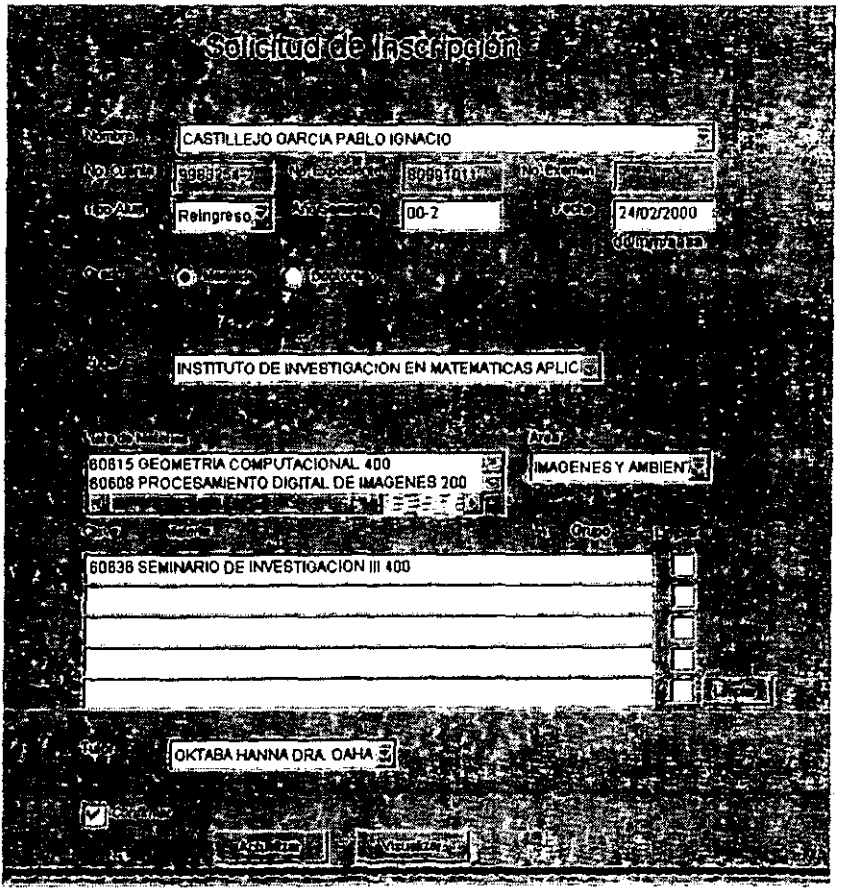
Problema

La pantalla tenía en ocasiones deficiencia al mostrar los datos de cada materia según el área de estudios que tuviera asignada. Hacían falta validaciones de la fecha en que se hacía la inscripción. El archivo

resultante del código Java intermedio, alcanzaba un tamaño de 300 Kb, con lo cual el acceso a la pantalla correspondiente, era lento si no había una buena conexión de Internet.

Había ocasiones en que el sistema se bloqueaba dejando nulas posibilidades de reestablecer la comunicación con el sistema. Esto se debía a que en ocasiones se dejaban demasiados espacios de memoria con resultados de peticiones de datos[38] que eventualmente no se volvían a usar, pero al no eliminarlos de la memoria, provocaban un desbordamiento de datos del cual no se recuperaba el sistema.

Fig 27. Pantalla de inscripción en la versión prototipo



[38] Llamados *asesors* en el estudio de las bases de datos.

Solución propuesta

Todos los cursores que se usaban para recuperar datos, en todo el sistema, fueron eliminados justo después de ser usados, con lo cual la memoria se mantiene todo el tiempo con el mínimo necesario para operar. De ésta manera no se volvió a bloquear el sistema en tiempo de ejecución.

Se usó la característica de Java de la generación de archivos jar, con lo cual se redujo el tamaño del archivo de código intermedio, produciendo menos tiempo de espera para acceder a la página. Se rediseño la presentación haciéndola más amigable.

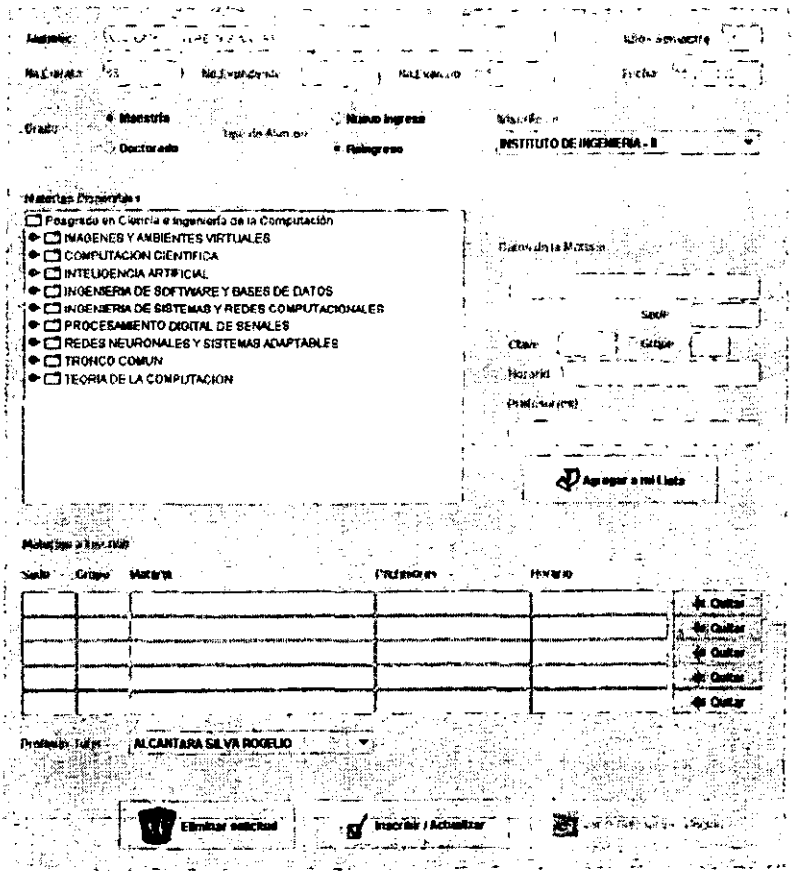


Fig. 28 Pantalla de inscripción de la nueva versión del sistema

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
COORDINACION GENERAL DE ESTUDIOS DE POSGRADO
SOLICITUD DE INSCRIPCION

No. de Cuenta 65 No. de Expediente

Anótalo únicamente si ha estado inscrito en la UNAM — Anótalo si ha estado inscrito al posgrado de la UNAM

Nombre AGUILAR CONTRERAS ANDRES

primer apellido — segundo apellido — nombre(s)

Ingreso [ ] Reingreso [X] Año y Semestre [01-I] Fecha [26/07/2000]

Solicita inscripción a: — Claves

Facultad o escuela: INSTITUTO DE INGENIERIA [0300]

Especialización en: [ ]

Maestría en: Maestría en Ciencia e Ingeniería de la Computación [4014]

Doctorado en: [ ]

En las siguientes asignaturas:

Anote el nombre completo de la(s) asignatura(s), clave y grupo, así como el nombre del o de los profesor(es) . Clave asignatura y Grupo.

GRAFICACION POR COMPUTADORA SOLIS GONZALEZ COSIO ANA LUISA MAT.	60609	400
MODELACION MATEMATICA Y COMPUTACIONAL I FRAGUELA A. ANDRES	60595	400
LOGICA MATEMATICA ROMAN C LEOPOLDO	60543	500
BASES DE DATOS II GARCIA GARCIA JAVIER M. EN C.	60553	400
TEMAS SELECTOS DE INGENIERA DE SOFTWARE MARQUEZ FLORES GUSTAVO M. EN C.	60563	400

Firma del Tutor — Firma del Coordinador del Programa — Firma del Alumno(s)

OAHA511015

Fig 29 Pantalla generada dinámicamente de la solicitud de inscripción

Justificación

Se validan todos los datos posibles. Se tiene un menor tiempo de acceso dado que los archivos usados son más pequeños (del orden de 50 % menos que en la versión primera).

Era necesario identificar el problema de bloqueo del sistema porque ya no se lograba recuperar después.

4.- Modificación de la solicitud de inscripción

Problema

Podría parecer un tanto confusa la pantalla de presentación de la solicitud de inscripción. No había mensajes claros de lo que estaba pasando en el sistema o por los posibles errores en el ingreso de los datos.

Solución propuesta

Se presentó otra interfaz gráfica para mejorar su aspecto y facilitar el uso y comprensión del usuario.

Justificación

La utilización de elementos gráficos, proporcionan un mensaje más en el entendimiento del uso de un sistema de cómputo cualquiera. Se introducen en ésta pantalla algunos elementos que mejoran su comprensión y facilidad de uso.

5.- Petición de ingreso al sistema del usuario administrativo

Problema.

Todas las páginas html y archivos punto class de Java, quedaron ubicadas en un mismo directorio del servidor http, sin ningún mecanismo que restringiera el acceso. Esto dejaba el sistema listo para usar cualquier pantalla, con sólo conocer el nombre de la página html, lo cual es relativamente sencillo de averiguar. El mecanismo de seguridad era mínimo, por lo que se comentó en el primer punto de ésta sección.

Solución propuesta

Se separó el acceso al sistema, como se mencionó en el punto uno de ésta sección. El mecanismo de seguridad quedó en manos del servidor http, con lo cual se ganó robustez y confianza.

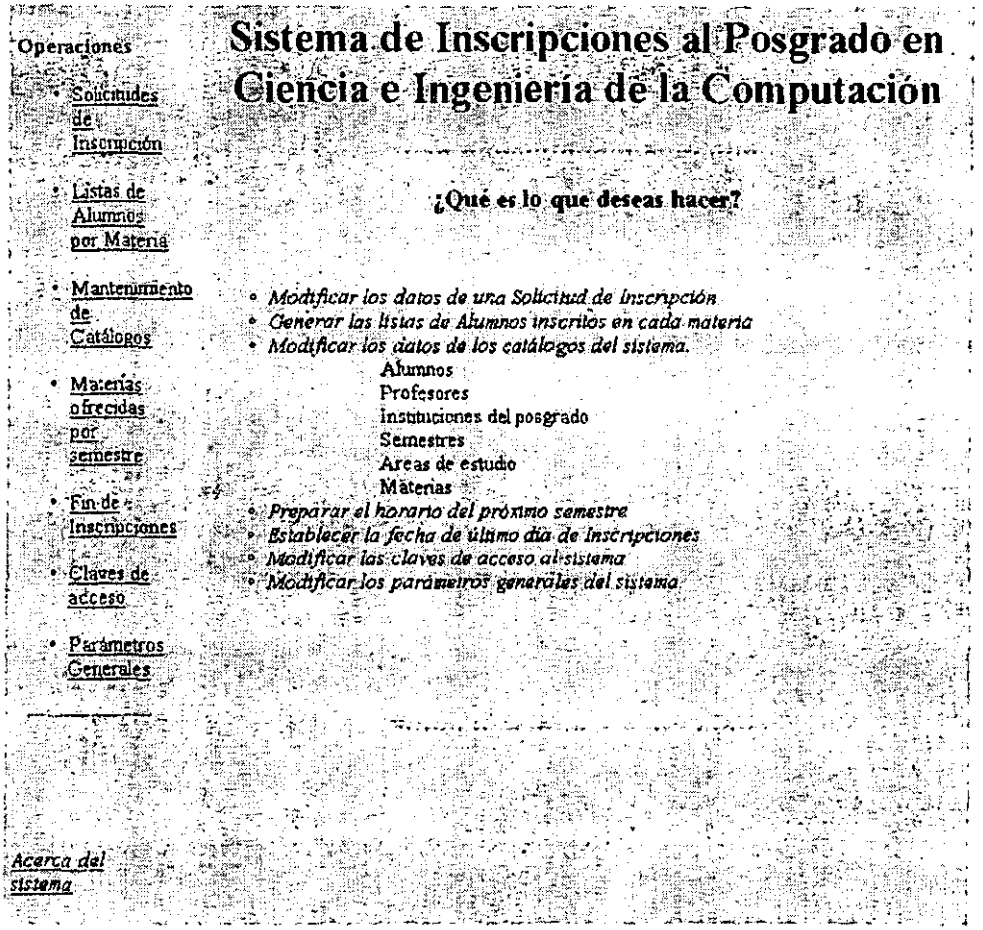


Fig. 30 Menu de operaciones para los usuarios administrativos en la nueva versión del sistema

Justificación

Se mejoró el aspecto general del módulo del usuario administrativo.

Se implementó un mecanismo de seguridad robusto.

6.- Modificación de Solicitud de Inscripción por parte del usuario administrativo

Tanto la descripción del problema como la solución y la justificación de ésta sección, se apegan a lo que se ha comentado en el punto cuatro de ésta misma sección.

7.- Preparación de datos del semestre

Problema

Al no existir la visión a largo plazo de usar el sistema una y otra vez al paso del tiempo, éste caso de uso no fue debidamente establecido y estudiado.

Solución propuesta

Las actividades que el usuario administrativo debe llevar a cabo son las siguientes:

- Cerrar el acceso de los alumnos al sistema, lo cual se hace con la pantalla mostrada en Apertura de Inscripción, punto once de ésta misma sección.

- Indicar cual será el nuevo semestre vigente, lo cual se hace con la pantalla mostrada en el punto once.

- Actualizar los catálogos de profesores, alumnos y materias del sistema, que se realiza con la pantalla mostrada en el punto ocho de ésta sección.

- Construir los horarios vigentes para el semestre en cuestión. Operación que se efectúa con la pantalla del punto doce de ésta sección. Si se desea generar una página html con la información resultante, se puede obtener desde la pantalla mencionada.

- Establecer la fecha del fin de las inscripciones. Que se hace con la pantalla mostrada en el punto catorce de ésta misma sección.

- Abrir las inscripciones en la fecha que establezca la institución, lo cual se hace modificando nuevamente el parámetro general de "***inscripción_abierta***", que se muestra en el punto once de ésta sección.

Justificación

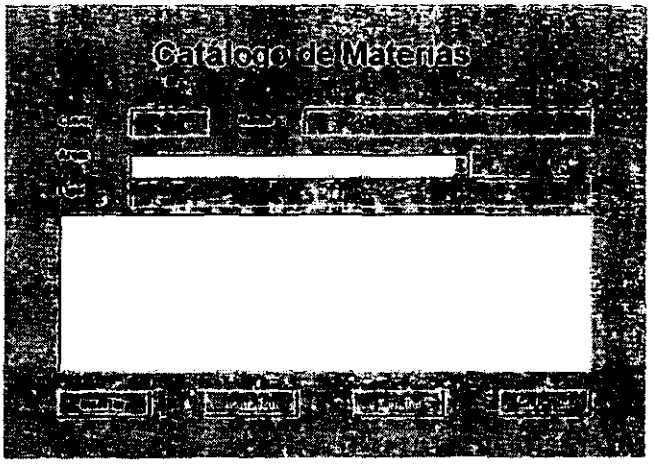
Era necesario un sistema que se pudieran usar cada que iniciara un nuevo periodo de inscripciones. El uso de cada una de las pantallas que conforman el módulo del usuario administrativo, permiten llevar con orden el proceso de generar toda la información para los alumnos en un nuevo proceso de inscripción.

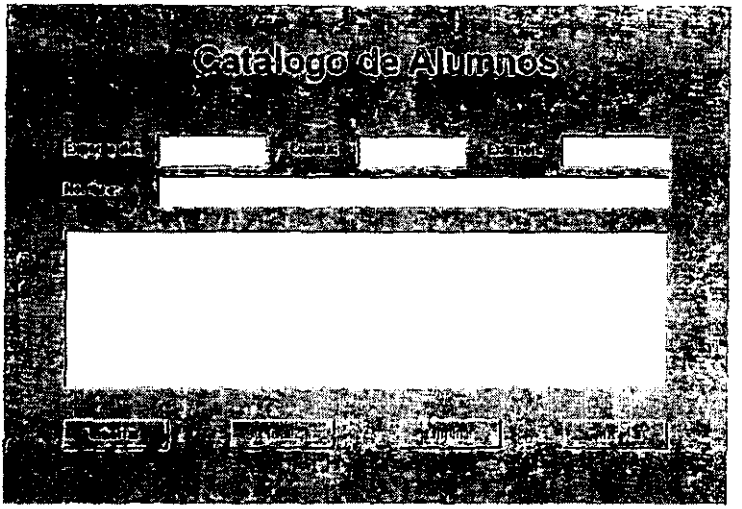
8.- Actualización de catálogos del sistema.

