UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE INFORMACIÓN INSTITUCIONAL Y DE PERSONAS RELACIONADAS CON LOS ESTUDIOS DE GÉNERO VERSIÓN 3

T E S I S

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA:

JUANA YADIRA LEÓN AMARO

DIRECTOR DE TESIS:
M. I. HONORATO SAAVEDRA HERNÁNDEZ

MÉXICO, D. F. ABRIL DE 2008

A mi Universidad:

La Universidad Nacional Autónoma de México, siempre tendrá mi total agradecimiento y mi amor por haberme albergado en sus aulas al iniciar mis estudios de bachillerato en la Preparatoria #5 "José Vasconcelos" y finalmente concluir mi carrera universitaria en la Facultad de Ingeniería. Estoy orgullosa de ser parte de los egresados de una de las mejores Universidades del Mundo. La UNAM ofrece no sólo la formación académica, sino también la posibilidad de formarnos en diferentes ámbitos (cultural, deportivo, político, social, etc.). Dentro de Ciudad Universitaria se queda parte de mi corazón, en mi mente se queda el recuerdo de sus hermosas instalaciones y las enseñanzas aprendidas que hoy me hacen quien soy.

A mis maestros:
Gracias por todas las enseñanzas en las aulas, porque al compartir su conocimiento dan no sólo conocimiento sino bases para seguir creciendo día a día y formar profesionales comprometidos consigo mismos y el país.

A mis padres:

Gracias, en todo momento pese a las adversidades nunca deje de sentir su apoyo y no sólo económicamente, sino hablo de todos esos días que se desvelaron conmigo, toleraron mi caracter, me dieron sus palabras de aliento, se levantaron muy temprano para preparar mi desayuno . Y además quisiera mencionar que me siento muy orgullosa de ustedes y decirles que los quiero con todo mi corazón.

A mi hermana:

Sabes eres la mejor hermana que puedo tener, gracias por tus palabras que siempre llegan en el momento adecuado y por la hermosa sobrina que me diste. Te quiero mucho no lo olvides.

A mis amigos y amigas:

Se que al mencionarlos me meteré en graves problemas, porque siempre hay alguien que se te olvida, por favor no se enojen si se me olvida alguien, gracias a Isela, Mirna, Pati, Ileana, Katia, Miriam, Diana, Liliana, Lorena, Laurita, Marisol, Pablito, Flor, Ismael, Elsa, Fabiola, Jose Carlos Pandal, Luis Flores, Omar Garzilazo, Levi Omar, Roberto, Sigfrido, Rafael Gómez, Rosio, Sara, Daniel, Josefina, Gabriela Castillo, Luis Arenas, Saul, Armando, Adrian Galicia, Priscilla con cada uno de ustedes compartí diferentes momentos y siempre recibí su apoyo incondicional, una sonrisa, un regaño y sobre todo su sincera amistad, nuevamente gracias.

A Alfredo Arenas González:

Gracias por tus conocimientos, por haberme brindado tu amistad en momentos difíciles y por ser mi amigo.

A mi entrenadora y amiga:

Gracias Geno, por compartir conmigo tus experiencias, tus conocimientos de atletismo, por estar en las competencias, por estar en mis lesiones tanto físicas como emocionales y sobre todo por tu amistad.

A Enrique Villeda:

Gracias por tu cariño, por tu paciencia y sobre todo porque me has sabido escuchar.

A Honorato Saavedra Hernández:

Gracias por confiar en mi para este proyecto, por compartir tus conocimientos conmigo, por todos los días que me dedicaste tu paciencia y tu tiempo en el desarrollo de la tesis.

A la poesía de mis días:

Gracias por todos los hermosos momentos que compartimos juntos, por tu entrega, tu confianza y que la vida te de la felicidad que mereces.

Y por que no puede faltar:

¡GOYA! ¡GOYA!
¡CACHUN, CACHUN, RA, RA!
¡CACHUN, CACHUN, RA, RA!
¡GOYA!
¡¡UNIVERSIDAD!!

Indice

1 Introducción

Actualmente la información se ha convertido en un activo importante para las organizaciones tanto públicas como privadas. Dicha información se obtiene mediante la extracción e interpretación de los datos que posee la organización, algunos de los cuales se encuentran en sus bases de datos.

Una base de datos en su concepto más simple, se refiere a un conjunto de datos relacionados entre sí con un objetivo en común. De acuerdo con (Date, 1986) "es una colección de datos integrados, con redundancia controlada y con una estructura que refleja las interrelaciones y restricciones existentes en el mundo real; los datos que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de dato, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos."

En ocasiones, se encuentra información contenida en diferentes bases de datos y con diferentes formatos, lo que hace que pueda tener errores, o simplemente sea de difícil acceso, por lo que es necesario realizar una migración de la información a un solo formato y en una sola base de datos. Una migración de base de datos es un proceso que se realiza para mover o trasladar los datos almacenados en un origen de datos a otro, para lo cual es indispensable que antes de empezar cualquier proceso de esta naturaleza, se tenga clara y documentada la razón por la cual se está migrando, además de elaborarse la planeación detallada de las actividades contempladas. Existen diversos motivos para hacer una migración tales como: mejorar el desempeño de la base de datos, cumplir con nuevos requerimientos de usuario, o de la aplicación así como la compatibilidad con otras aplicaciones, la actualización de versiones, la reducción de costos que se pueden tener al cambiar por software libre, entre otros escenarios posibles. Las bases de datos son de gran importancia, por lo que se desarrolla software para las empresas e instituciones que ayuda a manejarlas. Este software ayuda a utilizar la información contenida en la base de datos, de una manera fácil y rápida.

El desarrollo de software es esencial en el mundo moderno, ya que donde quiera que uno se encuentre existe la presencia del software, ya sea en el ámbito laboral, como en la educación, la industria, la medicina y muchos más. La disciplina encargada del desarrollo de software es la Ingeniería del software y su principio fundamental es la solución de problemas.

La creciente información hace que en ocasiones, los sistemas tengan que ser modificados para adaptarlos a las nuevas necesidades de la empresa o institución, por lo que es necesario, realizar todas las actualizaciones necesarias, para mejorar la eficiencia del sistema, de igual manera en ocasiones se pueden detectar errores en los sistemas por lo que es necesario que se realice la corrección de los mismos.

Junto con las bases de datos, en la actualidad, para muchas empresas es vital tener un sitio web si quieren tener publicidad y una presencia constante los 365 días del año a cualquier hora. Un sitio web ayuda a una mejor difusión de la información. Las ventajas que se obtienen con un sitio web, son muchas, ya que se puede tener un intercambio de información, no importando el lugar geográfico donde las personas se encuentren ubicadas, si la empresa ofrece algún tipo de servicio o producto, las posibilidades de incrementar su cartera de clientes aumenta de manera que puede tener clientes de cualquier parte del mundo y es más atractivo para los clientes por el impacto que tiene el contacto visual al visitar el sitio web. También es igual de importante que las instituciones y organizaciones que necesiten promover y difundir información específica de las mismas, encuentren la manera de tener actualizada su información, así como dar a conocer a sus integrantes las actividades a realizar a cualquier hora y en cualquier día, por lo que un sitio web es ideal para mantener una comunicación constante.

Desde 1992 el Programa Universitario de Estudios de Género (PUEG) trabaja en la identificación, coordinación, realización y promoción de actividades académicas que consoliden la perspectiva de género. Para cumplir con su objetivo se establecen vínculos con instituciones educativas que realizan actividades de investigación, docencia, extensión y difusión de la perspectiva de género, así como con la sociedad civil, agencias de financiamiento internacional y organismos multilaterales.

El PUEG también participa en tareas de sensibilización y capacitación en diferentes modalidades y niveles educativos, tanto en la zona metropolitana como en los diversos estados de la República Mexicana. En el ámbito internacional desarrolla proyectos de acción orientados a la creación de redes de especialistas en los estudios de género.

Para atender a la problemática planteada por el PUEG, la Facultad de Ingeniería a través del Laboratorio de Multimedia e Internet propone el desarrollo del Sistema de Información Institucional y de Personas Relacionadas con los Estudios de Género (SIIPREG) en su totalidad en ambiente web, identificando e instalando los niveles de seguridad en el servidor y en la base de datos requeridos para las áreas del PUEG y el público en general.

1.1 Justificación

La versión más reciente del SIIPREG (versión 2) está en funcionamiento, se han detectado fallas y se han generado nuevas necesidades. La versión tres del SIIPREG corregirá la versión dos, así como también se implementarán los cambios y mejoras que satisfagan las nuevas necesidades del PUEG. El SIIPREG ha permitido administrar de forma eficaz la información, así como también ha ayudado a vincular y a facilitar la comunicación entre las instituciones y personas involucradas en los estudios de género.

1.2 Alcance

Se mejorará el sistema SIIPREG cubriendo las nuevas necesidades del PUEG y se harán las correcciones de las fallas existentes.

1.3 Relevancia

El SIIPREG es una importante herramienta de trabajo para el PUEG y un instrumento de vinculación, difusión y promoción para quienes se encuentren registrados en el mismo. Los cambios y correcciones a realizar permitirán a los usuarios del sistema mantener su información actualizada de una manera más sencilla y eficiente y el manejo de las pantallas se hará más sencillo.

1.4 Planteamiento del problema

El Programa universitario de estudios de género (PUEG) ha generado una gran cantidad de información sobre personas e instituciones relacionadas con los estudios de género.

La información se encontraba almacenada en distintas bases de datos y archivos en diferentes formatos, en muchos casos estaba repetida, era poco accesible y existían dificultades para realizar búsquedas. El PUEG necesitaba poner cierta información a disposición de varias personas en diferentes localidades, por lo tanto, se requería de un sistema web (SIIPREG) que permitiera integrar y administrar toda la información en una sola base de datos, así como tener acceso a ella desde cualquier punto del país y del mundo.

Debido a esta problemática se creó un sistema de administración informático (SIIPREG) que es capaz de manipular toda la información del PUEG y ésta se encuentra en una única base de datos. El sistema permite consultar, borrar y actualizar la información del PUEG utilizando una clave de acceso desde el sitio web. Para manejar la seguridad del sistema, se habilitaron diferentes claves para la consulta y actualización de la información. El PUEG ha visto el éxito del sistema, ya que al ser consultada la información las búsquedas y consultas son más rápidas y precisas. La información se encuentra organizada en seis diferentes módulos.