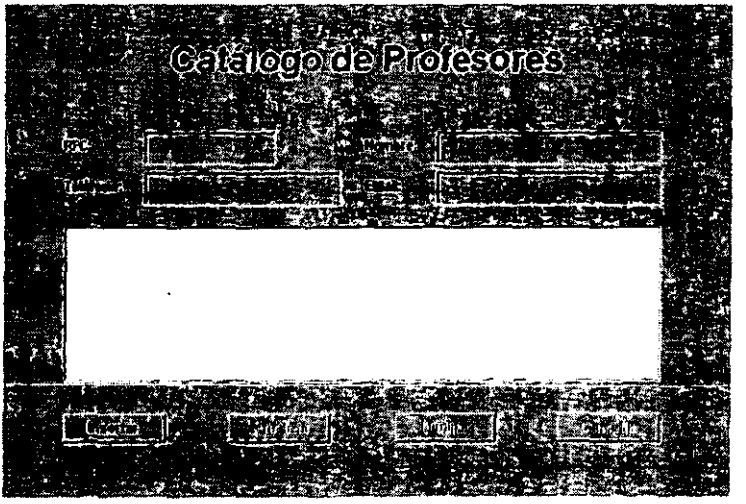
Problema

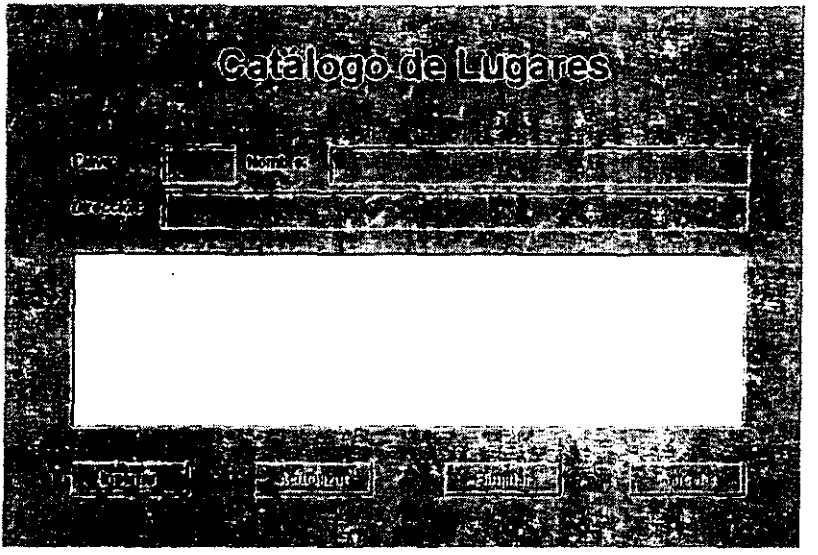
Se tenían cinco pantallas separadas, cada una conllevaba un "tiempo de carga" que ocupaba el visualizador para presentar la interfaz. Las operaciones que se llevaban a cabo eran las mismas: inserción, eliminación y modificación. Veamos las cuatro pantallas diferentes con que se manejaban los catálogos.

Fig. 31 Pantallas para mantenimiento de catálogos en la versión prototipo




Catalogo de Alumnos


Catalogo de Profesores



Solución propuesta

Se rediseñó la interfaz de usuario y se muestran ahora todos los catálogos en una sola pantalla, dando así más cuerpo a la presentación y unificando las operaciones de los catálogos en un solo lugar.

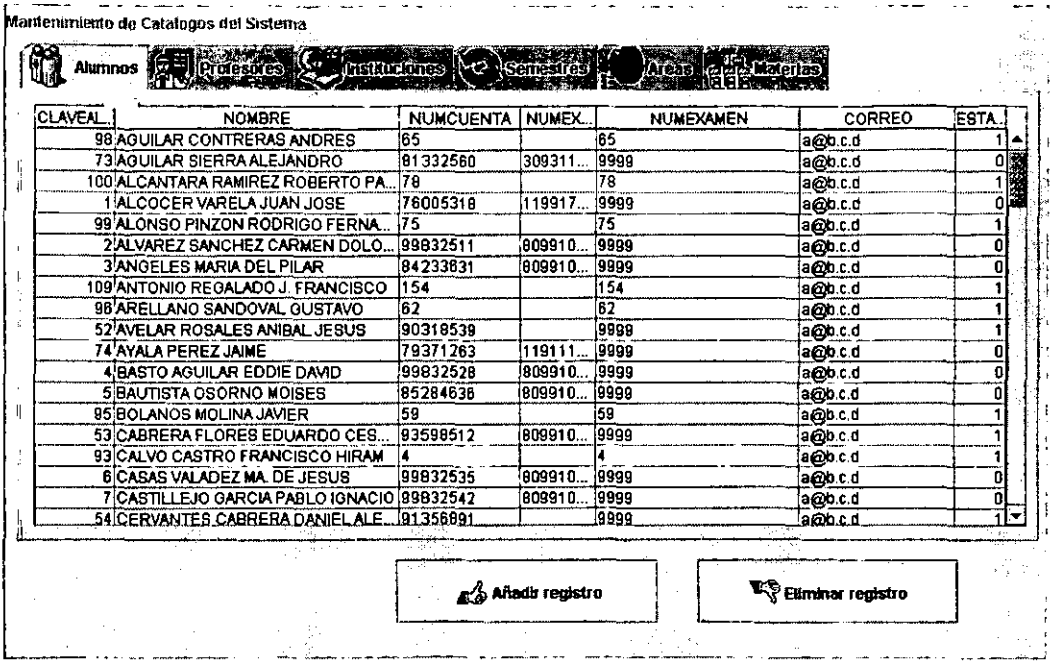


Fig. 32 Pantalla de la administración de catálogos en la nueva versión

Justificación

Las operaciones a realizar en los catálogos es siempre la misma, por lo cual era lógico utilizar la característica de polimorfismo en las operaciones de los métodos que efectuaran estos procesos: alta, baja y modificación de registros. Al cambiar la presentación se mejoró el aspecto presentando los datos en forma de tablas en lugar de listas. Al momento de la construcción de éste sistema, el paquete *Swing*, que fue el que proporcionó los elementos de las applets, estaba en fase de mejoramiento, principalmente en el desempeño de tiempo de reacción. Por esto el usuario reportó lentitud en el proceso y confusión por el manejo abstracto de los botones que controlaban los registros.

9.- Establecimiento del cupo máximo

Problema

No había forma de poner un límite de alumnos inscritos en un curso de los ofrecidos en el posgrado.

Solución propuesta

Se implementó un módulo de parámetros generales, donde se establecerán los valores generales para el sistema, en aspectos que impactan en la mayoría de las operaciones del sistema. Uno de tales parámetros generales es ***cupo_máximo***, que indica el número límite máximo de alumnos que se permite que estén inscritos en un curso.

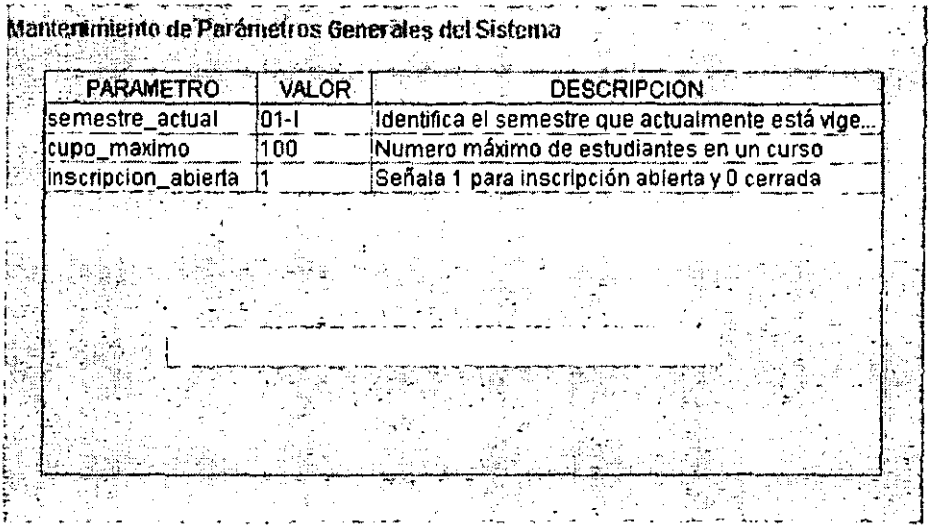


Fig. 33 Pantalla de parámetros generales en la nueva versión del sistema

Justificación

No había forma de limitar el número de alumnos inscritos a un curso.

10.- Selección del semestre actual.

Problema

Como ya se comentó en el punto de "Preparación de datos del semestre ", al inicio del proyecto no se consideró hacer inscripciones cíclicas, por lo cual una operación del tipo establecer un semestre actual no estaba prevista. Se sobreentendía que sólo un semestre podía manejar el sistema. Así que si se deseaba hacer otro periodo de inscripciones, se debía hacer un respaldo de la base de datos y después eliminar todos los datos, como proceso de preparación de ambiente del sistema.

Todo esto implicaba un conocimiento medianamente detallado del sistema pues al ser operaciones no implementadas, debían hacer "por fuera" sin usar el sistema como tal, sino operando únicamente la base de datos.

Solución propuesta

Uno de los parámetros generales del sistema que se implementaron, fue el llamado **"semestre_actual"**, que indica cual de los semestres que tiene almacenado el sistema, es el que se tomará como base para determinar los datos de todo el sistema

Justificación

Como ya se explicó, el sistema no guardaba información al respecto del semestre, así que este caso de uso es nuevo en ésta versión. Se cuenta ahora con la posibilidad de hacer tantas reinscripciones como capacidad tenga la base de datos.

11.- Apertura de Inscripción

Problema

No existía forma de que el usuario administrativos prohibiera el acceso de los alumnos al sistema. Esta operación podría ser útil en caso de que se intentara hacer una revisión de los datos ingresados al mismo.

Solución propuesta

Se agregó el parámetro ***"inscripción_abierta"***, que funciona a manera de semáforo para el acceso de los usuarios alumnos al sistema.

Justificación

El usuario administrativo tiene manera de prohibir el acceso al sistema para los usuarios alumnos.

12.- Generación de horarios semestrales.

Problema

Cuando se concibió la versión prototipo del sistema, se tuvo la visión general de construir un sistema que sería usado en el próximo periodo de inscripciones... pero no se pensó mas allá. Es decir que no hubo preocupación por almacenar la información de forma sistemática para funcionar en forma cíclica conforme avanzara el tiempo y llegara otro periodo de inscripciones.

En el caso específico de la pantalla con la cual se construía el horario para el semestre, era en ocasiones confusa y no había regularidad en la disposición de los elementos que la conformaban, creando una sensación de caos.

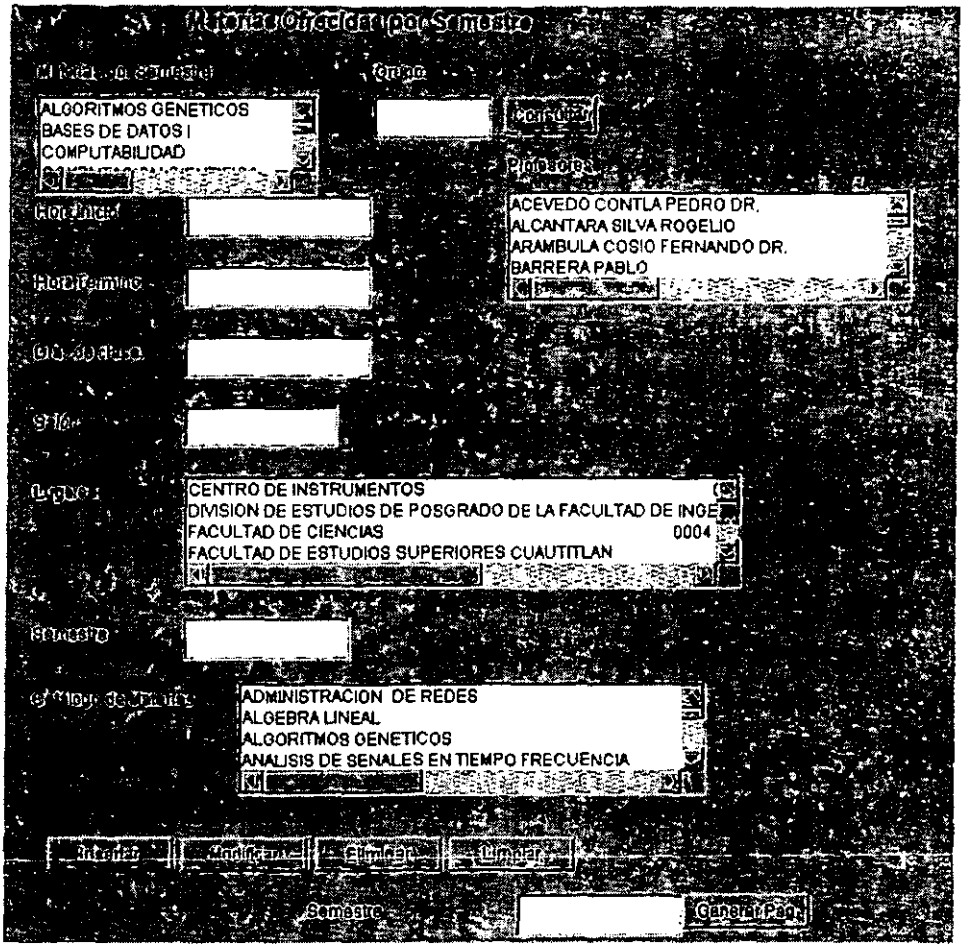


Fig 34. Pantalla de creación de horarios semestrales en la versión prototipo

Solución propuesta

Para posibilitar el uso del sistema a través del paso de los semestres, se modificó el esquema de la base de datos para guardar el historial de las inscripciones hechas. Con este cambio en mente se modificaron prácticamente todas las sentencias de recuperación, inserción y modificación de datos desde y hacia la base de datos.

Se rediseño la presentación de la interfaz de usuario y se utilizó ampliamente el modelo de reacción por eventos de la programación orientada a objetos, con lo cual se simplificó la pantalla que construía los horarios de los semestres.

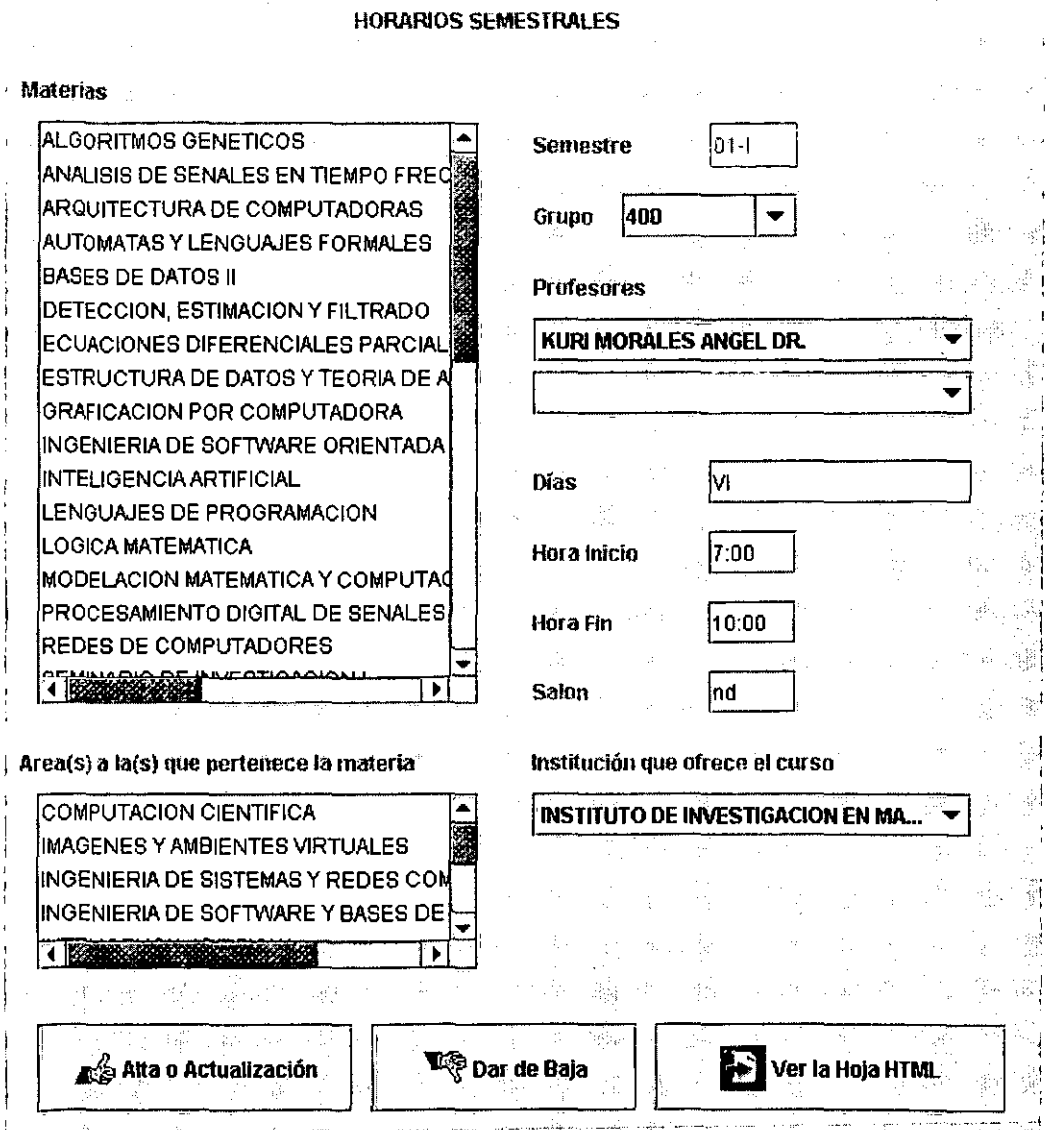


Fig. 35 Pantalla de creación de horarios semestrales en la nueva versión

Justificación

Se simplificó una de las pantallas más útiles para el usuario administrativo, que es la de construcción de los horarios semestrales. Para esto se estableció el flujo de los eventos que debía hacer el usuario administrativo para preparar el sistema. Anteriormente no se tenía porque el prototipo se creó pensando en el semestre en que se hicieron las pruebas pero sin dejar posibilidad de hacer inscripciones cíclicas.

13.- Mantenimiento de datos del horario

Este caso de uso tiene como problema, solución propuesta y justificación, la misma que el punto anterior.

14.- Establecimiento de fecha de fin de inscripciones

Problema

En la versión anterior, se hacía el procedimiento de "Cerrar inscripciones", con una pantalla para tal fin. El efecto era inhibir el acceso al sistema. La acción quedaba a responsabilidad del usuario administrativo, y no había "marcha atrás" para este procedimiento. Si por alguna razón el usuario se había equivocado se debía solicitar la ayuda de algún desarrollador para reestablecer el acceso, con lo que se implicaba una operación que debía hacer el sistema.

Fig 36. Pantalla de "cierre" de inscripciones en la versión prototipo del sistema

Solución propuesta

Se implementó un programa que permite establecer y modificar la fecha de fin de inscripciones, dejando al sistema la determinación del cierre de inscripciones. Así puede modificarse la fecha a juicio del usuario administrativo.

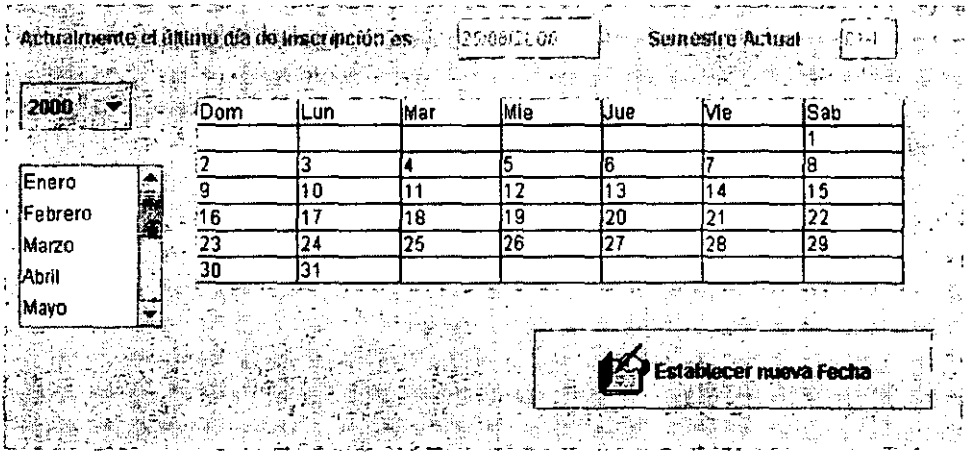


Fig 37 Pantalla para establecer la fecha de último día de inscripción

Justificación

Mejor presentación al usuario. Se transfiere la responsabilidad del cerrar las inscripciones del usuario administrativo al sistema. Se puede cambiar la fecha de cierre de inscripciones.

15.- Mantenimiento de datos de acceso

Problema

Los datos eran mostrados como una lista y la presentación al usuario era pobre.

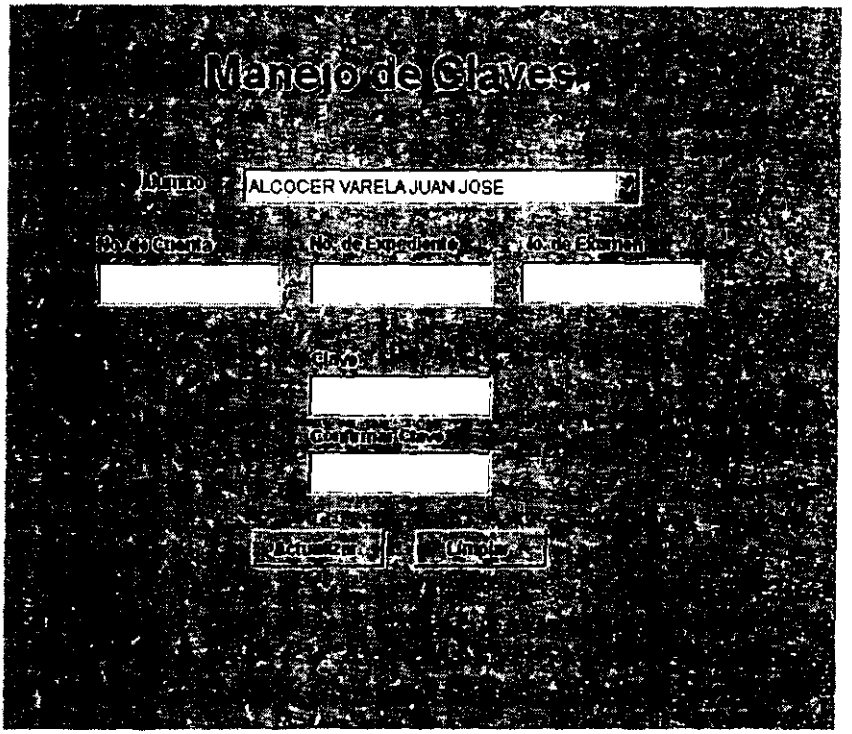


Fig. 38 Pantalla mantenimiento de claves de acceso en la versión prototipo

Solución propuesta

Se rediseñó la interfaz gráfica y se mostraron los datos en forma más simple y natural con una tabla.

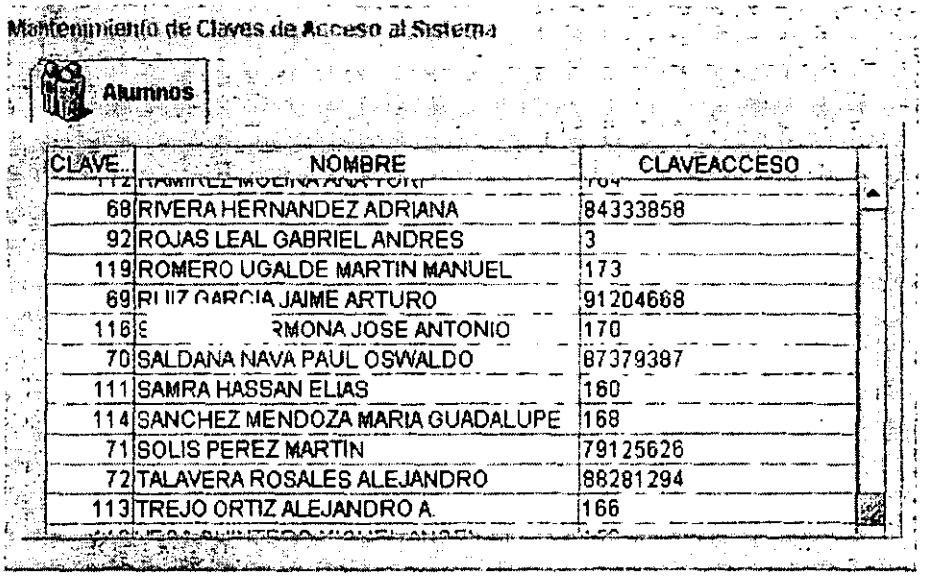


Fig 39 Pantalla del mantenimiento de claves de acceso para los alumnos en la nueva versión

Justificación

Los datos tienen una forma más sencilla de presentarse y se usa el polimorfismo en la operación de modificación de los datos en los registros, comunes a todas las presentaciones en tablas.

Revisión del diagrama de instalación.

La siguiente gráfica muestra la estructura de instalación de las paginas html que tenía el sitio web donde se instaló el sistema en la primer versión.

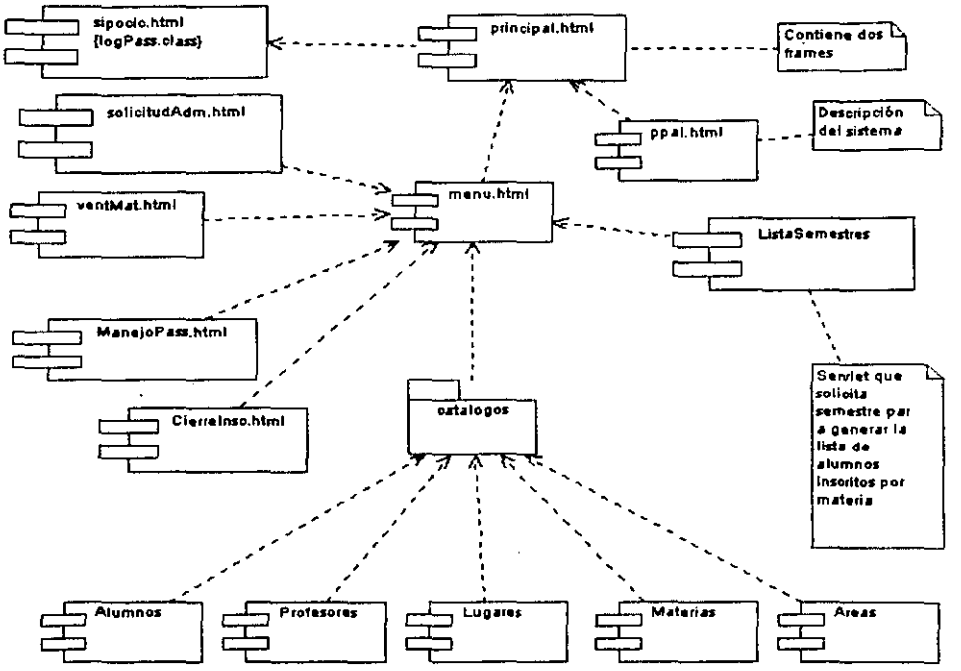


Fig 40. Diagrama de instalación del sistema

Ahora se muestra el diagrama de instalación para ambos tipos de usuarios, resultado de la versión de éste trabajo de tesis.

Fig 41. Diagrama de instalación para el módulo de los usuarios administrativos en la nueva versión.

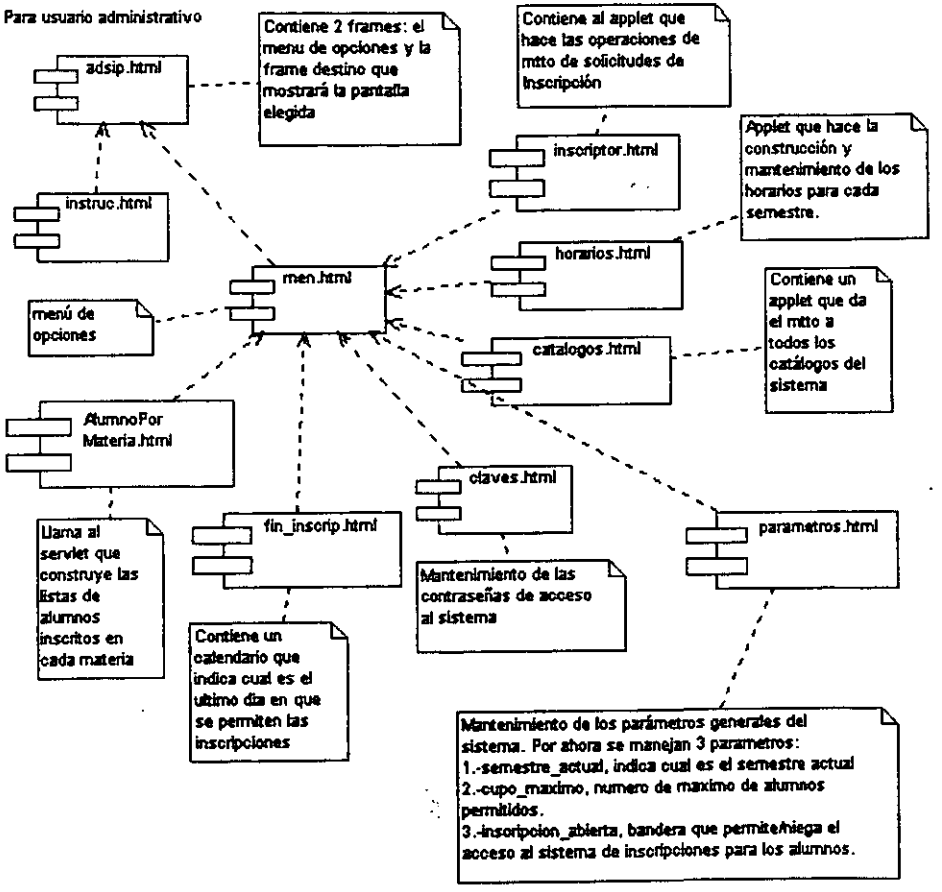
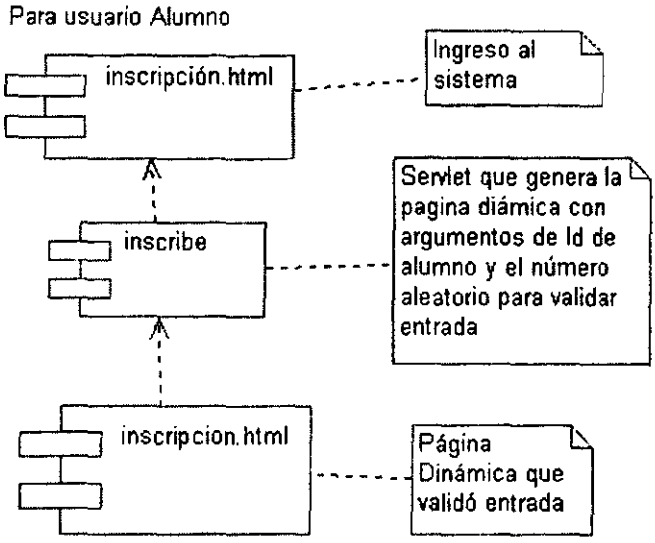