Al estar en funcionamiento la versión dos del SIIPREG se han ido encontrando fallas al sistema por lo que el PUEG requiere la corrección de las mismas, así como realizar cambios que resuelvan las nuevas necesidades, para una mejor administración y manipulación de la información que a diario se genera.

1.5 Objetivos

Implementación de cambios y mejoras para la consolidación del sistema de información institucional y de personas relacionadas con los estudios de género (SIIPREG) en su versión tres. En estos cambios se incluyen:

- Corrección de las fallas detectadas en la versión 2.0 del SIIPREG.
- Implementación de los cambios pedidos por el PUEG.

1.6 Organización

El presente trabajo se organiza de la siguiente manera, el capítulo 2 trata de los antecedentes necesarios para su elaboración, como la definición de bases de datos, sus componentes principales, la arquitectura de una base de datos, los tipos de bases de datos y los modelos para las bases de datos. En este capítulo también se habla acerca de la programación orientada a objetos y las ventajas que ofrece. Se habla también de la ingeniería de software, sus características, el proceso para la ingeniería de software, así como algunos modelos de desarrollo. Otro tema que se toca es el Lenguaje Unificado de Modelado y cómo es usado en el desarrollo de software. Por último en este capítulo se habla de la programación web. En el capítulo 3 se trata la definición de lo que es una metodología, en qué nos ayuda para el desarrollo de software, se mencionan algunas metodologías y la metodología seleccionada para el proyecto SIIPREG versión tres. El capítulo 4 describe el desarrollo de las cuatro etapas que conforman la metodología seleccionada (proceso unificado). El capítulo 5 muestra las conclusiones del proyecto.

2 Antecedentes

2.1 Bases de Datos

El concepto de bases de datos surgió a fines de la década de los sesenta del siglo pasado como propuesta de solución a un conjunto de problemas técnicos y administrativos presentados en el manejo de archivos. A medida que los sistemas de información se volvían más complejos y se extendían a nuevas áreas de la operación de una empresa o institución, las dificultades en mantener su funcionamiento y costo bajo control se acentuaban, por lo que surgieron las bases de datos (Mendelzon, 2000).

Antes de las bases de datos existieron los sistemas de procesamiento basados en registros guardados en archivos, creándose programas para su manipulación. Estos tenían varias desventajas: redundancia en la información, inconsistencia en la información, falta de manejo de concurrencia (múltiples usuarios con actualizaciones al mismo tiempo) y la dependencia entre el programa y el archivo.

2.1.1 Definición de base de datos

Algunas definiciones del término base de datos son las siguientes:

1. Un conjunto de información almacenada en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos.
2. Es un conjunto no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina, accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predecible en tiempo.
3. Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso.
4. Es un conjunto de datos gestionados por un SGBD (Sistema de Gestión de Bases de Datos) y asociados a una misma aplicación. (Gardarin, 1990)

5. Es un conjunto de información que es relevante a la operación de una organización, toda la información está apropiadamente organizada y codificada.

2.1.2 Tipos de Bases de Datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación (TBD, 2007) :

Según la variabilidad de los datos almacenados

- Bases de datos estáticas

Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

- Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un video club, etc.

Según el contenido:

- Bases de datos bibliográficas

Solo contienen una representación de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque entonces estaríamos en presencia de una base de datos de texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

- Bases de datos de texto completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

- Directorios

Un ejemplo son las guías telefónicas en formato electrónico. Bancos de imágenes, audio, video, etc.

2.1.3 Ventajas del uso de las Bases de Datos

1. Independencia de datos y tratamiento.- Los cambios en los datos no implican cambios en los programas y viceversa; lo que significa menor costo de mantenimiento.
2. Coherencia de resultados.- Disminuye la redundancia, se evita la inconsistencia.
3. Mejora en la disponibilidad de datos.
4. Cumplimiento de ciertas normas.- Restricciones de seguridad, acceso restringido a los distintos usuarios (VBD, 2007).

2.1.4 Arquitectura de una Base de Datos

La arquitectura de una base de datos, también llamada abstracción de la información se compone de 3 niveles: nivel físico, nivel conceptual y nivel de visión.

1.-Nivel Físico.

Es el nivel real de los datos almacenados. Describe cómo se almacenan los datos (bytes). Este nivel es usado por muy pocas personas que deben estar calificadas para ello, a la persona que se encarga de este nivel, se le asigna el nombre de administrador de la base de datos. Este nivel lleva asociada una representación de los datos, que es lo que se denomina esquema físico.

2.- Nivel Conceptual.

Es el nivel que describe cuáles son los datos reales que están almacenados en la base de datos y qué relaciones existen entre los datos. Este nivel contiene toda la base de datos en términos de unas cuantas estructuras relativamente sencillas, que pueden llegar a ser más complejas. En este nivel intervienen el analista y el desarrollador.

3.- Nivel de Visión.

El nivel de visión trata de cómo los usuarios finales interpretan la información.

Sistema de gestión de bases de datos

Un sistema de gestión de bases de datos, es un sistema cuyo propósito general es registrar y mantener información. Tal información puede estar relacionada con cualquier cosa que sea significativa para la organización donde el sistema opera, es decir, cualquier dato para la toma de decisiones inherentes a la administración de esa organización.

2.1.5 Componentes principales

Un sistema de bases de datos incluye cuatro componentes principales: datos, hardware, software y usuarios (Date, 1986).

Datos.- Son los valores registrados físicamente en la base de datos.

Hardware.- El hardware se compone de los volúmenes de almacenamiento físicos, donde reside la base de datos.

Software.- Son los programas para manejar la base datos, que a menudo recibe el nombre de sistema de gestión de la base de datos. Éste maneja todas las solicitudes de acceso a la base de datos formuladas por los usuarios.

Usuarios.- Existen cuatro clases de usuarios: el analista que es la persona encargada de esquematizar los datos y sus relaciones, el programador/desarrollador es la persona que programa la aplicación con la base de datos, el administrador de la base de datos y los usuarios finales que son los que manipulan e interpretan la información.

2.1.6 El Sistema de Gestión de Bases de Datos

La información necesita ser guardada y manipulada para que sea útil y éste es uno de los objetivos principales que han tenido la computadoras dentro de las empresas e instituciones.

A principios de su incorporación al ámbito empresarial, las computadoras trabajaban con lo que se conoce como "Sistema de procesamiento de archivos" en los que se guardaban los datos para ser procesados por programas escritos especialmente para cada clase de archivo; esto conducía a un sistema monolítico y de difícil mantenimiento.

Un sistema de gestión de bases de datos (en inglés, DBMS, database management system), es un conjunto de datos relacionados entre sí y un grupo de programas para tener acceso a esos datos. También es llamado sistema manejador de bases de datos). El objetivo primordial de un DBMS es crear un ambiente en que sea posible guardar y recuperar la información de la base de datos en forma conveniente y eficiente (Kort, 1987).

El DBMS está formado por una base de datos y el software para manipularlos. Sus funciones son:

- Almacenar, recuperar, eliminar y modificar los datos.
- Guardar la consistencia de los datos, es decir, que la información no sea incorrecta o contradictoria.
- Solucionar problemas de concurrencia.
- Tener seguridad, restringir la información a los usuarios, creando tipos de usuarios.
- Creación y modificación de la base de datos.

El manejo de los datos del DBMS incluye tanto la definición como la manipulación y la seguridad de los mismos a través de tres sistemas DDL, DML y DCL.

DDL.- Un esquema de base de datos se especifica por medio de una serie de definiciones que se expresan en un lenguaje especial llamado lenguaje de definición de datos (en inglés: DDL, Data Definition Language). El resultado de la compilación de las proposiciones en DDL es un conjunto de tablas que se almacenan en un archivo especial llamado diccionario o directorio de datos.

Los DBMS incluyen generalmente un diccionario de datos. Un diccionario de datos es un archivo que contiene metadatos, es decir, datos acerca de los datos . Este archivo se consulta antes de leer o modificar los datos reales en el sistema de base de datos. La información que proporciona el diccionario de datos incluye: nombre de los usuarios, privilegios (permisos) que tienen los usuarios y nombres de los objetos (tablas, vistas, sinónimos, clusters, secuencias, procedimientos, etc.).

La estructura de almacenamiento y los métodos de acceso empleados por el sistema de base de datos se especifican por medio de un conjunto de definiciones de un tipo especial de DDL llamado lenguaje de almacenamiento y definición de los datos. El resultado de la compilación de estas definiciones es una serie de instrucciones que especifican los detalles de implementación de los esquemas de la base de datos que normalmente no pueden ver los usuarios.

DML.- Un lenguaje de manejo de datos (en inglés: DML, data manipulation language) permite a los usuarios manejar o tener acceso a los datos que estén organizados por medio del modelo apropiado. La parte de un DML que implica la recuperación de información se conoce como lenguaje de consultas. Existen básicamente dos tipos de DML.

De procedimientos, necesitan que el usuario especifique qué datos quiere y cómo deben obtenerse.

Sin procedimientos, requieren que el usuario especifique qué datos quiere sin especificar cómo obtenerlos.

DCL.- Es el lenguaje de control encargado de plasmar los privilegios de los usuarios.

Viendo la necesidad de mejorar este estándar se desarrollaron los "Sistemas Gestores de Bases de Datos Relacionales" (SGBDR). Algunas de las características de los SGBDR, son que existe solo una copia de los datos para que todos los programas trabajen con ella, esto es lo que se denomina obtención de redundancia mínima y de esta manera se podrá eliminar la inconsistencia de los datos.

Los SGBDR brindan la capacidad de interactuar en un ambiente cliente/servidor donde los clientes o usuarios (ya sean de una red local o desde Internet) pueden trabajar con un conjunto único de datos alojados en un servidor y donde varios clientes pueden estar trabajando al mismo tiempo. Numerosas empresas se han volcado al desarrollo de sistemas gestores de bases de datos relacionales como Oracle e Informix.