Fig 42 Diagrama de instalación para el módulo de los usuarios alumnos en la nueva



versión

Algunos avisos al usuario en la versión final

Acceso incorrecto del usuario alumno

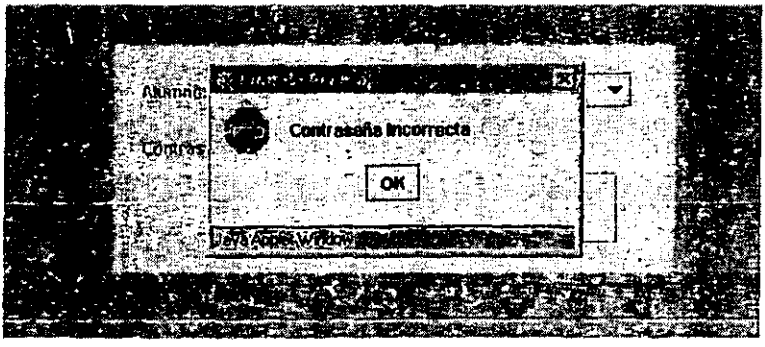


Fig 43. Aviso de contraseña incorrecta para los alumnos en la nueva versión

Inscripción para únicamente 5 materias

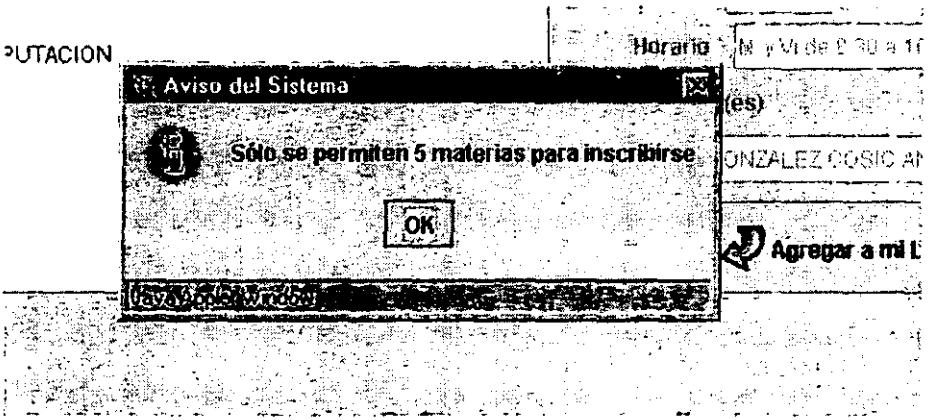


Fig 44. Aviso de cupo máximo de materias para inscripción en la nueva versión del sistema

Cargando el archivo con las clases útiles para la aplicación y conexión a la base de datos. Esto se muestra al inicio del sistema, y sólo en el visualizador *iexplorer*.

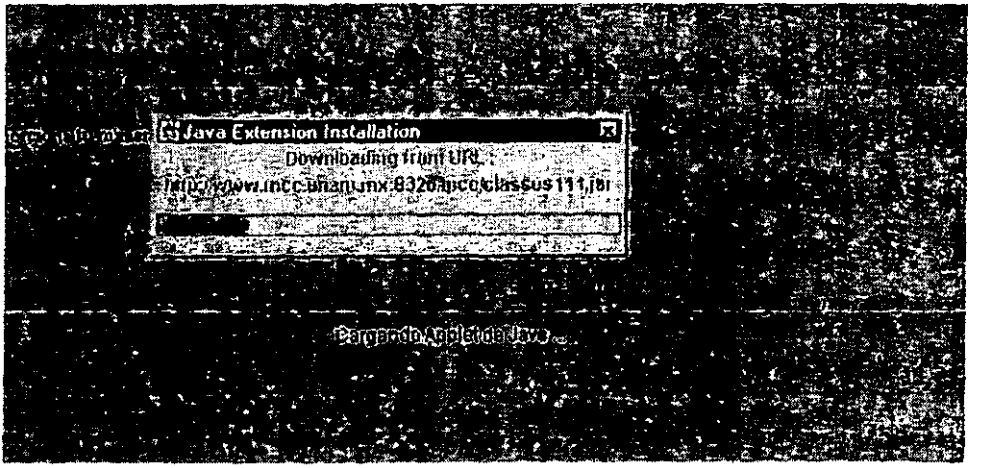


Fig 45. Mensaje de carga de las clases útiles en el visualizador de páginas web.

CONCLUSIONES

A través de la aplicación del Proceso Unificado se demostró que es factible construir un sistema de software de manera ordenada y con relativa velocidad. Los puntos de poco interés en el proceso, como el control de calidad, la planeación de proyectos o la captura de métricas, deberán ser cubiertas apoyándose en otras metodologías o recomendaciones como las del CMM.

La característica del Proceso Unificado de estar basado en casos de uso, es útil en tanto que no se pierden nunca de vista los objetivos de necesidades a satisfacer que son siempre las metas a conseguir al desarrollar un sistema de software.

En el experimento académico se tuvo la restricción del tiempo por lo cual no se logró avanzar más allá del nivel 2 del CMM. Así mismo se tuvo el problema de la falta de experiencia con la metodología del Proceso Unificado, dificultando en ocasiones la interpretación del CMM.

Para el caso de repetir la experiencia de desarrollo de un producto de software, es recomendable arrancar con sesiones que se dediquen al establecimiento de entendimientos comunes para facilitar el proceso de desarrollo. Es altamente recomendable contar con las aportaciones de aquellos que tengan experiencia en el campo.

El presente trabajo reporta las experiencias al desarrollar el sistema, desde un estado prácticamente de cero (tanto en productos como en experiencias con el empleo de una metodología de desarrollo y sin antecedentes de un sistema similar), hasta la conclusión de un ciclo entero de desarrollo. Tal como todos los productos de software, tiene aún varios caminos a seguir para mejorar.

La utilización del paradigma de la programación a objetos, reveló ser útil para el desarrollo de aplicaciones en Internet. Así también el Lenguaje Unificado de Modelación cumplió con sus metas por ser una forma fácil de comprender, proponer y registrar el conocimiento relativo al sistema de software.

El lenguaje de programación Java fue explorado y explotado de forma amplia, mostrándose en todo momento seguro, robusto y eficiente.

Las recomendaciones del CMM deben ser leídas cuidadosamente dada la redacción, en ocasiones confusa, del texto. Y siempre hay que tener en cuenta que el CMM trata de cubrir todos los casos posibles de un proyecto de desarrollo de sistemas, por lo cual deberán tomarse sólo aquellas recomendaciones que redunden en beneficios para el proyecto específico en que se encuentre, sin pretender aplicar todo de forma que se convierta en una carga de trabajo en lugar de una ayuda.

Finalmente se logró construir un sistema de software con capacidad suficiente para dar el servicio de inscripciones semestrales a la comunidad de estudiantes del posgrado, la cual está entre 60 y 100 alumnos.

Queda como trabajos a futuro la construcción de un modulo de instalación porque actualmente habría que estudiar el correspondiente diagrama de instalación para poder implementar el sistema en otro lugar; así también queda por hacer, un módulo que verifique la seriación de las materias a las que tiene acceso cada alumno. Finalmente, si se desea hacer una nueva iteración para el mejoramiento del sistema, habría que procurar tender a crear más parámetros generales que otorguen características más amplias para poder adaptarse a diferentes procesos de inscripción en cualquier entidad académica posible.

APÉNDICE A

ESTÁNDARES EN LOS DOCUMENTOS USADOS

A. CABECERA ESTÁNDAR PARA LOS DOCUMENTOS

Entregable: Procedimientos de Interpretación del Rol de **administrador de calidad**	Ref. AQ004-AQ	Versión 0.1	Pág. 1 / 1

B. PIE ESTÁNDAR PARA DOCUMENTOS

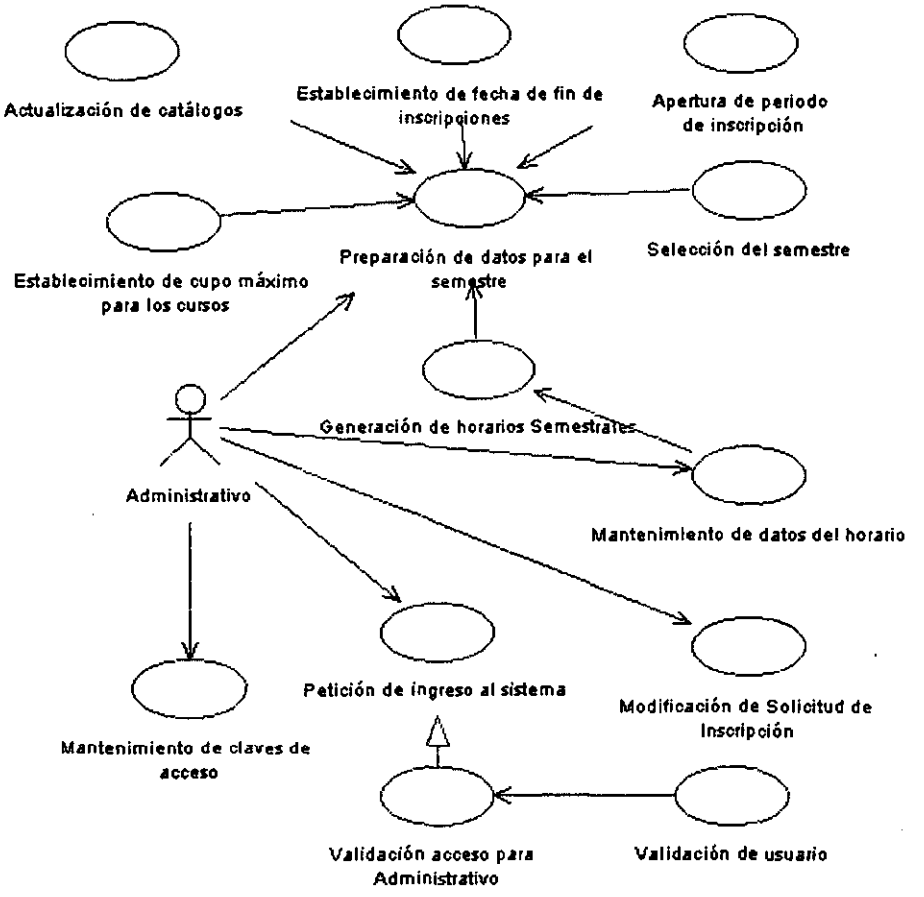
Fase: Elaboración	Sistema: Sistema de Inscripción al Posgrado en Ciencia e Ingeniería de la Computación de la UNAM			B	R	A X
Etapa: Análisis	Realizó: Pablo Ignacio Castillejo García			Fecha: 24/10/1999	
Líder: José Estrada Solís		Calidad: Pablo I. Castillejo G.		Administrador: Rafael Cerecedo Luna.	
Firma:	Fecha:	Firma:	Fecha:	Firma:	Fecha:

APÉNDICE B

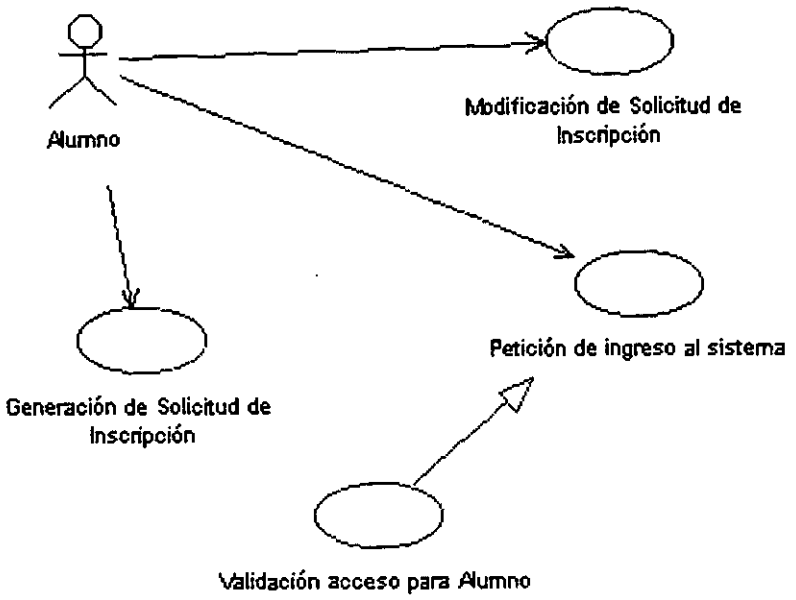
SELECCIÓN DE DIAGRAMAS EN UML DEL SISTEMA

Los Casos de Uso.

Los casos de uso para el usuario administrativo

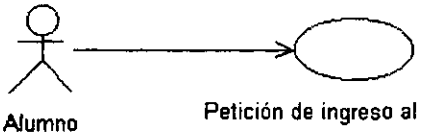


Los casos de uso para el usuario alumno



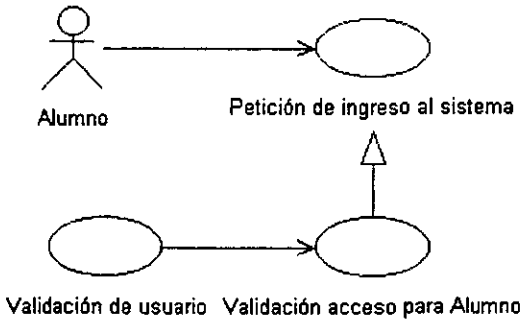
Los casos de uso en detalle

1.- Petición de ingreso al sistema del usuario alumno



1. El usuario alumno solicita ingresar al sistema.
2. Selecciona su nombre de una lista de alumnos en condición de inscribirse.
3. Ingresa su contraseña
4. En caso de ser correcta se le permite el acceso al sistema.

2.- Validación de usuario

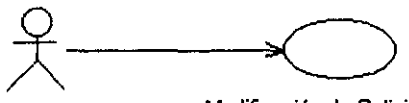


1. El sistema recibe un identificador del tipo de usuario que pide entrar al sistema.
2. Se recibe la contraseña del usuario
3. Se verifica con los datos que tenemos en la Base de datos.
4. Si la contraseña dada coincide con la que se almacena en el sistema, se permite el acceso.

3.- Generación de Solicitud de Inscripción

	1. El alumno escoge las materias que desea cursar. 2. Pide que sea aceptada la solicitud en el sistema. 3. Imprime la hoja de solicitud de inscripción.

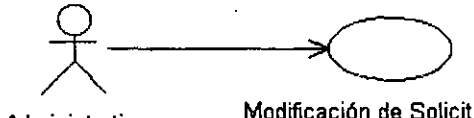
4.- Modificación de la solicitud de inscripción

	1. El alumno solicita modificar su solicitud de inscripción. 2. Si no se ha cerrado el periodo de inscripciones, el sistema permite el acceso 3. El sistema presenta la información almacenada de la solicitud. 4. El alumno quita las materias que ya no desea cursar. 5. Se agregan las materias que desee cursar. 6. Pide que le sea aceptada la solicitud. 7. El sistema verifica cupo y registra los datos de la nueva solicitud. 8. El alumno imprime su nueva solicitud de inscripción

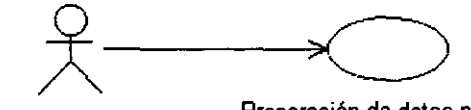
5.- Petición de ingreso al sistema del usuario administrativo

	1. El usuario administrativo solicita ingresar al sistema. 2. Introduce su nombre y contraseña de acceso. 3. En caso de ser correcta se le permite el acceso al sistema, en el modelo de mantenimiento general.

6.- Modificación de Solicitud por parte del usuario administrativo

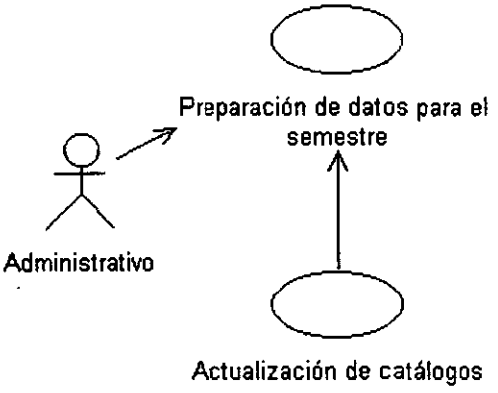
	1. El usuario administrativo entra al sistema. 2. Accesa a la parte de mantenimiento de solicitudes de inscripción. 3. Selecciona de la lista de alumnos al que desea modificarle sus datos. 4. Hace los cambios que desee. 5. Pide actualizar los datos almacenados en la base de datos.

7.- Preparación de datos del semestre.

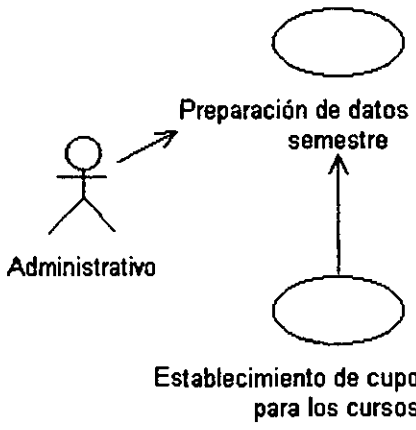
	1. El usuario ingresa al sistema. 2. Cierra el acceso al sistema para los usuarios alumnos. 3. Establece el nuevo semestre vigente. 4. Actualiza los catálogos de materias, profesores y alumnos. 5. Construye el horario para el

	semestre. 6. Establece la fecha de fin de inscripciones. 7. Abre las inscripciones en la fecha que convenga a la institución.

8.- Actualización de catálogos del sistema

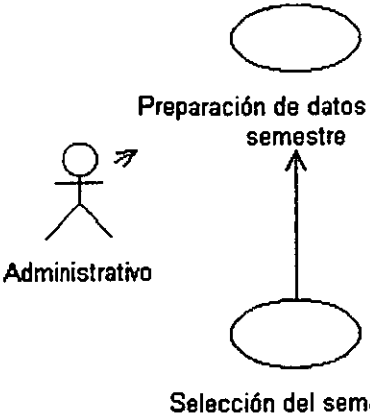
	1. El usuario ingresa al sistema. 2. Se ingresa a los catálogos del sistema 3. Elige el catálogo a modificar. 4. Modifica los datos que considere 5. El sistema debe avisar si se pueden cambiar los datos, pues por ejemplo no podría modificarse el grupo de una materia que ya tiene un registro de un alumno inscrito. 6. Actualiza los datos almacenados en la base de datos.

9.- Establecimiento de Cupo Máximo



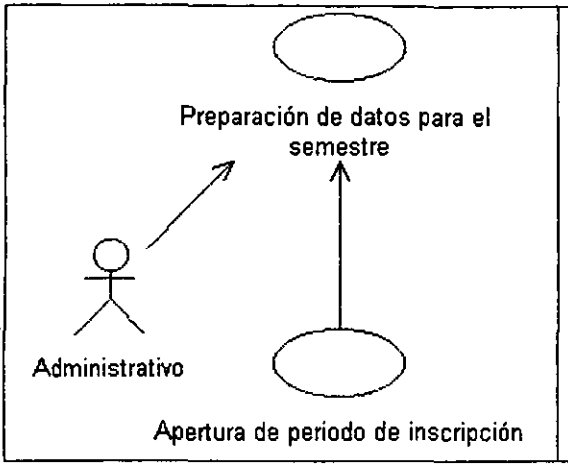
1. El usuario ingresa al sistema.
2. Solicita ver la fecha de último día de inscripciones.
3. Lo modifica si desea.
4. El sistema deberá ver si es posible modificar la fecha. En cuyo caso deberá avisar el resultado de la validación.

10.- Selección del semestre actual



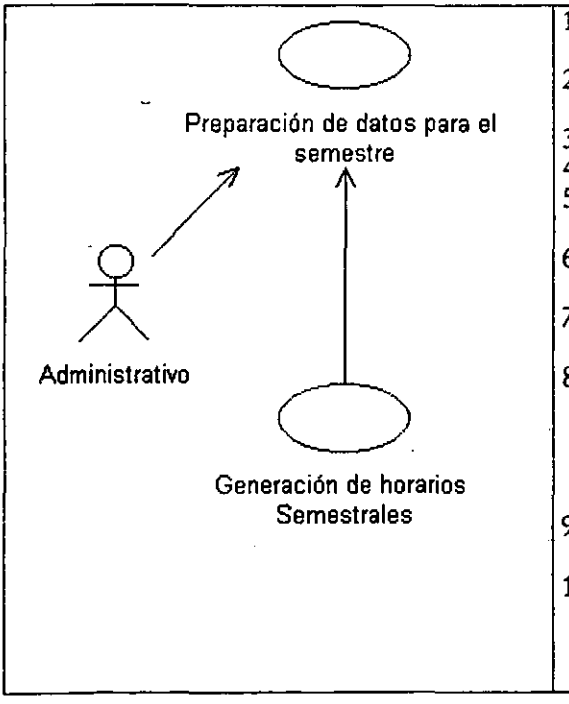
1. El usuario ingresa al sistema.
2. Verifica cual es el semestre actual registrado en el sistema.
3. Lo modifica si es necesario.
4. El sistema verifica si la clave del semestre es válida.
5. Si la clave del semestre es válida, se establece como el semestre actual que el sistema tomará para todas las operaciones.

11.- Apertura de Inscripción



1. El usuario ingresa al sistema
2. Verifica si el sistema permite el acceso a los alumnos.
3. Establece la apertura del acceso al sistema con lo cual inaugura el periodo de inscripciones.
4. El sistema no tiene nada que verificar.

12.- Generación de Horarios Semestrales



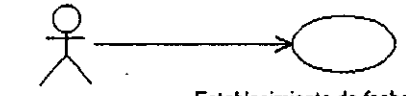
1. El usuario administrativo ingresa al sistema.
2. Entra al apartado de horarios semestrales.
3. Elige una materia.
4. Escribe grupo.
5. Asigna profesores, uno o dos, que darán el curso.
6. Escribe salón donde se impartirá el curso.
7. Se asigna la institución que ofrecerá el curso.
8. Si la materia va a pertenecer a otra área de concentración que la que tiene asignada, o debe pertenecer a otra más, se seleccionan las áreas a las que pertenecerá la materia.
9. Se almacena la información en la base de datos.
10. El sistema debe validar que todos los datos sean validos, en especial la materia, los profesores, los lugares que ofrecen el curso y las

	áreas de concentración a las que pertenecerá el curso. 11. Genera la página en html para eventualmente colocarla en el servidor web de la institución para la consulta de la comunidad estudiantil.

13.- Mantenimiento de datos del horario.

	1. El usuario ingresa al sistema 2. Observa los datos que quiere cambiar del horario. 3. Realiza los cambios que crea pertinentes. 4. Guarda la información en la base de datos. 5. El sistema deberá verificar que los cambios son válidos, al revisar en sus registros si no hay gente inscrita en el curso, en cuyo caso la modificación no sería posible.

14.- Establecimiento de fecha de fin de inscripciones

	1. El usuario ingresa al sistema 2. Observa cual es la fecha que está establecida como fin de inscripciones. 3. La modifica si es el caso.

15.- Mantenimiento de claves de acceso

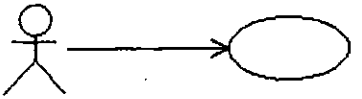
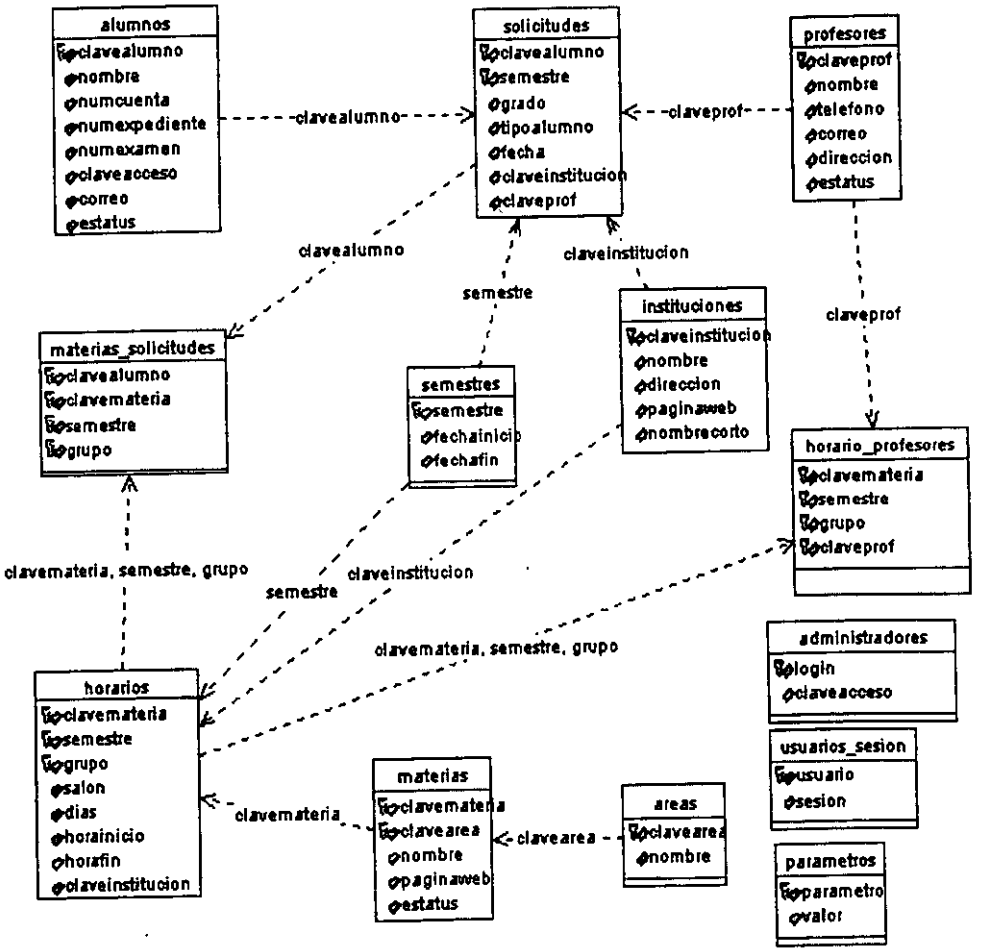
	1. Se identifica el alumno al cual se le modificará la clave de acceso 2. Se cambia la clave 3. Se informa del cambio personalmente..

Diagrama de clases del dominio del problema



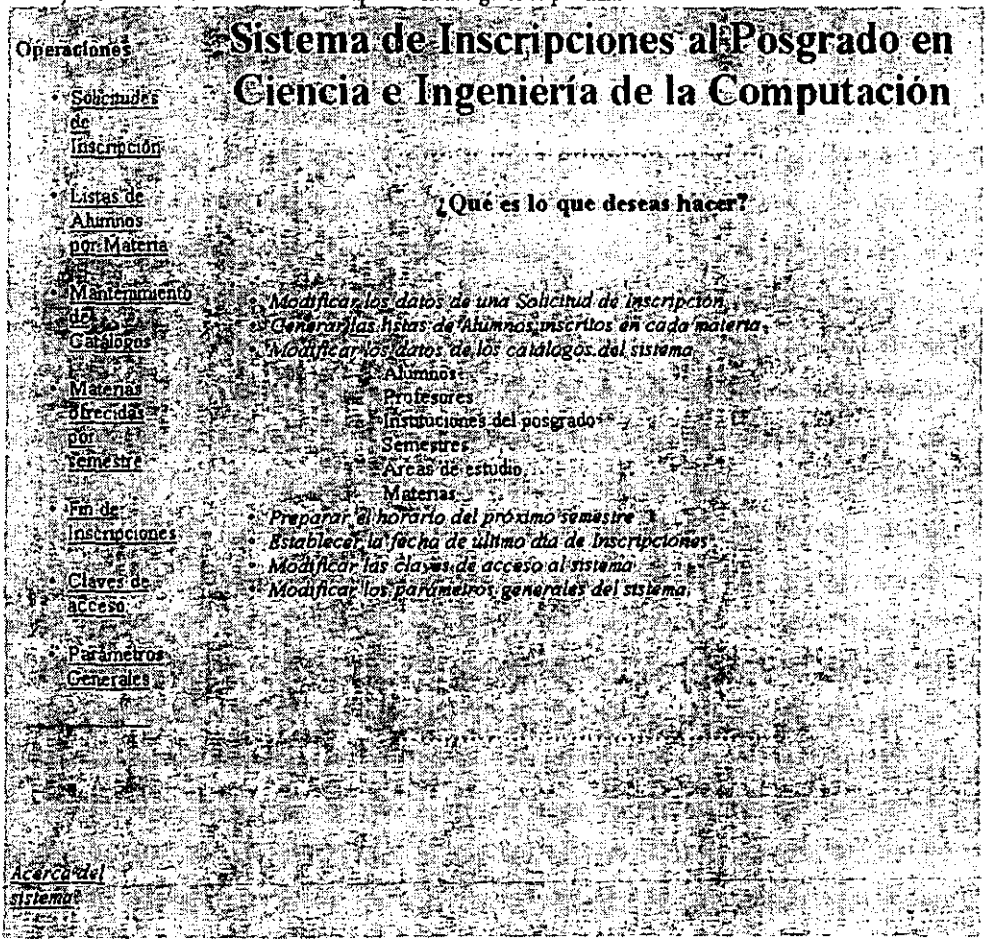
Contenido

I. Ingreso al sistema

a) Ingresa a la página inicial del sistema para los usuarios administrativos http://www.mcc.unam.mx/inscripciones/adsip.html

b) Se pedirá el nombre de usuario y contraseña. Tanto el nombre como la clave serán las mismas que se utilizan para ingresar al servidor uxmcc2

c) Si la contraseña es correcta aparecerá la siguiente pantalla



d) IMPORTANTE: Cuando salga del sistema, deberá cerrar el visualizador para terminar la sesión.

II. El menú de opciones

Se contemplan las siguientes opciones:

- *Solicitudes de Inscripción*

Permite hacer alta o modificaciones de la información de las solicitudes de inscripción de todos los alumnos del posgrado.

- *Listas de alumnos por materia*

Visualiza e Imprime las listas de los alumnos inscritos en las materias.

- *Mantenimiento de catálogos*

Agrega, elimina, o modifica la información, de

- Alumnos
- Profesores
- Instituciones
- Semestres, y
- Materias

- *Materias ofrecidas por semestre*

Construye o modifica la información sobre las materias que son ofrecidas en cada semestre.

- *Fin de inscripciones*

La operación de inhibir la entrada de los alumnos al sistema de información, la establece el sistema mismo. En esta pantalla se establece la fecha en que serán cerradas las inscripciones.

- *Claves de acceso*

Cambia las contraseñas de acceso para los usuarios del sistema.

- *Parámetros generales*

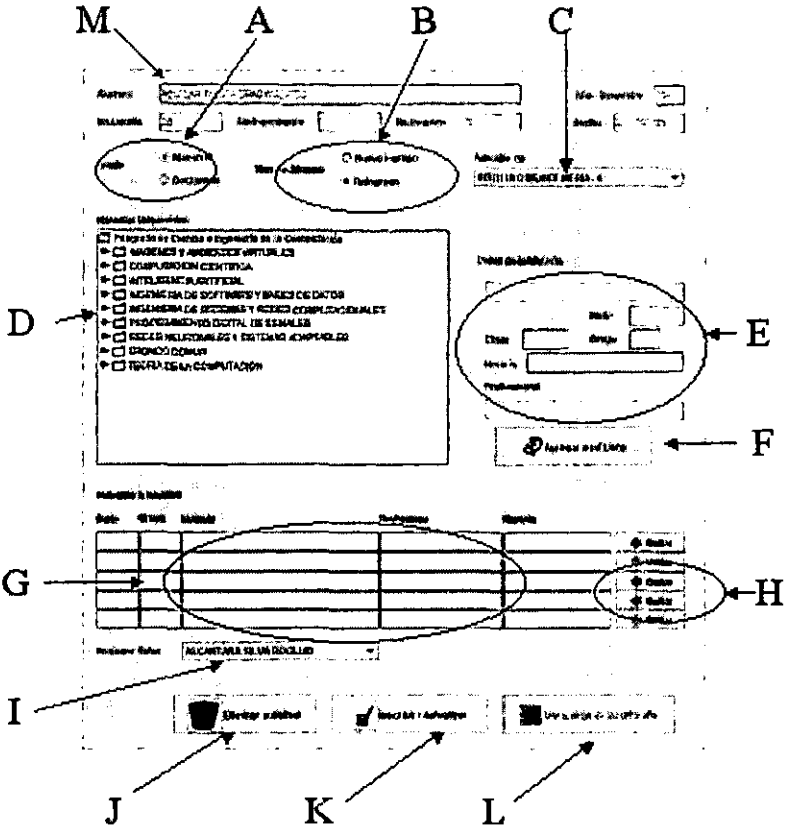
Modifica los parámetros generales del sistema.