El Administrador de la Base de Datos

Una de las razones principales para contar con sistemas de manejo de base de datos es tener un control centralizado tanto de los datos como de los programas que tienen acceso a ellos. La persona que tiene este control centralizado sobre el sistema es el administrador de la base de datos (en inglés database administrator). Las funciones del administrador de la base de datos son (Kort, 1987):

- Definición del esquema, es decir, la creación del esquema original de la base de datos, esto se logra escribiendo una serie de definiciones que el compilador de DDL traduce a un conjunto de tablas que se almacenan permanentemente en el diccionario de datos. Esta tarea se conoce como diseño de la base de datos.
- Definición de la estructura de almacenamiento y del método de acceso. Esto se lleva a cabo escribiendo una serie de definiciones que posteriormente son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.
- Modificación del esquema y de la organización física, ya sea la modificación del esquema de la base de datos o de la descripción de la organización física del almacenamiento.
- Concesión de autorización para el acceso a los datos, es decir conceder diferentes tipos de permisos para acceso a los datos a los distintos usuarios de la base de datos. Esto permite al administrador de la base de datos regular cuáles son las partes de la base de datos a las que van a tener acceso los diversos usuarios.
- Especificación de las limitantes de integridad.- Estas limitantes se conservan en una estructura especial del sistema que consulta el manejador de la base de datos cada vez que se lleva a cabo una actualización en el sistema.

2.1.7 Modelos de Base de Datos

Un modelo, en general, es una representación simplificada de un sistema real. Si la representación es adecuada, es posible interrogar al modelo para conocer alguna propiedad del sistema real. Una base de datos es un modelo del conjunto de los datos relativos a una organización o al menos a una aplicación en esa organización (Mendelzon, 2000).

Un modelo de datos debe poder representar tanto las características estáticas como las dinámicas del sistema real que se pretende modelar. En consecuencia, un modelo de datos se define por los siguientes componentes:

- Un conjunto de objetos y sus interrelaciones. Esto representa las características estáticas o invariantes e incluye las propiedades de los objetos.
- Un conjunto de operaciones, o lenguaje, que representa las características dinámicas.
- Restricciones sobre los objetos, sus interrelaciones y las operaciones definidas sobre ellos.

El conjunto de objetos y sus interrelaciones resultan de abstracciones realizadas sobre la estructura del sistema real y constituye de este modo la estructura del modelo. Ejemplo de los objetos en los modelos de datos son entidades, tablas, registros, eventos, estados, etc.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos. Se han propuesto varios modelos de datos diferentes, los cuáles pueden dividirse en:

1. Modelos lógicos basados en objetos

Los modelos lógicos basados en objetos se utilizan para describir los datos en los niveles conceptual y de visión. Se caracterizan por el hecho de que permiten una estructuración bastante flexible y hacen posible especificar claramente las limitantes de los datos.

- Modelo E-R.
- Modelo orientado a objetos.

Modelo E- R

El modelo de datos entidad-relación (E-R) se basa en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y de las relaciones entre estos objetos. Una entidad es un objeto que existe y puede distinguirse de otros. La distinción se logra asociando a cada entidad un conjunto de atributos que describen al objeto. (Kort, 1987)

Además de entidades y relaciones, el modelo E-R representa ciertas limitantes que debe cumplir el contenido de una base de datos. Una de estas limitantes importantes es la cardinalidad de mapeo, que expresa el número de entidades con las que pude asociarse otra entidad por medio de un conjunto de relaciones.

La estructura lógica general de una base de datos puede expresarse por medio de un diagrama E R que consta de los siguientes componentes:

- Rectángulos, que representan conjunto de entidades.
- Elipses, que representan atributos.
- Rombos, que representan relaciones entre conjuntos de entidades
- Líneas, que conectan los atributos a los conjuntos de entidades y los conjuntos de entidades a las relaciones.

Modelo orientado a objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones (MOO, 2008).

2. Modelos lógicos basados en registros

Los modelos lógicos basados en registros se utilizan para describir los datos en el nivel conceptual y de visión. A diferencia de los modelos de datos basados en objetos, estos modelos sirven para especificar tanto la estructura lógica general de la base de datos como una descripción en un nivel más alto de la implementación.

- Modelo jerárquico.
- Modelo de red.
- Modelo relacional.

Modelo de Red. Los datos se representan por medio de conjuntos de registros y las relaciones entre los datos se representan con ligas que son consideradas apuntadores. Los registros de las bases de datos se organizan en forma de conjuntos de gráficas arbitrarias.

Modelo Jerárquico. Similar al de red en cuanto a que los datos y relaciones se representan por medio de registros y ligas respectivamente. El modelo jerárquico difiere en que los registros están organizados como un conjunto de árboles.

Modelo relacional. Los datos y las relaciones entre los datos se representan por medio de una serie de tablas, cada una de las cuales tiene varias columnas con nombres únicos.

2.2 Programación estructurada y la orientación a objetos

La programación orientada a objetos desde el punto de vista computacional "es un método de implementación en el cuál los programas son organizados como grupos cooperativos de objetos, cada uno de los cuales representa una instancia de alguna clase, y estas clases, todas son miembros de una jerarquía de clases unidas vía relaciones de herencia" (Greiff, 1994).

Actualmente una de las áreas más utilizadas en la industria y en el ámbito académico es la orientación a objetos. La orientación a objetos mejoró ampliamente el alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software como la falta de portabilidad de código, reutilización de código y ciclos de desarrollo largos.

La programación orientada a objetos (POO) es un modelo de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. La idea central es simple: organizar los programas a imagen y semejanza de la organización de los objetos en el mundo real (Ceballos, 2004).

El desarrollo de la POO empieza a destacar durante la década de los años 80 del siglo pasado tomando en cuenta la programación estructurada, a la que engloba y dotando al programador de nuevos elementos para el análisis y desarrollo de software.

Básicamente la POO permite a los programadores escribir software, de forma que esté organizado en la misma manera que el problema que trata de modelar. Los lenguajes de programación convencionales son poco más que una lista de acciones a realizar sobre un conjunto de datos en una determinada secuencia.

La POO aporta un enfoque nuevo, convirtiendo la estructura de datos en el centro sobre el que pivotan las operaciones. De esta forma, cualquier modificación de la estructura de datos tiene efecto inmediato sobre las acciones a realizar sobre ella, siendo esta una de las diferencias radicales respecto a la programación estructurada.

En la programación estructurada, el diseño descompone los requerimientos del programa paso a paso, hasta llegar a un nivel que permite expresarlos mediante procedimientos y funciones. La POO estructura los datos en objetos que pueden almacenar, manipular y combinar información. La programación estructurada presta atención al conjunto de acciones que manipulan el flujo de datos (desde la situación inicial a la final), mientras que la programación orientada a objetos presta atención a la interrelación que existe entre los datos y las acciones a realizar con ellos.

Ventajas de la Programación orientada a objetos sobre otras formas de programación (VPOO, 2007)

Uniformidad. La representación de los objetos implica tanto el análisis como el diseño y la codificación de los mismos.

Comprensión. Tanto los datos que componen los objetos, como los procedimientos que los manipulan, están agrupados en clases, que se corresponden con las estructuras de información que el programa trata.

Flexibilidad. Al tener relacionados los procedimientos que manipulan los datos con los datos a tratar, cualquier cambio que se realice sobre ellos quedará reflejado automáticamente en cualquier lugar donde estos datos aparezcan.

Estabilidad. Dado que permite un trato diferente para aquellos objetos que permanecen constantes en

el tiempo sobre aquellos que cambian con frecuencia permite aislar las partes del programa que permanecen inalterables en el tiempo.

Reutilización. La noción de objeto permite que programas que traten las mismas estructuras de información reutilicen las definiciones de objetos empleadas en otros programas e incluso los procedimientos que los manipulan. De esta forma, el desarrollo de un programa puede llegar a ser una simple combinación de objetos ya definidos donde estos están relacionados de una manera particular.

2.2.1 Objetos

La capacidad de reconocer objetos físicos es una habilidad que los humanos aprenden en edades muy tempranas. Desde la perspectiva de cognición humana, un objeto es cualquiera de las siguientes cosas (Booch, 1996):

- Una cosa tangible o visible.
- Algo que puede comprenderse intelectualmente.
- Algo a lo que se dirige un pensamiento o acción.

A la definición informal se añade la idea de que un objeto modela alguna parte de la realidad y es, por lo tanto, algo que existe en el tiempo y el espacio. Una definición más refinada sugiere que un objeto representa un elemento, unidad o entidad individual e identificable, ya sea real o abstracta, con un dominio bien definido en el dominio del problema.

En la programación orientada a objetos el elemento fundamental es, como su nombre lo indica, el objeto. Podemos definir un objeto para POO como un conjunto complejo de datos y programas que poseen estructura y forman parte de una aplicación.

Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

2.2.2 Estructura de un objeto

Un objeto está integrado por clases, propiedades o atributos y métodos (Ceballos, 2004).

Clases

Una clase equivale a la generalización de un tipo específico de objetos, pero cada objeto que se construya de una clase tendrá sus propios datos. Un objeto de una determinada clase se crea en el momento que se define una variable de dicha clase. Algunos autores emplean el término instancia, en el sentido de que una instancia es la representación concreta y específica de una clase.

Cuando se utiliza un lenguaje orientado a objetos lo primero es definir las clases de objetos, donde una clase se ve como una plantilla para múltiples objetos con características similares. Las clases permiten la agrupación de objetos que comparten las mismas propiedades y comportamiento. Si bien clase y objeto suelen usarse como sinónimos, no lo son. El esfuerzo del programador ante una aplicación orientada a objetos se centra en la identificación de las clases, sus atributos y operaciones asociadas.

Las propiedades de cada clase deben cumplir una serie de premisas:

- Las propiedades deber ser significativas dentro del entorno de la aplicación, es decir, deben servir para identificar claramente y de una manera única a cada uno de los objetos.
- El número de propiedades de un objeto debe ser el mínimo para realizar todas las operaciones que requiera la aplicación.

Diseño de una clase de objetos

Cuando se escribe un programa orientado a objetos, lo que se hace es diseñar un conjunto de clases, desde las cuáles se crearán los objetos necesarios cuando el programa se ejecute. Cada una de las clases incluye dos partes: los atributos y los métodos. Los atributos definen el estado de cada uno de los objetos de esa clase y los métodos su comportamiento. (Ceballos, 2004)

Normalmente, los atributos, la estructura más interna del objeto, se ocultan a los usuarios del objeto, manteniendo como única conexión con el exterior los mensajes. Esto quiere decir que los atributos de un objeto solamente podrán ser manipulados por los métodos del propio objeto.

Propiedades

Las propiedades distinguen un objeto determinado de los restantes que forman parte de la misma aplicación y tienen valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.

Todo objeto puede tener cierto número de propiedades, cada una de las cuales tendrá, a su vez, uno o varios valores. En POO, las propiedades corresponden a las clásicas "variables" de la programación estructurada. Son, por lo tanto, datos encapsulados dentro del objeto, junto con los métodos (programas). Las propiedades de un objeto pueden tener un valor único o pueden contener un conjunto de valores más o menos estructurados (matrices, vectores, listas, etc.). Además, los valores pueden ser de cualquier tipo (numérico, alfabético, etc.) si el sistema de programación lo permite.

Pero existe una diferencia con las "variables", y es que las propiedades se pueden heredar de unos objetos a otros. En consecuencia, un objeto puede tener una propiedad de maneras diferentes:

-Propiedades propias. Están formadas dentro de la cápsula del objeto.

-Propiedades heredadas. Están definidas en un objeto diferente, antepasado de éste (“padre”, "abuelo", etc.).

Existen tres propiedades fundamentales de los objetos: encapsulamiento, herencia y polimorfismo.

Encapsulación y ocultación de datos

La capacidad de presentación de información dentro de un objeto se divide en dos partes bien diferenciadas:

Interna: La información que necesita el objeto para operar y que es innecesaria para los demás objetos de la aplicación. Estos atributos se denominan privados y tienen como marco de aplicación únicamente a las operaciones asociadas al objeto.

Externa: La que necesitan el resto de los objetos para interactuar con el objeto que se definió.

Estas propiedades se denominan públicas y corresponden a la información que necesitan conocer los restantes objetos de la aplicación respecto del objeto definido para poder operar.

Es posible imaginar la encapsulación como introducir el objeto dentro de una caja negra donde existen dos ranuras denominadas entrada y salida. Si introducimos datos por la entrada automáticamente obtendrá un resultado en la salida. No necesita conocer ningún detalle del funcionamiento interno de la caja.

El término encapsulación indica la capacidad que tienen los objetos de construir una cápsula a su alrededor, ocultando la información que contienen (aquélla que es necesaria para su funcionamiento interno, pero innecesaria para los demás objetos) a las otras clases que componen la aplicación.

Aunque a primera vista la encapsulación puede parecer superflua, se toma en cuenta que existen muchas variables utilizadas de forma temporal: contadores y variables que contienen resultados intermedios, etc. De no ser por la encapsulación estas variables podrían modificarse y podrían interferir en el funcionamiento del resto de los objetos.

La encapsulación no es exclusiva de los lenguajes de programación orientados a objetos. Aparece en los lenguajes basados en procedimientos (PASCAL, C, COBOL, etc.) como una forma de proteger los datos que se manipulan dentro de las funciones.

Los lenguajes de POO incorporan la posibilidad de encapsular también las estructuras de datos que sirven como base a las funciones. Aportan por tanto un nivel superior en cuanto a protección de información. La encapsulación permite el uso de librerías de objetos para el desarrollo de programas. Las librerías son definiciones de objetos de propósito general que se incorporan a los programas. Al ser el objeto parcialmente independiente en su funcionamiento del programa en donde está definido, ya que contiene y define todo lo que necesita para poder funcionar, es fácil utilizarlo en los variados tipos de aplicaciones. Si se asegura, depurando las propiedades y las operaciones dentro de la clase, que el objeto funciona bien dentro de una aplicación, con una correcta encapsulación el objeto podrá funcionar en cualquier otra.

Otra de las ventajas de la encapsulación es que, al definir el objeto como una caja negra con entradas y salidas asociadas, en cualquier momento podemos cambiar el contenido de las operaciones del objeto, de manera que no afecte al funcionamiento general del programa. La encapsulación está en el núcleo de dos grandes pilares de la construcción de sistemas; mantenibilidad y reutilización.

Mantenibilidad. Cualidad que indica que un programa o sistema debe ser fácilmente modificable. Es decir que los cambios en las condiciones externas (como la definición de una nueva variable) implicarán modificaciones pequeñas en el programa / sistema. El concepto de mantenibilidad implica que un programa, al igual que un ser vivo debe ser capaz de adaptarse a un medio ambiente siempre cambiante.

Reutilización. Cualidad que indica qué partes del programa (en este caso objetos) pueden ser reutilizados en la confección de otros programas. Ello implica que los objetos definidos en un programa pueden ser extraídos del mismo e implantados en otro sin tener que realizar modificaciones importantes en el código del objeto. El objetivo final es que el programador construya una librería de objetos que le permita realizar programas basándose en la técnica de cortar y pegar. Ésta extrae (corta) código de otras aplicaciones ya realizadas y las implementa (pega) en la aplicación a realizar, donde, tras algunos retoques, la nueva aplicación estará lista para funcionar. Esta cualidad permite reducir el tiempo de realización, ganando en claridad, mantenibilidad y productividad. La encapsulación de datos se muestra como una herramienta poderosa que nos permite ganar en tiempo de desarrollo y claridad, con el único costo adicional de definir con precisión las entradas y salida de nuestras operaciones.

Herencia

La herencia es una de las características más importantes de la POO porque permite que una clase herede los atributos y métodos de otra clase. La herencia permite crear estructuras jerárquicas de clases donde es posible la creación de subclases que incluyan nuevas propiedades y atributos. Estas subclases admiten la definición de nuevos atributos, así como crear, modificar o inhabilitar propiedades. Con la herencia todas las clases están clasificadas en una jerarquía estricta. Cada clase tiene su superclase (la clase superior en la jerarquía, también llamada clase base), y cada clase puede tener una o más subclases (las clases inferiores en la jerarquía; también llamadas clases derivadas). Las clases que están en la parte inferior en la jerarquía se dice que heredan de las clases que están en la parte superior.

El término heredar significa que las subclases disponen de todos los métodos y propiedades de su superclase. Este mecanismo proporciona una forma rápida y cómoda de extender la funcionalidad de una clase. Cada clase puede tener una superclase (o clase base), lo que se denomina herencia simple, o dos o más superclases, lo que se denomina herencia múltiple.

La herencia es, sin duda alguna, una de las propiedades más importantes de la POO, ya que permite, a través de la definición de una clase básica, ir añadiendo propiedades a medida que sean necesarias y, además, en el subconjunto de objetos que sea preciso.

La herencia permite que los objetos puedan compartir datos y comportamientos a través de las diferentes subclases, sin incurrir en redundancia. Más importante que el ahorro de código, es la claridad que aporta al identificar que las distintas operaciones sobre los objetos son en realidad una misma cosa.

Polimorfismo

Esta propiedad indica la posibilidad de definir varias operaciones con el mismo nombre, diferenciándolas únicamente en los parámetros de entrada. Dependiendo del objeto que se introduzca como parámetro de entrada, se elegirá automáticamente cual de las operaciones se va a realizar.

Una de las ventajas más importantes, es permitir la construcción de las clases que definen un programa de forma totalmente independiente al programa donde se utilizan. Gracias a la encapsulación y el polimorfismo, aunque se utilicen los mismos nombres en las operaciones en dos clases distintas, el programa reconoce a que clase se aplica durante la ejecución.

Métodos

Se puede definir método como un programa procedimental escrito en cualquier lenguaje, que está asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por éste o por sus descendientes.

Un método se escribe en una clase de objetos y determina cómo tiene que actuar el objeto cuando recibe el mensaje vinculado con ese método. A su vez un método también puede enviar mensajes a otros objetos solicitando una acción o información. Los métodos son funciones miembro de la clase, que se ejecutan en respuesta a alguna acción tomada desde dentro de un objeto de esa clase, o bien desde otro objeto de la misma o de otra clase.

Son sinónimos de “método” todos aquellos términos que se han aplicado tradicionalmente a los programas, como procedimiento, función, rutina, etc. Sin embargo, es conveniente utilizar el término “método” para que se distingan claramente las propiedades especiales que adquiere un programa en el entorno POO, que afectan fundamentalmente a la forma de invocarlo (únicamente a través de un mensaje) y a su campo de acción, limitado a un objeto y a sus descendientes, aunque posiblemente no a todos. Si los métodos son programas, se deduce que podrían tener argumentos, o parámetros.

Constructores. Un constructor es un método especial de una clase que es llamado automáticamente siempre que se crea un objeto de esa clase. Su función es iniciar el objeto. (Ceballos, 2004)

2.3 Ingeniería de Software

Introducción

El software es el conjunto de instrucciones que permite al hardware de la computadora desempeñar trabajo útil. En las últimas décadas del siglo XX, las reducciones de costo en hardware llevaron a que el software fuera un componente ubicuo de los dispositivos usados por las sociedades industrializadas (IS, 2007).

El software se ha vuelto esencial para la toma de decisiones comerciales, así como también, ha servido de base para la investigación científica y la resolución de problemas de ingeniería. El software está inmerso en todo tipo de sistemas: transportes, médicos, de telecomunicaciones, militares, procesos industriales, entretenimiento, etc. El software es imprescindible en el mundo moderno.

La Ingeniería de Software, término utilizado por primera vez por Fritz Bauer en la primera conferencia sobre desarrollo de software patrocinada por el Comité de Ciencia de la OTAN celebrada en Garmisch, Alemania, en octubre de 1968, puede definirse según Alan Davis como "la aplicación inteligente de principios probados, técnicas, lenguajes y herramientas para la creación y mantenimiento, dentro de un costo razonable, de software que satisfaga las necesidades de los usuarios" (IIS, 2007).

Se puede decir que un producto de software, en contraposición con el software de uso personal, tiene una gran cantidad de usuarios y muchas veces tiene muchos programadores y personas de mantenimiento. En la mayor parte de los casos son personas distintas quienes desarrollan los productos, los usan y les dan mantenimiento. El desarrollo y mantenimiento de productos de software requiere de un enfoque más sistemático que el necesario en el desarrollo de programas para uso personal.

Para desarrollar productos de software o de programación, las necesidades y limitaciones deben estar determinadas y explícitamente establecidas; el producto debe diseñarse considerando tanto a los usuarios como los principios de operación. Las tareas de mantenimiento de software incluyen solicitudes de análisis de cambios, rediseño y modificación del código fuente, pruebas completas de la versión modificada, actualización de la documentación para mostrar dichos cambios, así como la distribución de la nueva versión en los lugares apropiados.