- Semestre actual
- Cupo máximo
- Apertura de inscripción

Veremos ahora en detalle lo que puede hacerse en cada opción

i. Solicitudes de Inscripción

Refiérase a la siguiente figura



Dar de alta la solicitud de un alumno.

a) Seleccione el alumno de la lista de alumnos. Señalado con M.
b) Establezca su grado académico, sección A.
c) Establezca el tipo de alumno, de nuevo ingreso o reingreso, sección B.
d) Indique en que institución está adscrito el alumno, sección C.
e) Elija la materia en la sección D, en la que se le inscribirá al alumno. La información correspondiente aparecerá en la sección E.
f) Agregue a la lista del alumno (sección G), la materia elegida, con el botón F.
g) Por cada materia a agregar, repita incisos e y f.
h) Si desea quitar alguna materia, oprima el botón correspondiente al renglón, en la sección H.
i) Una vez que se tengan las materias en la lista, sección G, asigne el tutor correspondiente, señalado con I.
j) Inscriba al alumno oprimiendo el botón de inscripción o actualización, señalado con K.

Eliminar la solicitud de un alumno.

a) Seleccione al alumno de la lista de alumnos, señalada con M.

b) Oprima el botón Eliminar solicitud, señalado con J.

Imprimir la solicitud de un alumno.

a) Seleccione al alumno de la lista de alumnos, señalada con M.
b) Oprima el botón Ver hoja de Inscripción, señalado con L.

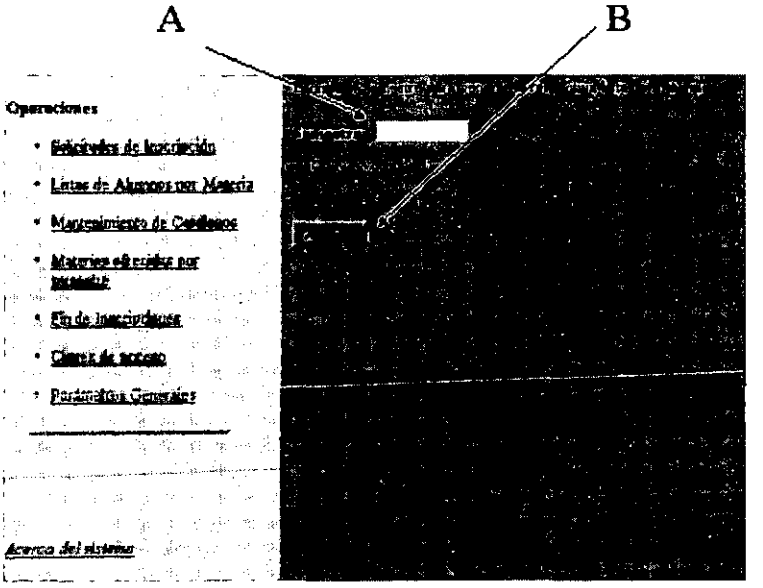
Modificar la solicitud de un alumno

a) Seleccione al alumno de la lista de alumnos, señalada con M.
b) Si tiene materias inscritas, éstas aparecerán en la lista de materias, sección G.
c) Si quiere quitar materias de la lista, oprima el botón correspondiente al renglón de dicha materia, en la sección H.
d) Si quiere agrega una materia, selecciónela del árbol de materias (indicado con D) y agregue la materia con el botón Agregar a mi lista (indicado con F).
e) Por cada materia a agregar repita inciso d.
f) Finalmente actualice la información del alumno oprimiendo el botón Inscribir o Actualizar, señalado con K.

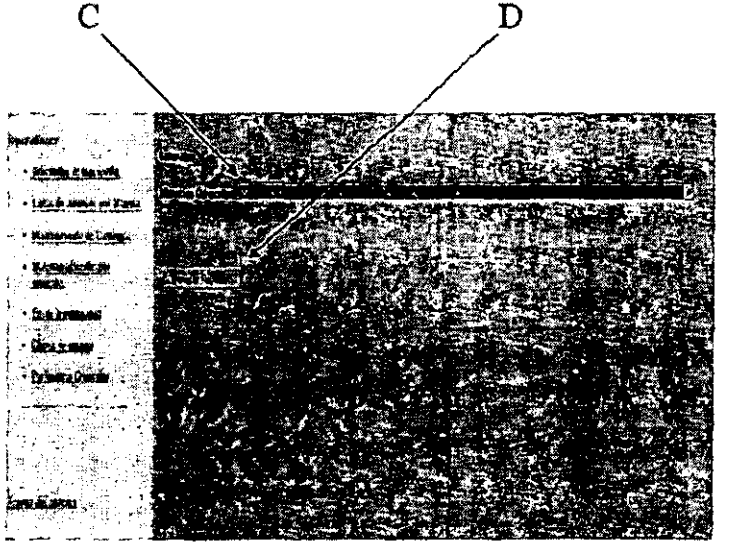
ii. Listas de alumnos por materia

Visualización de los alumnos inscritos en una materia

Refiérase a la siguiente figura



a) Indique el semestre en la parte A, con el formato numero de dos dígitos seguido de guión seguido de numero romano I o II.
b) Oprima el botón Aceptar, señalado con B.
c) Refiérase a la siguiente figura. Seleccione la materia (señalado con C) para la cual desea generar la lista de alumnos inscritos.
d) Oprima el botón de generar lista (señalado con D).



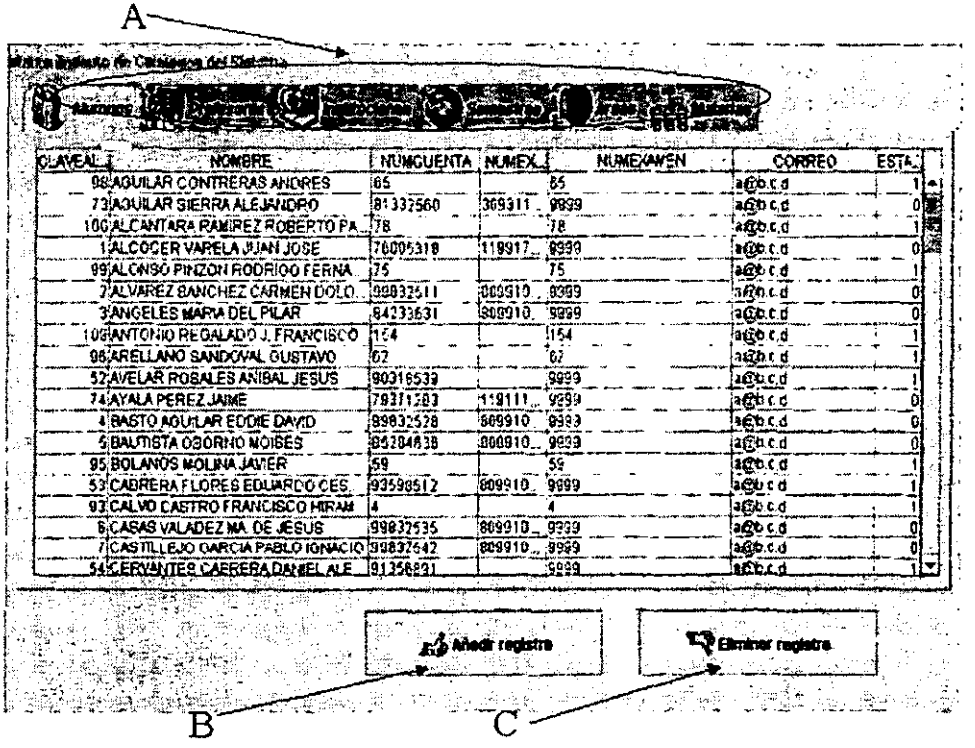
Impresión de la lista de los alumnos inscritos en una materia

a) Una vez hecha la operación descrita anteriormente, imprima la página visualizada desde el menú del visualizador.

iii. Mantenimiento de catálogos

Por registro se entenderá un elemento del catálogo referido, es decir un registro en el catálogo de materias corresponderá a una materia, un registro en el catálogo de alumnos será un alumno, etc.

Refiérase a la siguiente figura



CLAVEAL	NOMBRE	NUMCUENTA	NUMEX	NUMEXAMEN	CORREO	ESTA
98	AGUILAR CONTRERAS ANDRES	65		65	a@b.c.d	1
73	AGUILAR SIERRA ALEJANDRO	81337560	369311	9999	a@b.c.d	0
100	ALCANTARA RAMIREZ ROBERTO PA	78		78	a@b.c.d	1
1	ALCOCER VARELA JUAN JOSE	76005318	119917	9999	a@b.c.d	0
99	ALONSO PINZON RODRIGO FERNA	75		75	a@b.c.d	1
7	ALVAREZ SANCHEZ CARMEN DOLO	99832511	809910	9999	a@b.c.d	0
3	ANGELES MARIA DEL PILAR	84233631	809910	9999	a@b.c.d	0
100	ANTONIO REGALADO J. FRANCISCO	154		154	a@b.c.d	1
96	ARELLANO SANDOVAL GUSTAVO	67		67	a@b.c.d	1
52	AVELAR ROSALES ANIBAL JESUS	90316539		9999	a@b.c.d	1
74	AYALA PEREZ JAIME	79371583	119111	9999	a@b.c.d	0
4	BASTO AGUILAR EDDIE DAVID	99832528	809910	9999	a@b.c.d	0
5	BAUTISTA OSORNO MOISES	86284838	809910	9999	a@b.c.d	0
95	BOLAÑOS MOLINA JAVIER	59		59	a@b.c.d	1
53	CABRERA FLORES EDUARDO CES	93598512	809910	9999	a@b.c.d	1
97	CALVO CASTRO FRANCISCO MIRAM	4		4	a@b.c.d	1
6	CASAS VALADEZ MA. DE JESUS	99832535	809910	9999	a@b.c.d	0
7	CASTILLEJO GARCIA PABLO IGNACIO	99832542	809910	9999	a@b.c.d	0
54	CERVANTES CARRERA DANIEL ALE	91358931		9999	a@b.c.d	1

Dar de alta un nuevo registro

a) Seleccione el catálogo al cual desea añadirle un registro.
b) Oprima el botón Añadir registro, señalado con B.
c) Abra la ventana que aparece para pedir la clave con que se dará de alta el nuevo registro.
d) Si el catálogo es diferente al de alumnos, ingrese la clave del registro en la ventana que aparece.
e) La clave a ingresar será según la tabla siguiente

Si es el catálogo de	La clave a ingresar deberá ser
Alumnos	Ninguna ya que el sistema la genera.
Profesores	El RFC del profesor
Instituciones	Un número de 4 dígitos que identifique a la institución
Semestres	El semestre con formato dos dígitos, guión, numero romano I o II. Por Ejemplo 01-I.
Áreas	Dos letras que identifiquen al área de estudios. Por ejemplo IS para Ingeniería de Software.
Materias	Un número de 5 dígitos que representa la clave de la materia. Éste valor será el que aparezca como clave en las solicitudes de inscripción.

Eliminar un elemento

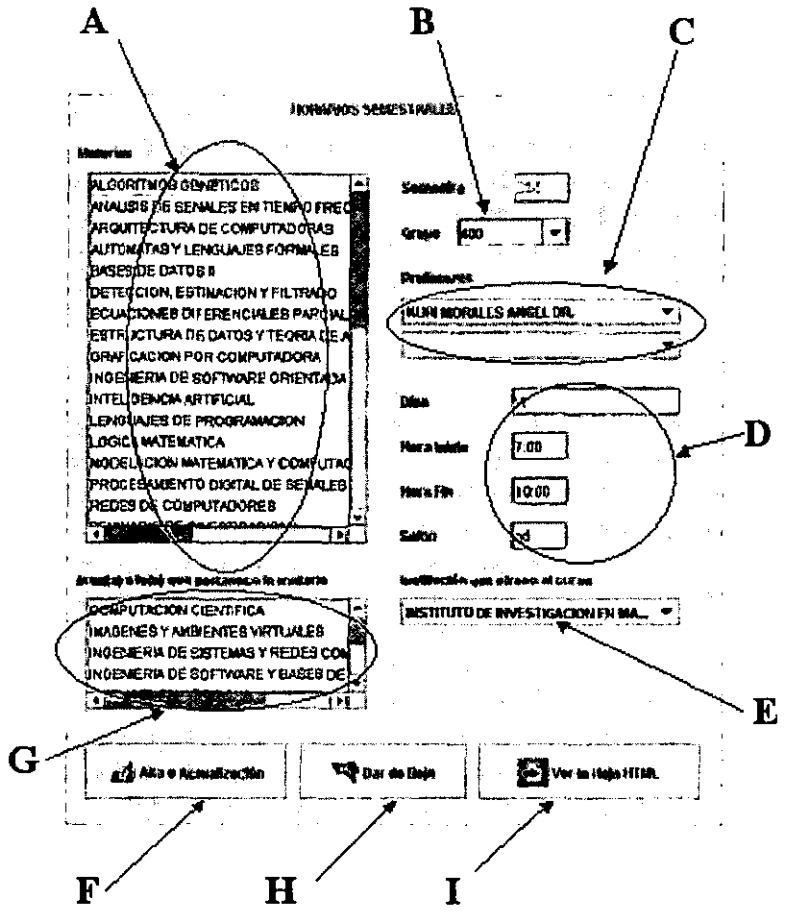
a) Seleccione el catálogo del que desea eliminar un registro (sección A).
b) Seleccione el elemento en la tabla.
c) Oprima el botón Eliminar registro, señalado con C.
d) Si el registro tiene datos históricos (por ejemplo una materia con registros de haber sido dada en algún semestre), no se podrá eliminar y se avisará al usuario en un mensaje.

Modificar un registro

a) Seleccione el catálogo en la sección A.
b) Busque el registro que quiere modificar.
c) Haga doble clic en el campo que modificará.
d) Introduzca el nuevo valor.
e) OPRIMA ENTER o haga clic en otro valor, para guardar el cambio hecho.

iv. Materias ofrecidas por semestre

Refiérase a la siguiente figura



<u>Dar de alta un curso</u>

a) Seleccione la materia de la lista señalada con A.
b) Escriba el número que identificará al grupo, en la parte señalada con B.
c) Asigne al menos un profesor para la materia, pudiendo indicar dos, en la sección C.
d) Asigne valores de días en que se dará la materia, hora de inicio, hora de fin y salón, en la sección D.
e) Indique que institución ofrece el curso en la parte E.

f) Si quiere cambiar el área de concentración de la materia, selecciónela en la parte G. Si quiere asignar dos o más áreas para la materia elíjalas en la lista haciendo clic y oprimiendo la tecla control.
g) Oprima el botón F.
h) Por cada grupo que se ofrezca en el semestre repita los pasos de los incisos b al g.

Eliminar un curso

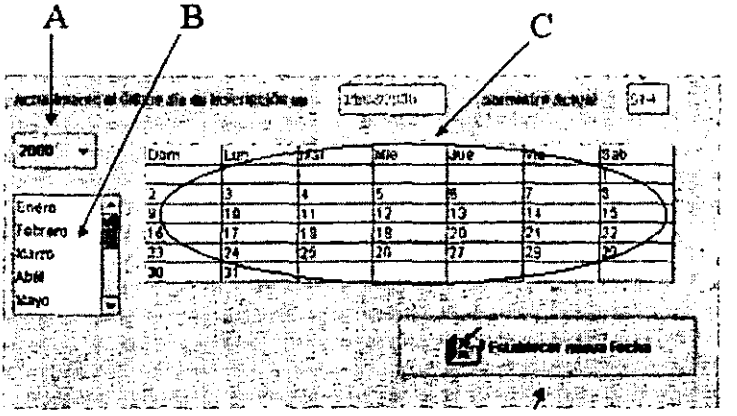
a) Seleccione la materia de la lista señalada con A.
b) Elija el grupo que desea eliminar, en la sección B.
c) Oprima el botón Dar de baja, señalado con H.
d) Si no hay alumnos inscritos a la materia, se dará de baja, en otro caso no se puede dar de baja porque ya hay alumnos inscritos al curso. Refiérase a la sección *Referencia rápida para los procesos más comunes* de éste mismo manual.