El quehacer de la ingeniería de software se desarrolla dentro del contexto organizacional de una empresa o institución, por lo que necesita un alto grado de comunicación entre los clientes, administradores, ingenieros en computación y demás técnicos.

La ingeniería de software comparte, junto con las otras ramas de la ingeniería, el enfoque pragmático para el desarrollo y mantenimiento de artefactos tecnológicos; existen, sin embargo, diferencias significativas entre esta ingeniería y las otras, siendo la fuente principal de dichas diferencias la falta de las leyes físicas para la programación, la intangibilidad del producto y lo oculto de las interfaces entre los diversos módulos de programación.

Un principio fundamental de la ingeniería de programación es diseñar productos que minimicen la distancia intelectual entre el problema y la solución, pero la variedad de enfoques en el desarrollo de software está limitada únicamente por la creatividad del ingenio del programador.

La ingeniería de software, como las disciplinas tradicionales de ingeniería, tiene que ver con el costo y la confiabilidad. Algunas aplicaciones de software contienen millones de líneas de código que se espera que se desempeñen bien en condiciones siempre cambiantes.

La ingeniería del software difiere de la programación tradicional en que se utilizan técnicas de ingeniería para especificar, diseñar, instrumentar, validar y mantener los productos dentro del tiempo y presupuesto establecidos para el proyecto.

2.3.1 Definiciones de Ingeniería de Software

- La Ingeniería de software es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación y otros campos (IS, 2007).

- Según la definición del IEEE, citada por (Lewis, 1994) "software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo". Según el mismo autor, "un producto de software es un producto diseñado para un usuario". En este contexto, la Ingeniería de Software es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones eficaces en costo o económicas a los problemas de desarrollo de software", es decir, "permite elaborar consistentemente productos correctos, utilizables y efectivos en costo" (Cota, 1994).

- La ingeniería del software se define como la disciplina tecnológica preocupada de la producción sistemática y mantenimiento de los productos de software que son desarrollados y modificados en tiempo y dentro de un presupuesto definido (Fairley, 1994).

- Los conceptos de las ciencias de la computación y administración de la economía y de la comunicación están combinados dentro del marco de la resolución de problemas; el producto de esto recibe el nombre de ingeniería de software.

- La ingeniería de software es el proceso de diseño y análisis que desglosa una aplicación en hardware y software.

2.3.2 Características fundamentales en la Ingeniería del software

La calidad del producto de software, es decir, la calidad de los programas es una preocupación primordial de los ingenieros en programación o también llamados ingenieros del software, las características importantes dependerán, obviamente, del producto en particular. Existen algunas características fundamentales en todo producto de software; entre ellas están la utilidad, claridad, confiabilidad, eficiencia y economía (Fairley, 1994).

El factor más importante de la calidad de un producto es su utilidad, es decir, que el producto de software satisfaga las necesidades del usuario. La confiabilidad del producto está definida como la "capacidad de un programa para desempeñar una función requerida bajo ciertas condiciones durante un tiempo en específico". Los productos de programación deben estar escritos con claridad y ser fáciles de entender. La clave para realizar un sistema fácil de probar y mantener radica en hacerlo comprensible; los productos de programación que se presentarán a los usuarios deben tener una integridad conceptual y ser claros con el propósito del proyecto. Un producto de programación deberá ser eficiente, pero sólo tanto como la aplicación particular lo amerite. El producto debe ser costeable en su desarrollo, mantenimiento y uso; los esfuerzos en el desarrollo y mantenimiento dedicados al aumento de la eficiencia y la confiabilidad del producto deben ser los apropiados para las aplicaciones del proyecto.

Factores de tamaño

Categorías de acuerdo con el tamaño

El tamaño de un proyecto es un factor importante que determina el nivel de control administrativo y el tipo de herramientas técnicas necesarias en un proyecto de programación. En la tabla 1 se presenta un cuadro de las categorías en el tamaño de un producto.

Factores sobre la calidad y productividad

La calidad del producto y la productividad del programador pueden elevarse al mejorar los procesos necesarios para el desarrollo y mantenimiento de los productos. Algunos de los factores que influyen sobre la calidad y productividad son:

- Capacidad individual
- Complejidad del producto
- Enfoque sistemático
- Confiabilidad requerida
- Entendimiento del problema
- Facilidades y recursos
- Metas adecuadas
- Comunicación del equipo
- Notación adecuada
- Nivel tecnológico
- Tiempo disponible
- Estabilidad de los requerimientos
- Capacitación adecuada
- Expectativas crecientes

Categoría	Número de Programadores	Duración	Tamaño del Producto
Trivial	1	1-4 semanas	500 líneas de código
Pequeño	1	1-6 meses	1K-2K
Mediano	2-5	1-2 años	5K-50K
Grande	5-20	2-3 años	50K-100K
Muy Grande	100-1000	4-5 años	1M
Extremadamente Grande	2000-5000	5-10 años	1M-10M

Tabla 1 "Categorías respecto al tamaño de un producto de software"

2.3.3 Proceso de ingeniería de software

El proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de logra un objetivo, en este caso, la obtención de un producto de software de calidad" (Jacobson,1992). El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Jacobson,1992). El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software.

Pasos del proceso

La ingeniería de software requiere llevar a cabo muchas tareas, sobre todo las siguientes (IS, 2007) :

Análisis de requisitos

Extraer los requisitos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y experiencia en la ingeniería de software para reconocer requisitos incompletos, ambiguos o contradictorios.

Especificación

Es la tarea de describir detalladamente el software a ser escrito, en una forma matemáticamente rigurosa. En la realidad, la mayoría de las buenas especificaciones han sido escritas para entender y afinar aplicaciones que ya estaban desarrolladas. Las especificaciones son más importantes para las interfaces externas, que deben permanecer estables.

Diseño y arquitectura

Se refiere a determinar como funcionará de forma general sin entrar en detalles, consiste en incorporar consideraciones de la implementación tecnológica, como el hardware, la red, etc.

Programación

Dar solución a un problema a través de una serie de pasos ordenados, es decir un algoritmo, el cuál se implementará en algún lenguaje de programación.

Prueba

Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral.

Documentación

Realización del manual de usuario, y posiblemente un manual técnico con el propósito de mantenimiento futuro y ampliaciones al sistema.

Mantenimiento

Mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar mantenimiento. Una pequeña parte de este trabajo consiste en arreglar errores.

2.3.4 Desarrollo de software

La ingeniería de software tiene varios modelos de desarrollo en los cuales se puede apoyar para la realización de software, de los cuales podemos destacar a éstos por ser los más utilizados y los más completos:

- Modelo en cascada (ciclo de vida clásico)
- Modelo en espiral
- Modelo de prototipos
- Modelo en V

2.4 Lenguaje Unificado de Modelado (UML) (UML, 2007)

2.4.1 Introducción a UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" (modelo) del sistema , incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. Se ha convertido en el estándar de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software para crear una notación unificada en la que basar la construcción de sus herramientas CASE(***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, Ingeniería de Software Asistida por Ordenador).

En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas “guerras de métodos” que se mantuvieron a lo largo de los 90's del siglo pasado, en las que los principales métodos creaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros de software que trabajan en el desarrollo orientado a objetos.

Uno de los objetivos principales de la creación de UML fue posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación.

El punto importante a notar es que UML es un "lenguaje" para especificar y no un método o un proceso. UML se usa para definir un sistema de software; para detallar los artefactos en el sistema; para documentar y construir. Es el lenguaje en el que está descrito el modelo. UML se puede usar en una gran variedad de formas para soportar una metodología de desarrollo de software pero no especifica en sí mismo qué metodología o proceso usar.

2.4.2 Modelos o Diagramas

Un modelo o diagrama representa a un sistema de software desde una perspectiva específica. Al igual que la planta y el alzado de una figura en dibujo técnico nos muestran la misma figura vista desde distintos ángulos, cada modelo nos permite fijarnos en un aspecto distinto del sistema.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas como los siguientes:

- Diagramas de Estructura Estática.
- Diagramas de Casos de Uso.
- Diagramas de Secuencia.

- Diagramas de Colaboración.
- Diagramas de Estados.

2.4.3 Diagramas de Estructura Estática

Los diagramas de estructura estática de UML se van a utilizar para representar tanto modelos conceptuales como diagramas de clases de diseño. Ambos usos son distintos conceptualmente, mientras los primeros modelan elementos del dominio, los segundos presentan los elementos de la solución de software. Ambos tipos de diagramas comparten una parte de la notación para los elementos que los forman (clases y objetos) y las relaciones que existen entre los mismos (asociaciones). Sin embargo, hay otros elementos de notación que serán exclusivos de uno u otro tipo de diagrama.

Clases

Una clase se representa mediante una caja subdividida en tres partes: En la parte superior se muestra el nombre de la clase, en la media los atributos y en la inferior las operaciones. Una clase puede representarse de forma esquemática, con los atributos y operaciones suprimidos, siendo entonces tan solo un rectángulo con el nombre de la clase. En la figura 2.1 se ve cómo una misma clase puede representarse a distinto nivel de detalle según interese y según la fase en la que se esté.

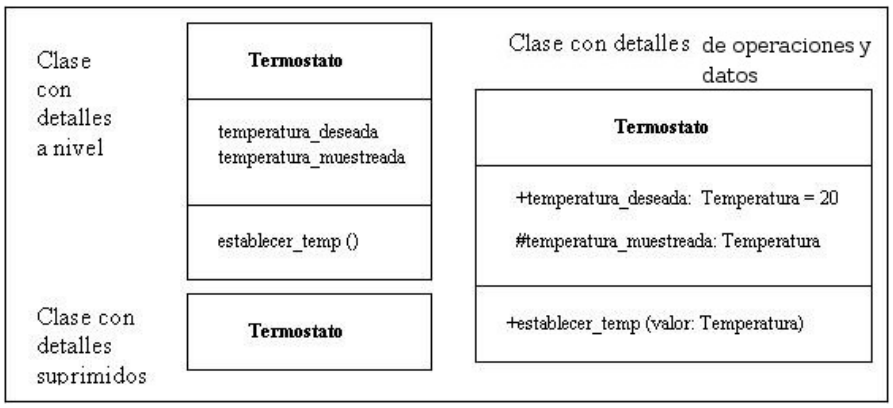

Figura 2.1 Representación de una clase a distintos niveles.

Objetos

Un objeto se representa de la misma forma que una clase. En el compartimento superior aparecen el nombre del objeto junto con el nombre de la clase subrayados, según la siguiente sintaxis: `nombre_del_objeto:nombre_de_la_clase` Puede representarse un objeto sin un nombre específico, entonces sólo aparece el nombre de la clase. Esto se puede observar en la figura 2.2.

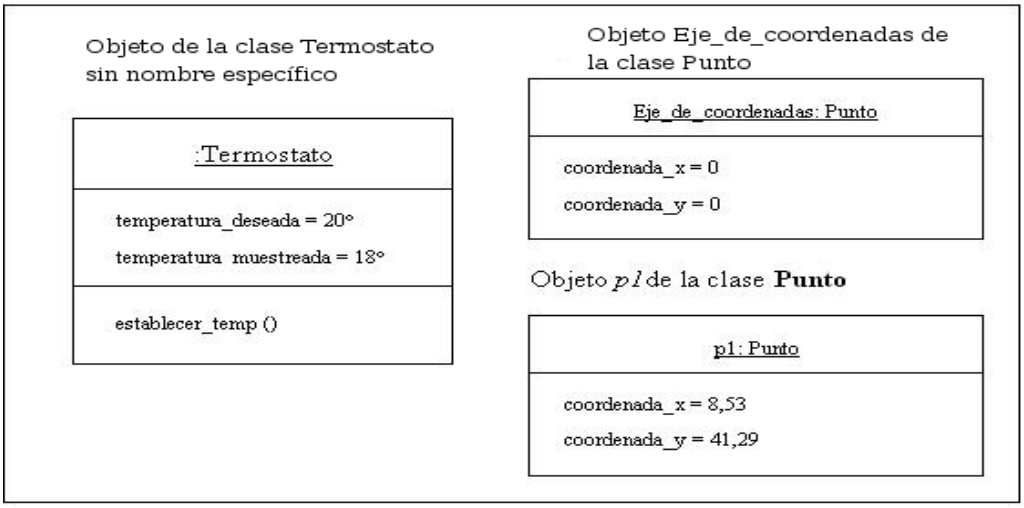

Figura 2.2 Diferentes representaciones de un objeto.

Asociaciones

Las asociaciones entre dos clases se representan mediante una línea que las une. La línea puede tener una serie de elementos gráficos que expresan características particulares de la asociación. A continuación se verán los más importantes de entre dichos elementos gráficos.

Nombre de la asociación y dirección

El nombre de la asociación es opcional y se muestra como un texto que está próximo a la línea. Se puede añadir un pequeño triángulo negro sólido que indique la dirección en la cual leer el nombre de la asociación. En el ejemplo de la figura 2.3 se puede leer la asociación como Director manda sobre Empleado .

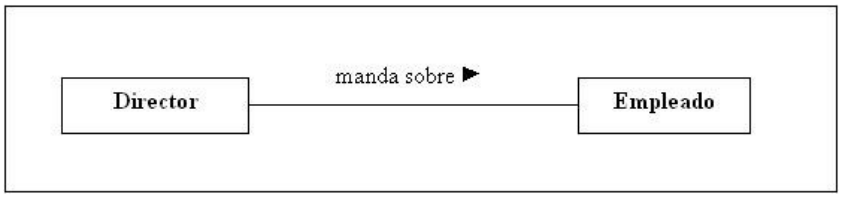

Figura 2.3 Ejemplo de una asociación.

Los nombres de las asociaciones normalmente se incluyen en los modelos para aumentar la legibilidad. Sin embargo, en ocasiones pueden hacer demasiado abundante la información que se presenta, con el consiguiente riesgo de saturación. En ese caso se puede suprimir el nombre de las asociaciones consideradas como suficientemente conocidas. En las asociaciones de tipo agregación y de herencia no se suele poner el nombre.

Multiplicidad

La multiplicidad es una restricción que se pone a una asociación, que limita el número de instancias de una clase que pueden tener esa asociación con una instancia de la otra clase.

Puede expresarse de las siguientes formas:

" Con un número fijo: 1

" Con un intervalo de valores: 2..5

" Con un rango en el cual uno de los extremos es un asterisco. Significa que es un intervalo abierto. Por ejemplo, 2..* significa 2 o más.

" Con una combinación de elementos como los anteriores separados por comas: 1, 3..5, 7, 15..*.
" Con un asterisco: * . En este caso indica que puede tomar cualquier valor (cero o más). Un ejemplo de multiplicidad lo vemos en la figura 2.4.

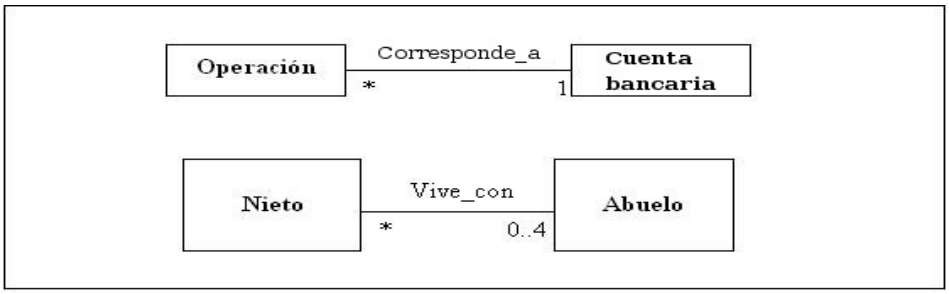

Figura 2.4 Multiplicidad.

Roles

Para indicar el papel que juega una clase en una asociación se puede especificar un nombre de rol. Se representa en el extremo de la asociación junto a la clase que desempeña dicho rol. Esto se representa en la figura 2.5.

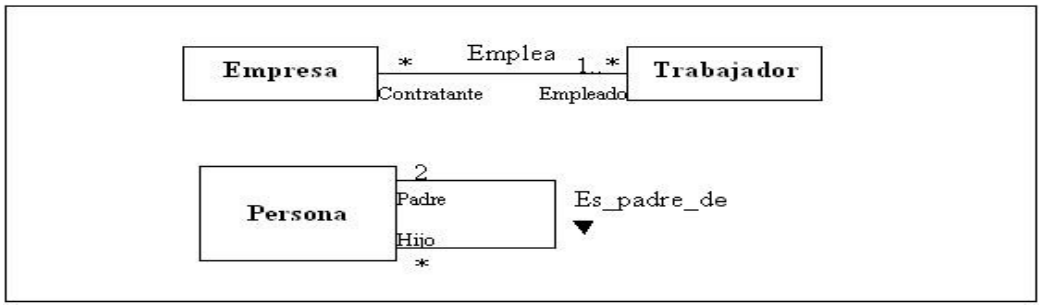

Figura 2.5 Roles.

Agregación

El símbolo de agregación es un diamante colocado en el extremo en el que está la clase que representa el todo . Esto se presenta en la figura 2.6.

Clases Asociación

Cuando una asociación tiene propiedades propias se representa como una clase unida a la línea de la asociación por medio de una línea a trazos. Tanto la línea como el rectángulo de clase representan el mismo elemento conceptual: la asociación. Por tanto ambos tienen el mismo nombre, el de la asociación.

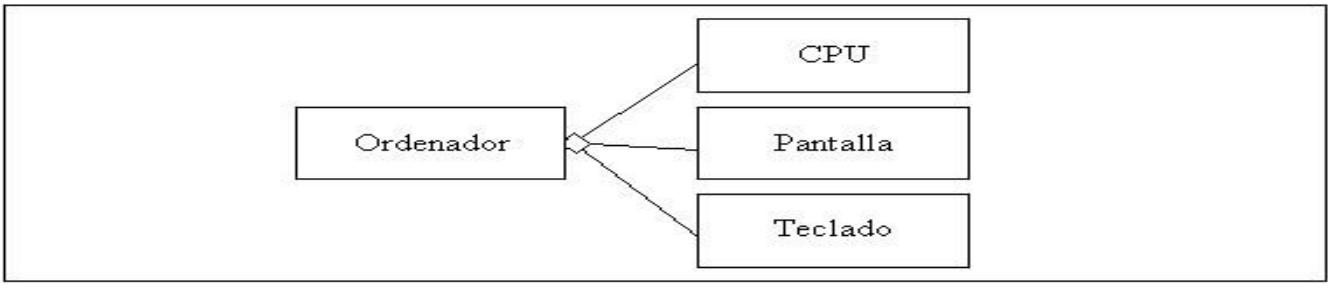

Figura 2.6: Agregación.

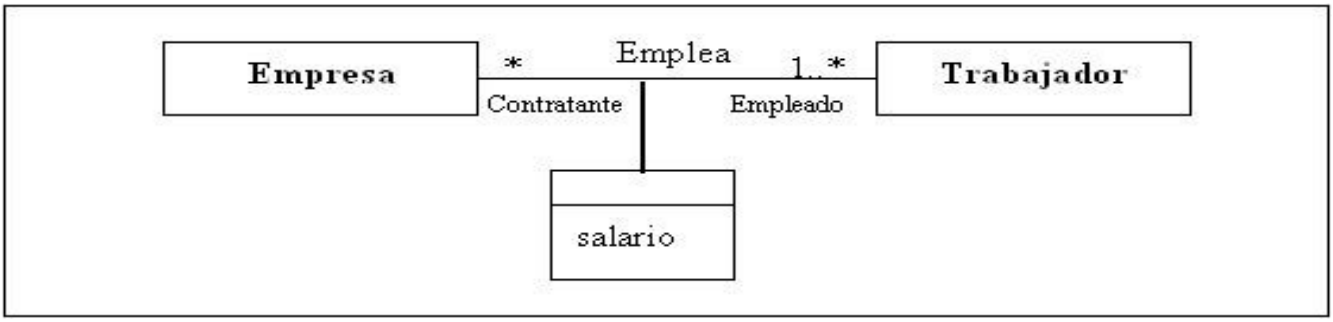

Figura 2.7 Clases Asociación.

Cuando la clase asociación sólo tiene atributos el nombre suele ponerse sobre la línea (como ocurre en el ejemplo de la figura 2.7). Por el contrario, cuando la clase asociación tiene alguna operación o asociación propia, entonces se pone el nombre en la clase asociación y se puede quitar de la línea.