Generar la página html de los cursos

a) Oprima el botón Ver la hoja HTML señalada con I.

v. Fin de inscripciones

Refiérase a la siguiente figura



Consulta de la fecha de cierre de inscripciones

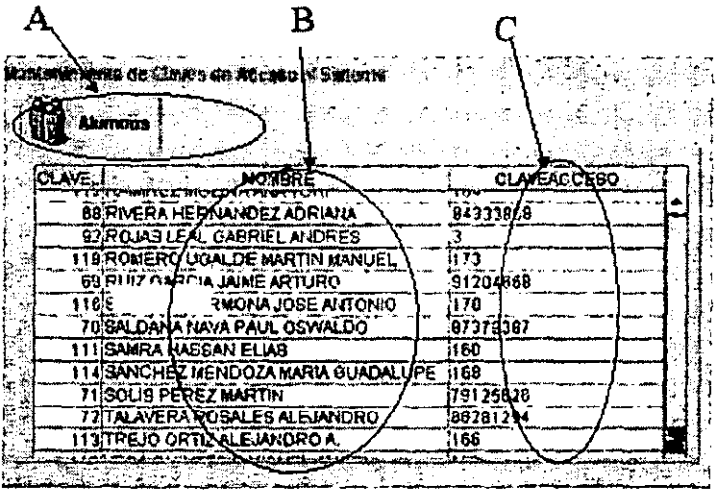
a) Elija en el menú la opción Fin de Inscripciones.
b) La fecha de fin de inscripción aparece en la parte superior de la pantalla.

Establecimiento de nueva fecha de fin de inscripciones

a) Elija el año, sección A.
b) Elija el mes, sección B.

c) Haga clic en el día que desea cerrar las inscripciones, en la sección C.
d) Oprima el botón Establecer nueva Fecha.
e) Se le pedirá confirmar la nueva fecha.

vi. Claves de acceso
Refiérase a la siguiente figura.



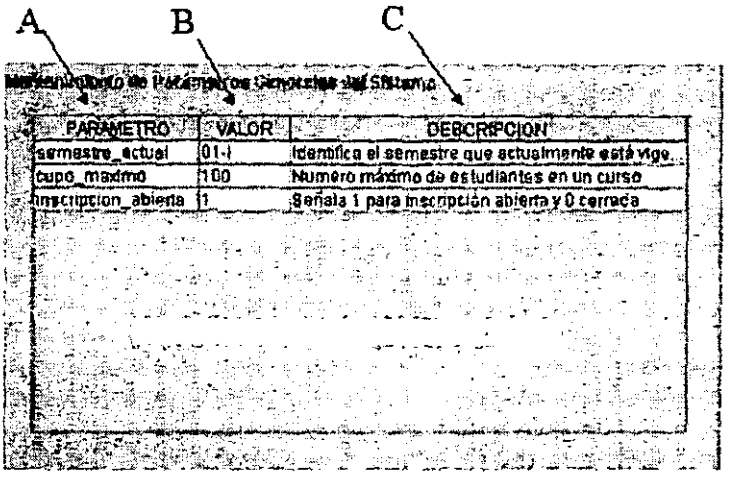
Modificación de la contraseña de un alumno

a) Busque al Alumno en la lista, sección B.
b) Haga doble clic en la clave correspondiente, sección C.
c) Modifique el valor y OPRIMA ENTER.

Modificación de la contraseña de un usuario administrativo

a) Actualmente la clave de acceso al sistema para el usuario administrativo es la misma que ocupan para ingresar al servidor *ucmx2.iimas.unam.mx*, por lo tanto si se desea cambiar la clave de acceso, deberá solicitarse al administrador del servidor

vii. Parámetros generales
Refiérase a la siguiente figura.



Estableciendo el semestre actual para todo el sistema

a) Seleccione la celda de valor para el renglón del parámetros semestre_actual.
b) Escriba el valor del semestre actual.
c) El semestre debe estar dado de alta en el catálogo de semestres. Opción de menú catálogos, pestaña de semestres.

Establecimiento del cupo máximo para todas las materias

a) Seleccione la celda de valor en el renglón cupo_máximo.
b) Escriba el nuevo valor.

Abrir o cerrar el periodo de inscripciones

a) Elija la celda de valor del parámetro inscripción abierta.
b) Escriba cero para inhibir la entrada de alumnos al sistema. Escriba uno para permitir el acceso de los alumnos al sistema.

III. Referencia rápida para los procesos más comunes.

Activar a un alumno

Elija en el menú la opción *Catálogos* y elija la pestaña Alumnos, si el alumno ya existe cambie su estatus a 1 (activo), si no está, agréguelo.

Consultar la solicitud de un alumno

Elija en el menú la opción de Solicitudes de Inscripción. Seleccione al alumno

Imprimir la lista de alumnos inscritos en una materia

Elija en el menú la opción de *Lista de alumnos por materia*. Ingrese el semestre y elija la materia.

Abrir y cerrar las inscripciones

Para abrir el periodo de inscripciones, elija en el menú la opción parámetros y establezca en 1 el renglón que dice inscripción_abierta.

Para cerrar las inscripciones, deberá establecer la fecha de fin de inscripciones. El sistema se encargará de cerrar las inscripciones en el día indicado. Elija en el menú la opción de Fin de Inscripción y establezca la fecha.

Dar de alta una nueva materia

Seleccione en el menú la opción catálogos, vaya a la pestaña de materias y oprima el botón añadir registro, escriba la clave de la materia. El nuevo registro aparecerá al final de la lista. Seleccione los valores que desea cambiar como el nombre o la referencia a la página web.

Cambiar las áreas a las que pertenece una materia

Elija en el menú la opción de *Materias ofrecidas por semestre* y elija la materia. Asigne las áreas en que estará la materia y oprima el botón de alta o actualización

IV. Preparando un nuevo periodo de inscripciones

Si se desea preparar el sistema para otro periodo de inscripciones, síganse los siguientes pasos:

- Cerrar el acceso de los alumnos al sistema, lo cual se hace con la opción *Parámetros Generales* del menú, y se establece en 0 el registro del parámetro "*inscripcion_abierta*".

- Verificar que esté dado de alta el semestre, en la opción *Mantenimiento de catálogos* del menú. Si no está el semestre entonces agregarlo a la lista correspondiente.

- Indicar cual será el nuevo semestre vigente, lo cual se hace con la opción *Parámetros generales* del menú, escribiendo el semestre correcto en el registro de "*semestre_actual*".

- Actualizar los catálogos de profesores, alumnos y materias, que se realiza con la opción *Parámetros generales* del menú.

- Construir los horarios para el semestre en cuestión. Operación que se efectúa con la opción *Materias ofrecidas por semestre*, del menú. Si se desea generar una página html con la información resultante, se puede obtener desde la pantalla mencionada.

- Establecer la fecha del fin de las inscripciones. Que se hace con la pantalla mostrada en la opción *Fin de inscripciones*, del menú.

- Abrir las inscripciones en la fecha que establezca la institución, lo cual se hace modificando nuevamente el parámetro general de "***inscripción_abierta***", en la opción *Parámetros generales* del menú.

V. Posibles errores

a) **No puedo entrar al sistema**
La validación de acceso se hace a través del servidor de http. En este caso habrá que acudir con el administrador del servidor.

b) **No puedo generar los archivos de alumnos inscritos a las materias.**
Revise que el semestre está dado de alta en la base de datos, opción de menú **catálogos**, pestaña de semestres.
Revise que está ingresando bien el semestre, con el formato número de dos dígitos, seguido de guión, seguido de número romano uno o dos.

c) **No sé que valor dar al nuevo registro en catálogos**

Refiérase a la tabla de la sección "Dar de alta un nuevo registro" de éste manual.

d) **No aparece una materia que quiero agregar dar de alta como curso disponible para el semestre.**

Asegúrese de que la materia tiene estatus de activa (número 1) en el catálogo de materias, menú catálogos, pestaña materias, columna de estatus.

e) **Quiero eliminar un grupo pero no es posible**

Si el sistema no permite borrar el curso es porque ya hay alumnos inscritos en él.

Primero habría que ver quienes están inscritos con la opción **Lista de alumnos por materia,** después se eliminaría la materia de sus solicitudes en la opción **Solicitudes de Inscripción,** finalmente se elimina la materia de los cursos ofrecidos desde la opción **Materias ofrecidas por semestre.**

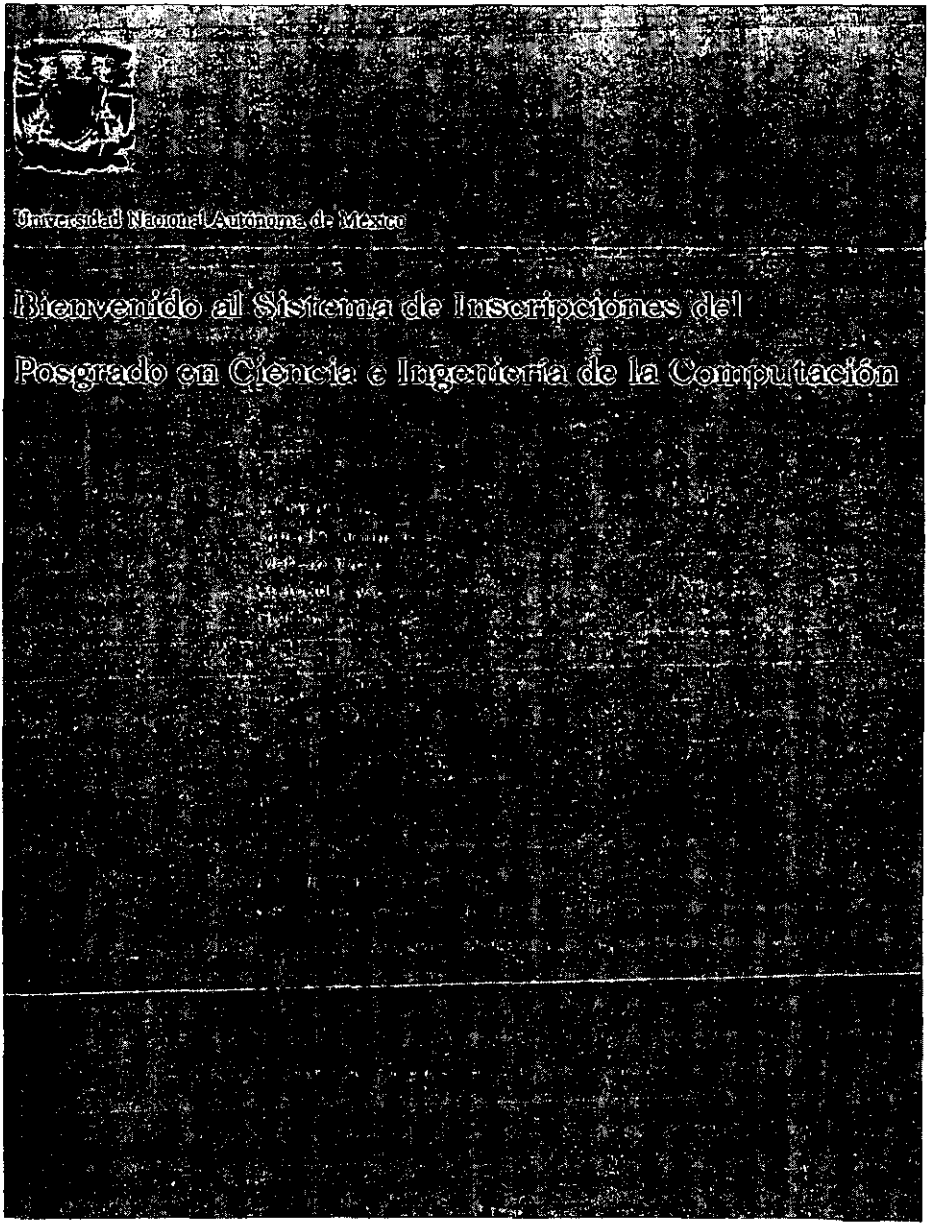
VI. Referencias para problemas no previstos

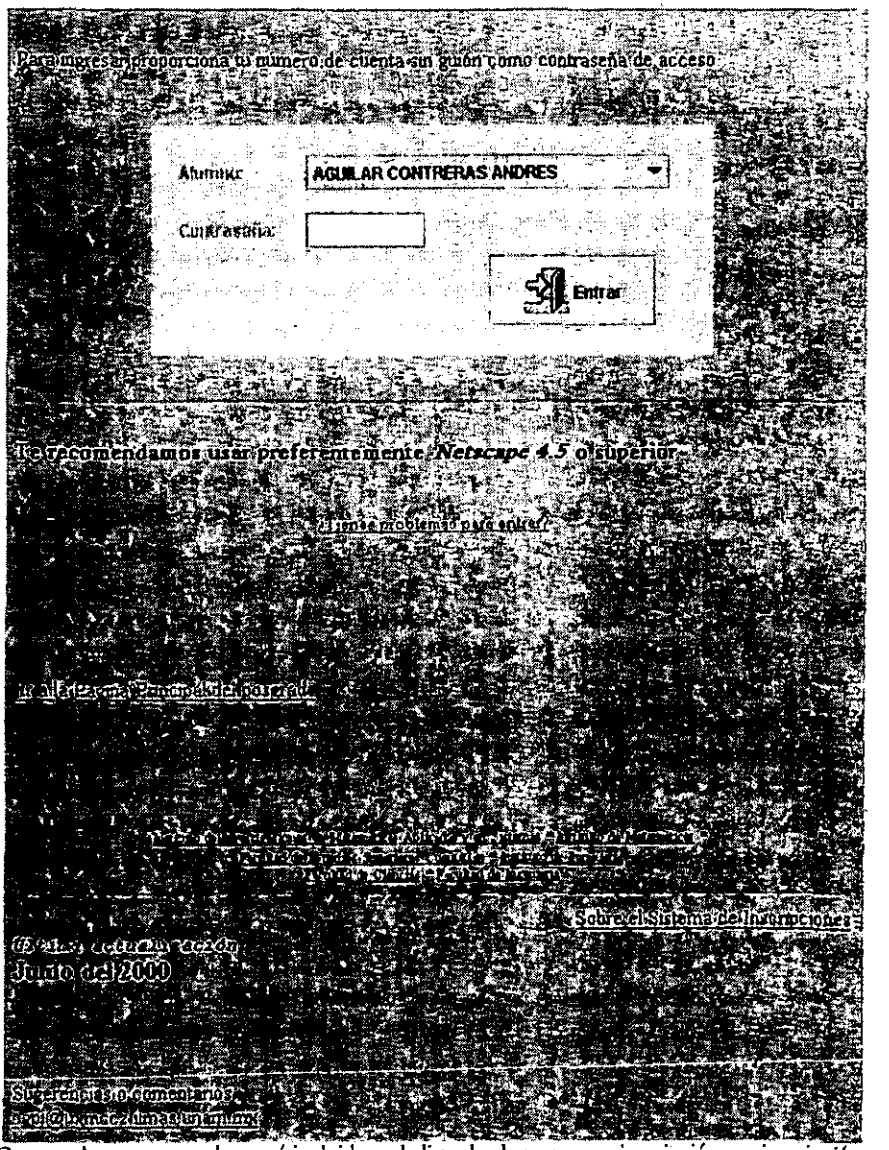
En caso de tener otro problema no previsto aquí, reportarlo con:

- Correo electrónico administrador del servidor y la red: Dante Ortiz dortiz@uxmcc2.iimas.unam.mx
- Correo electrónico creador de la versión del sistema: Pablo Castillejo cgpi@uxmcc2.iimas.unam.mx

I. Ingreso al sistema

d) Ingresa a la página inicial del sistema
http://www.mcc.unam.mx:8326/inscripcion.html

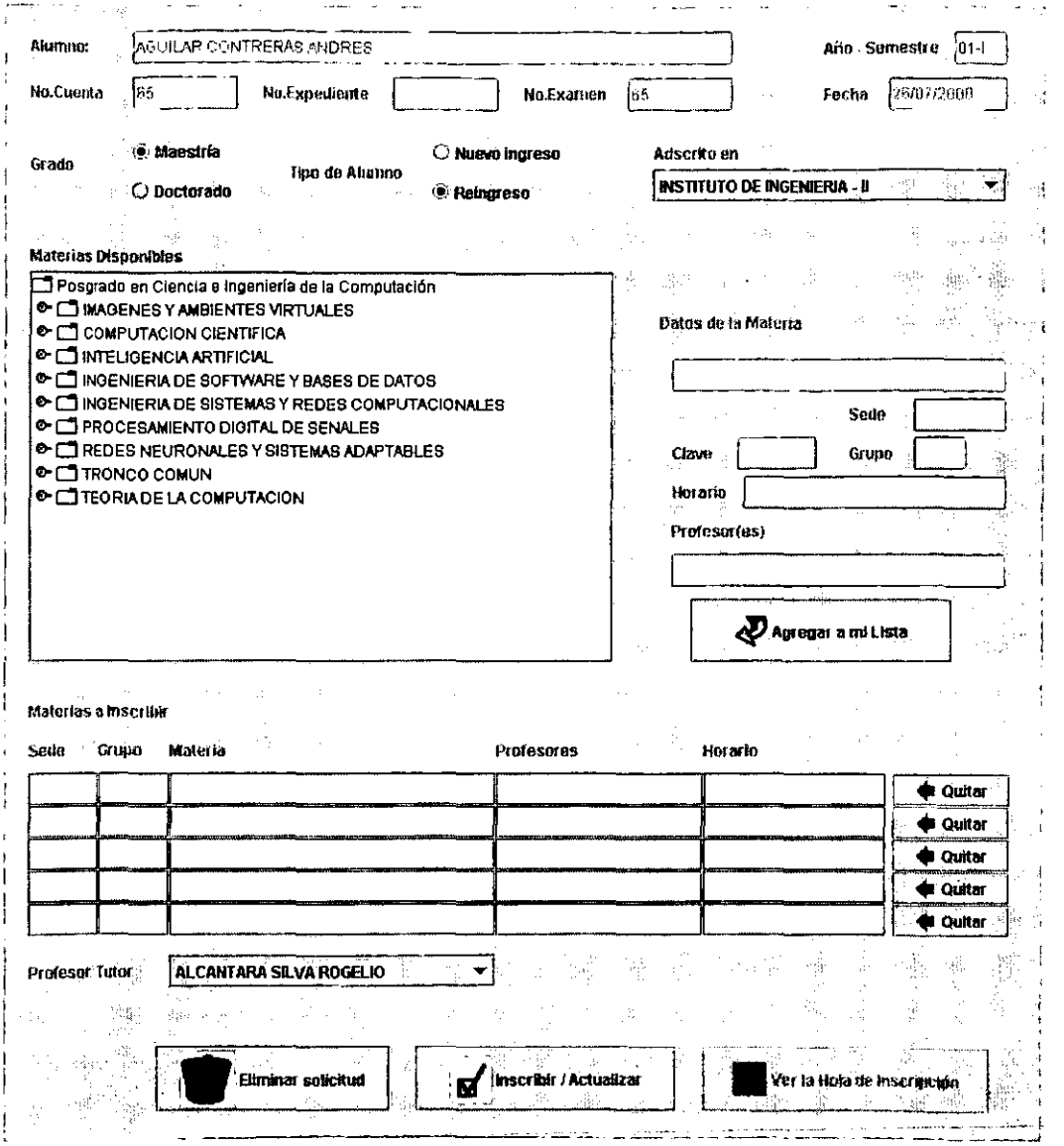




e) Comprueba que tu nombre esté incluido en la lista de alumnos para inscripción o reinscripción

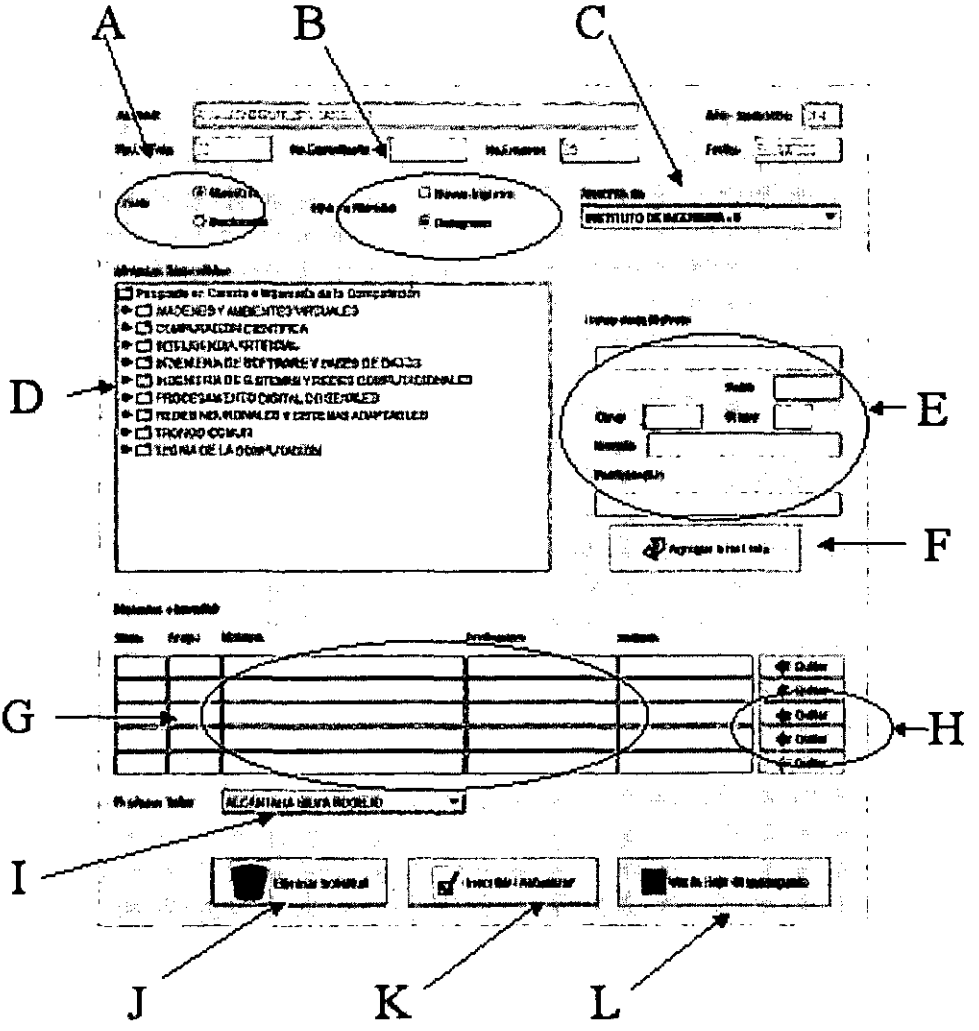
f) Introduce la contraseña de acceso

g) Si la contraseña es correcta aparecerá la siguiente pantalla


Alumno: AGUILAR CONTRERAS ANDRES
Año - Semestre 01-I
No.Cuenta 65
No.Expediente
No.Examen 65
Fecha 26/07/2000
Grado
Maestría
Doctorado
Tipo de Alumno
Nuevo Ingreso
Reingreso
Adscrito en
INSTITUTO DE INGENIERIA - II
Materias Disponibles
Posgrado en Ciencia e Ingeniería de la Computación
IMAGENES Y AMBIENTES VIRTUALES
COMPUTACION CIENTIFICA
INTELIGENCIA ARTIFICIAL
INGENIERIA DE SOFTWARE Y BASES DE DATOS
INGENIERIA DE SISTEMAS Y REDES COMPUTACIONALES
PROCESAMIENTO DIGITAL DE SENALES
REDES NEURONALES Y SISTEMAS ADAPTABLES
TRONCO COMUN
TEORIA DE LA COMPUTACION
Datos de la Materia
Sede
Clave
Grupo
Horario
Profesor(es)
Agregar a mi Lista
Materias a Inscribir
Sede
Grupo
Materia
Profesores
Horario
Quitar
Quitar
Quitar
Quitar
Quitar
Profesor Tutor
ALCANTARA SILVA ROGELIO
Eliminar solicitud
Inscribir / Actualizar
Ver la Hoja de Inscripción

II. Solicitando inscripción.

Refiérase a la siguiente figura



a) Selecciona el grado académico que te corresponde. Señal A.

b) Selecciona tu condición de alumno, de ingreso o reingreso. Señal B.

c) Indica en que institución estas adscrito. Señal C.

d) En la parte central de la pantalla tenemos una sección con el árbol de materias disponibles, señalados con D; estas se organizan por el área de concentración. Selecciona el área que te interesa. Aparecerán las materias correspondientes.

e) Siempre que hagas clic en alguna materia, aparecerán los detalles en la parte E.

f) Aplicando el botón de Agregar a mi lista, señalada con F, la materia que aparece en E, será agregada a tu lista de materias que solicitarás para inscripción, señaladas con G. Se permite la inscripción de hasta cinco materias.

g) Si quieres quitar de tu lista alguna materia ya seleccionada, oprime el botón del renglón correspondiente, señalados con H.

h) Selecciona quien es tu tutor. Señalado con I.

i) Realiza tu petición para inscripción oprimiendo el botón Inscribir o actualizar, señalado con K.

j) Hecho esto se verifica que halla cupo para los cursos elegidos y se muestran los resultados.

k) Finalmente imprime la solicitud de inscripción, oprimiendo el botón Ver Hoja de Inscripción, señalada con L.

l) El proceso terminará con la impresión de la solicitud. Te resta recabar las firmas del tutor y del coordinador para entregarla en la oficina administrativa del posgrado.

III. Modificación posterior de la Inscripción

Si deseas modificar la solicitud de inscripción:

a) Ingresar nuevamente al sistema.
b) Si es el caso, quita (botones sección H) las materias que ya no desees de tu lista.
c) Si deseas eliminar todas las materias que tiene tu solicitud, oprime el botón de Elimina solicitud, indicado por J.
d) Si es el caso, elige las materias del árbol de materias disponibles (sección D) que deseas agregar (botón F).
e) Actualiza tu solicitud de inscripción (botón K).
f) Visualiza la solicitud, botón L.
g) Imprime tu solicitud, recaba las firmas y entrega a la oficina administrativa del posgrado.

IV. Posibles errores

f) Mi nombre no aparece en la lista de alumnos
El sistema tiene registrados a los alumnos que administrativamente hablando pueden inscribirse, así que si tu nombre no aparece, habrá que preguntar en la oficina administrativa del posgrado tu situación.

g) Mi contraseña no es aceptada
Por omisión el sistema tiene registrado como contraseña tu número de boleta, pero puede suceder que la contraseña sea otro número de identificación, como tu número de examen si eres de nuevo ingreso. En cualquier caso habrá que acudir a la oficina administrativa a fin de conocer la contraseña correcta.

h) No puedo ver la pantalla de acceso al sistema
Refiérase a la liga de la página que pregunta ¿tienes problemas para entrar?. Lo que explica esencialmente es: el sistema requiere un plug-in preinstalado en el visualizador, si no lo tienes habrá que instalarlo.

i) No puedo inscribirme a alguna materia
Para cada curso hay un cupo máximo de inscritos permitidos. Cuando se solicita la inscripción, el sistema verifica que exista capacidad para inscribirse, así que si ya no hay cupo se eliminará la materia de tu lista de elegidas. No hay apartados o preferencias para inscripción. La inscripción queda determinada por los que entren primero.

V. Referencias para problemas no previstos

En caso de tener otro problema no previsto aquí, puedes reportarlo con:

- Oficina administrativa del posgrado (teléfono 5622 3613) Sra. Lourdes González
- Correo electrónico administrador del servidor y la red: Dante Ortiz dortiz@uxmcc2.iimas.unam.mx
- Correo electrónico creador de la versión del sistema: Pablo Castillejo cgpi@uxmcc2.iimas.unam.mx

APÉNDICE E

SEGURIDAD BÁSICA EN EL SERVIDOR HTTP APACHE

El servidor de apache permite establecer mecanismos de seguridad para restringir el acceso a ciertas páginas web. Se explicará el proceso para implementar eso. Se supone sistema operativo Unix o similar[39].

1.- Ejecutar el archivo htpasswsd para agregar la contraseña para el usuario que tendrá acceso a las páginas.

2.- Se da de alta el usuario en el sistema.

3.- Se crea un grupo para agregar a los usuarios que tendrán acceso a las páginas web.

4.- Se crea un directorio donde se pondrán las páginas y demás elementos que desean establecerse con acceso restringido.

5.- Se agrega en el archivo access.conf las siguientes líneas

<Directory nombre del directorio>

AuthName Introduzca nombre y clave

AuthType BASIC

AuthUserFile *directorio con el archivo de los grupos y los miembros del grupo*

AuthGroupFile *directorio con grupos y miembros*

</Directory>

---

[39] Para más información véase [Tackett1999]

Libros

- [Jacobson1999] Jacobson Ivar, Grady Booch, James Rumbaugh ***The Unified Software Development Process***, Addison-Wesley, EU, 1999.
- [Paulk1996] Paulk, Mark, et al. ***The Capability Maturity Model***, Addison-Wesley, EU, 1994.
- [Gheezi1991] Ghezzi Carlo, et al. ***Fundamentals of Software Engineering***, Prentice Hall, EU, 1991.
- [Booch1998] Booch Grady, et al. ***The Unified Modeling Language User Guide***, Addison-Wesley, EU, 1998
- [Freiberg1986] Freiberg Paul, Swaine Michael, ***Microinformatica. Origenes-Personajes, Evolucion y Desarrollo***, Osborne/McGraw-Hill, Madrid, 1986.
- [Booch1994] Booch Grady, ***Object Oriented Análisis and Design with Applications***, Prentice may, EU, 1994.
- [Rumbaugh1991] Rumbauh James, et. al, ***Object-Oriented Modeling and Design***, Prentice Hall, EU, 1991.
- [Jacobson1992] Jacobson Ivar, ***Object-Oriented Software Engineering: A Use Case Driven Approach***, Addison-Wesley, EU, 1992
- [Naughton1997] Naughton, Patrick, et. al, ***Java Manual de Referencia***, Ed Mc Graw Hill, Traducción e impresión España, 1997.
- [Baena1997] Baena, Gullermina, ***Instrumentos de Investigación***, Editores Mexicanos Unidos, México, 1997.
- [Jaworski1998] Jaworski, James, ***Java Unleashed***, Editorial Sams, EU, 1998
- [Tackett1999] Tackett, Jack, et. al, ***Special Edition Using Linux*** 4th Edition, Ed. Que, EU, 1999
- [Koch1994] Koch George, ***Oracle 7 Complete Reference***, Ed. Mc Graw Hill, EU, 1994
- [Silberschatz1997] Silberschatz, Abraham, et. al. ***DataBase System Concepts 3rd ed***, Mc Graw Hill, EU, 1997
- [Tackett1999] Tackett Jack y Steve Burnet, ***Special Edition Using Linux 4th edition***, Editorial Que, EU, 1999.

Revistas:

- [Aguilar1994] Aguilar Cota Manuel, **Ingeniería de Software**, revista Soluciones Avanzadas en Julio, México, 1994.

Sitios en Internet:

La página del sistema:

- [WebSIS] http://www.mcc.unam.mx:8326/inscripcion.html
- [WebIIMAS] http://www.iimas.unam.mx

Sobre el Proceso Unificado

- [WebOMG] http://www.omg.org

Sobre el UML

- [WebUML] http://www.ambysoft.com/umlAndBeyond.html
- [WebUML2] http://www.uml-zone.com
- [WebUML3] http://www.cdt.luth.se
- [WebUML4] http://www.dcs.warwick.ac.uk/people/research/Ananda.Amatya/lnotes/node2.html

Sobre el CMM

- [WebSEI] http://www.sei.cmu.edu
- [WebCMM] http://home.okstate.edu/homepages.nsf/toc/level2.index.html

Sobre Java y Apache

- [WebAPA] http://www.apache.org