Asociaciones N-Arias

En el caso de una asociación en la que participan más de dos clases, las clases se unen con una línea a un diamante central. Si se muestra multiplicidad en un rol, representa el número potencial de tuplas de instancias en la asociación cuando el resto de los N-1 valores están fijos. En la figura 2.8 se ha impuesto la restricción de que un jugador no puede jugar en dos equipos distintos a lo largo de una temporada, porque la multiplicidad de "Equipo" es 1 en la asociación ternaria.

Navegabilidad

En un extremo de una asociación se puede indicar la navegabilidad mediante una flecha. Significa que es posible "navegar" desde el objeto de la clase origen hasta el objeto de la clase destino. Se trata de un concepto de diseño, que indica que un objeto de la clase origen conoce al (los) objeto(s) de la clase destino, y por tanto puede llamar a alguna de sus operaciones.

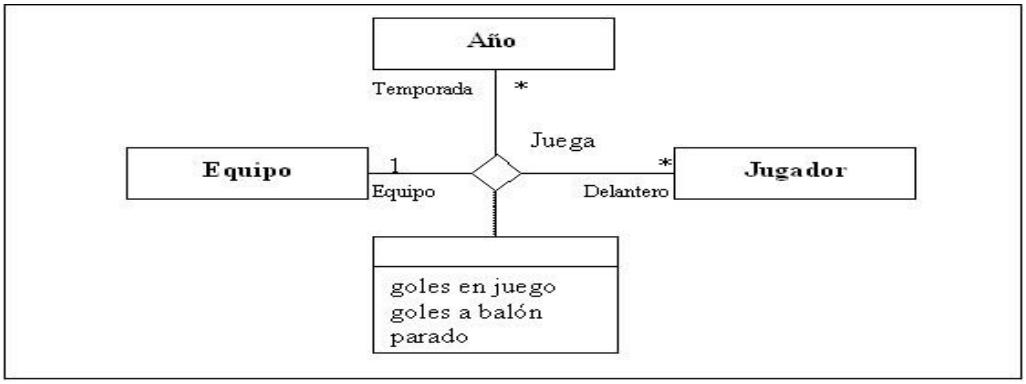

Figura 2.8 Asociación con más de una clase.

Herencia

La relación de herencia se representa mediante un triángulo en el extremo de la relación que corresponde a la clase más general o clase "padre". Si se tiene una relación de herencia con varias clases subordinadas, pero en un diagrama concreto no se quieren poner todas, esto se representa mediante puntos suspensivos. En el ejemplo de la Figura 2.9, sólo aparecen en el diagrama 3 tipos de departamentos, pero con los puntos suspensivos se indica que en el modelo completo (el formado por todos los diagramas) la clase "Departamento" tiene subclases adicionales, como podrían ser "Recursos Humanos" y "Producción".

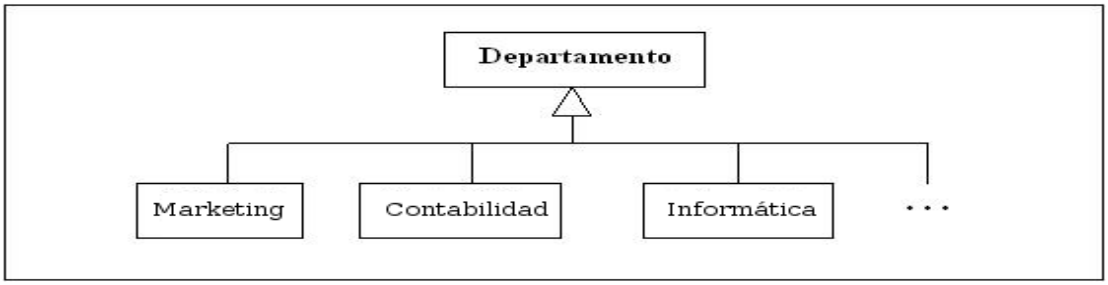

Figura 2.9 Herencia.

Elementos Derivados

Un elemento derivado es aquel cuyo valor se puede calcular a partir de otros elementos presentes en el modelo, pero que se incluye en el modelo por motivos de claridad o como decisión de diseño. Se representa con una barra "/" precediendo al nombre del elemento derivado. Como se muestra en la figura 2.10.

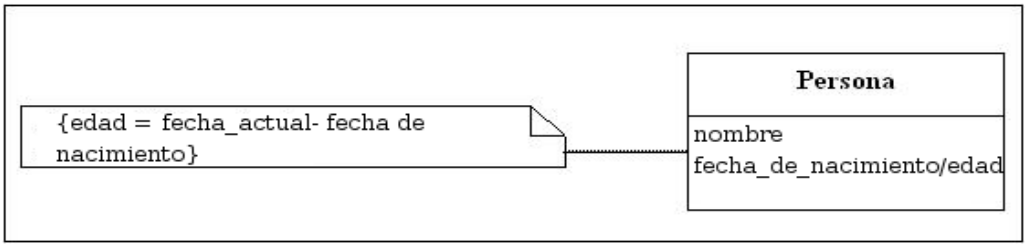

Figura 2.10 Elemento derivado.

2.4.4 Diagramas de Casos de Uso

Un diagrama de casos de uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea. En la figura 2.11 se muestra un ejemplo de diagrama de casos de uso para un cajero automático.

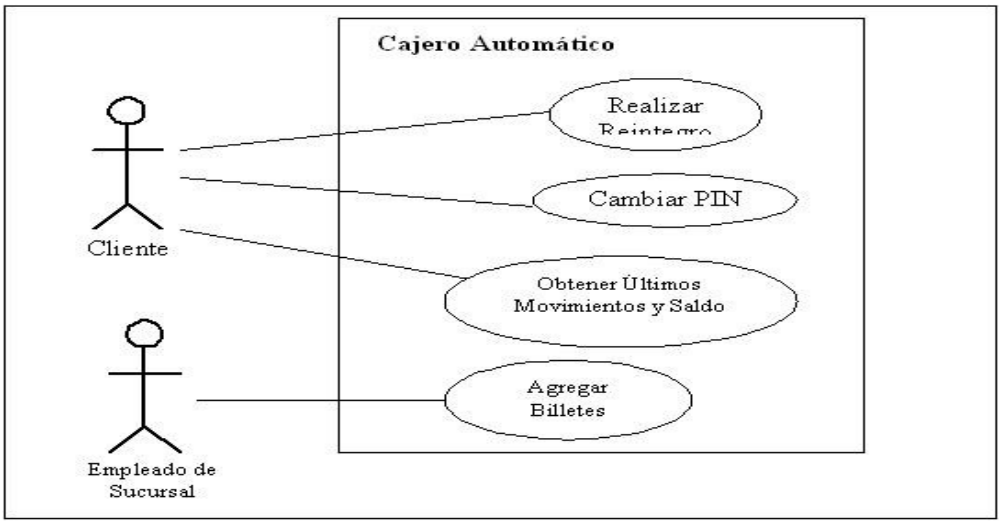

Figura 2.11 Diagrama de un caso de uso.

Elementos

Los elementos que pueden aparecer en un diagrama de casos de uso son: actores, casos de uso y relaciones entre casos de uso.

Actores

Un actor es algo con comportamiento, como una persona (identificada por un rol), un sistema informatizado u organización, y que realiza algún tipo de interacción con el sistema. Se representa mediante una figura humana. Esta representación sirve tanto para actores que son personas como para otro tipo de actores.

Casos de uso

Un caso de uso es una descripción de la secuencia de las interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el diagrama de casos de uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre casos de uso

Un caso de uso, en principio, debería describir una tarea que tiene un sentido completo para el usuario. Sin embargo, hay ocasiones en las que es útil describir una interacción con un alcance menor como caso de uso. La razón para utilizar estos casos de uso no completos en algunas ocasiones, es mejorar la comunicación en el equipo de desarrollo. Las relaciones entre casos de uso pueden ser de los siguientes tres tipos:

• Incluye: Un caso de uso base incorpora explícitamente a otro caso de uso en un lugar especificado en dicho caso base. Se suele utilizar para encapsular un comportamiento parcial común a varios casos de uso. En la figura 2.12 se muestra cómo el caso de uso "Realizar reintegro" puede incluir el comportamiento del caso de uso Autorización.

• Extiende: Cuando un caso de uso base tiene ciertos puntos (puntos de extensión) en los cuales, dependiendo de ciertos criterios, se va a realizar una interacción adicional. El caso de uso que extiende describe un comportamiento opcional del sistema (a diferencia de la relación incluye que se da siempre que se realiza la interacción descrita). En la figura 2.13 se muestra como el caso de uso "Comprar producto" permite explícitamente extensiones en el siguiente punto de extensión: "info regalo". La interacción correspondiente a establecer los detalles sobre un producto que se envía como regalo están descritos en el caso de uso "Detalles regalo".

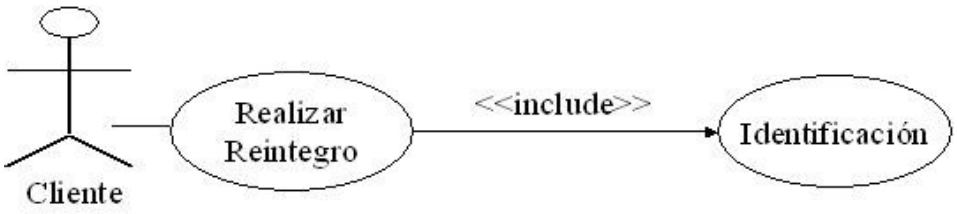

Figura 2.12 Caso de uso realizar reintegro.

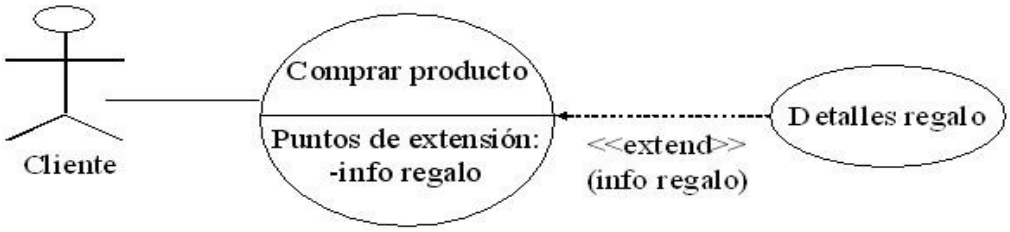

Figura 2.13 Caso de uso comprar producto.

Ambos tipos de relación se representan como una dependencia etiquetada con el estereotipo correspondiente, de tal forma que la flecha indique el sentido en el que debe leerse la etiqueta. Junto a la etiqueta se puede detallar el/los puntos de extensión del caso de uso base en los que se aplica la extensión.

• Generalización: Cuando un caso de uso definido de forma abstracta se particulariza por medio de otro caso de uso más específico. Se representa por una línea continua entre los dos casos de uso, con el triángulo que simboliza generalización en UML (usado también para denotar la herencia entre clases) pegado al extremo del caso de uso más general. Al igual que en la herencia entre clases, el caso de uso hijo hereda las asociaciones y características del caso de uso padre. El caso de uso padre se trata de un caso de uso abstracto, que no está definido completamente. Este tipo de relación se utiliza mucho menos que las dos anteriores.

2.4.5 Diagramas de Secuencia

Diagramas de Interacción: En los diagramas de interacción se muestra un patrón de interacción entre objetos. Hay dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular: diagramas de secuencia y diagramas de colaboración.

Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba hacia abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

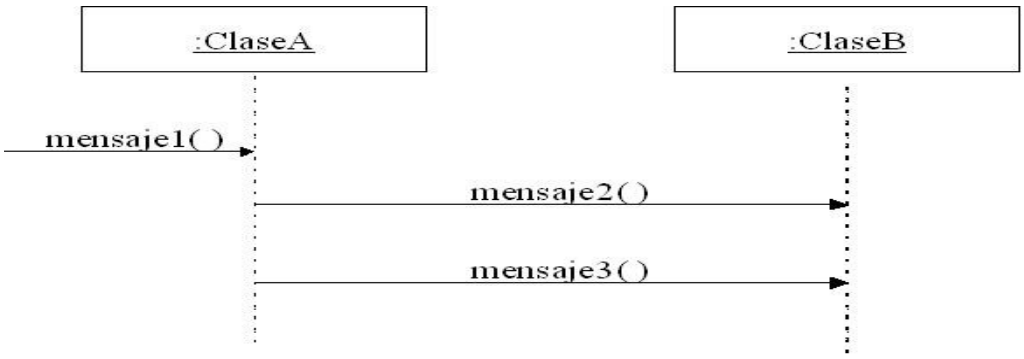

Figura 2.14 Diagrama de secuencia.

2.4.6 Diagramas de Colaboración

Un diagrama de colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia. Un ejemplo de diagrama de colaboración lo podemos ver en la figura 2.15.

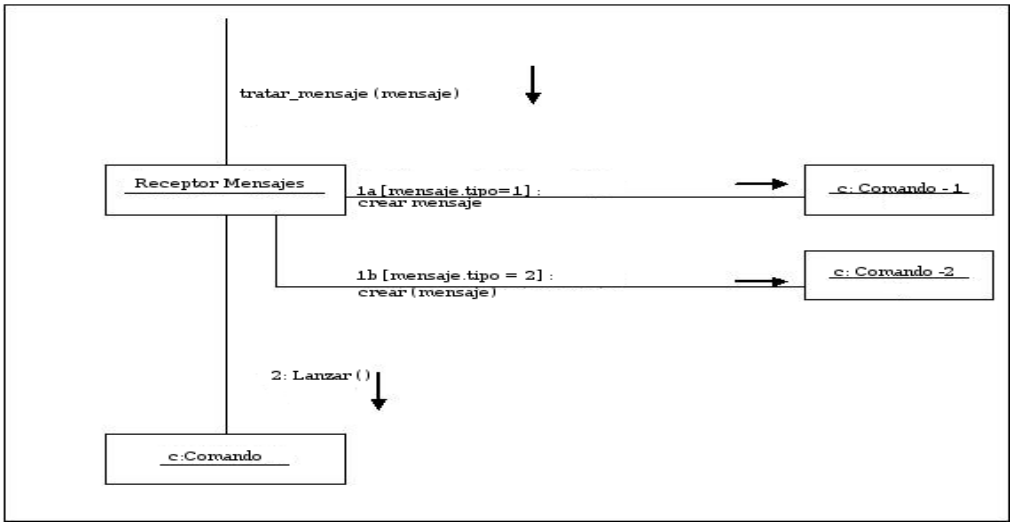

Figura 2.15 Diagrama de colaboración.

En cuanto a la representación, un diagrama de colaboración muestra a una serie de objetos con los enlaces entre los mismos, y con los mensajes que se intercambian dichos objetos. Los mensajes son flechas que van junto al enlace por el que "circulan", y con el nombre del mensaje y los parámetros (si los tiene) entre paréntesis. Cada mensaje lleva un número de secuencia que denota cuál es el mensaje que le precede, excepto el mensaje que inicia el diagrama, que no lleva número de secuencia. Se pueden indicar alternativas con condiciones entre corchetes (por ejemplo `3 [condición_de_test]:nombre_de_método()`). También se puede mostrar el anidamiento de mensajes con números de secuencia como 2.1, que significa que el mensaje con número de secuencia 2 no acaba de ejecutarse hasta que no se han ejecutado todos los 2.x .

2.4.7 Diagramas de Estados

Un diagrama de estados muestra la secuencia de estados por los que pasa bien un caso de uso, bien un objeto a lo largo de su vida, o bien todo el sistema. En él se indican qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera.
En cuanto a la representación, un diagrama de estados es un grafo cuyos nodos son estados y cuyos arcos dirigidos son transiciones etiquetadas con los nombres de los eventos.
Un estado se representa como una caja redondeada con el nombre del estado en su interior. Una transición se representa como una flecha desde el estado origen al estado destino.
La caja de un estado puede tener 1 o 2 compartimentos. En el primer compartimento aparece el nombre del estado. El segundo compartimento es opcional, y en él pueden aparecer acciones de entrada, de salida y acciones internas.

Una acción de entrada aparece en la forma entrada/acción_asociada donde acción_asociada es el nombre de la acción que se realiza al entrar en ese estado. Cada vez que se entra al estado por medio de una transición la acción de entrada se ejecuta.
Una acción de salida aparece en la forma salida/acción_asociada. Cada vez que se sale del estado por una transición de salida la acción de salida se ejecuta.
Una acción interna es una acción que se ejecuta cuando se recibe un determinado evento en ese estado, pero que no causa una transición a otro estado. Se indica en la forma nombre_de_evento/acción_asociada.

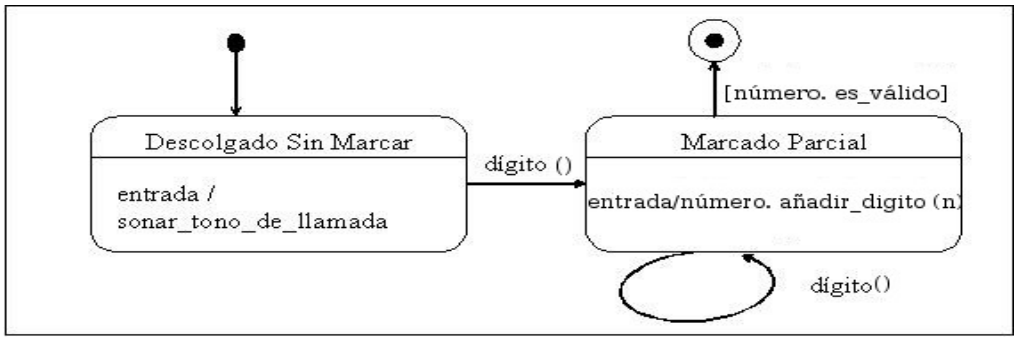

Figura 2.16 Diagrama de estado.

Un diagrama de estados puede representar ciclos continuos o bien una vida finita, en la que hay un estado inicial de creación y un estado final de destrucción (finalización del caso de uso o destrucción del objeto). El estado inicial se muestra como un círculo sólido y el estado final como un círculo sólido rodeado de otro círculo. En realidad, los estados inicial y final son pseudoestados, pues un objeto no puede "estar" en esos estados, pero nos sirven para saber cuáles son las transiciones iniciales y finales.

2.4.8 Estandarización de UML

Además de haberse convertido en un estándar de facto, UML es un estándar industrial promovido por el grupo OMG al mismo nivel que el estándar CORBA (Common Object Request Broker Architecture — arquitectura común de intermediarios en peticiones a objetos) para intercambio de objetos distribuidos. Para la revisión de UML se formaron dos "corrientes" que promovían la aparición de la nueva versión desde distintos puntos de vista. Finalmente se impuso la visión más industrial frente a la académica. Recientemente se ha publicado la versión 2.0 en la que aparecen muchas novedades y cambios que, fundamentalmente, se centran en resolver carencias prácticas. Además, esta versión recibe diversas mejoras que provienen del lenguaje SDL.

2.5 Programación web

2.5.1 Los inicios de la web

La world wide web e Internet han introducido a la población en general en el mundo de la informática (Pressman, 5). Ya que es posible comprar fondos de inversión colectivos y acciones, descargar música, video, obtener asesoramiento médico, comprar y reservar boletos para eventos culturales, deportivos y vender artículos. También es posible conocer gente, tomar cursos, consultar información de Universidades, Institutos, es decir, en el mundo virtual se puede hacer casi todo lo que se necesite. El Internet y la web son los avances más importantes en la historia de la informática.

Tim Berners Lee junto con su equipo en el CERN (Organización Europea para la Investigación Nuclear (CERN, 2008)), el Laboratorio Europeo de Física de Partículas en Génova, Suiza, desarrolló las bases del world wide web a principios de 1990 (Millar,1997). Al principio se creó como un medio para presentar documentos de hipertexto y compartir investigaciones en redes basadas en TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet). Los documentos se mostraban en una herramienta modo texto. HTML (Lenguaje de Marcado de Hipertexto (LEMAY, 1998)) fue el lenguaje de descripción de una página utilizado para presentar información y crear enlaces de hipertexto y HTTP (Protocolo de Transferencia de Hipertexto) fue el protocolo de comunicación utilizado para conectar las máquinas al world wide web. (Millar,1997)

En febrero de 1993, el Centro Nacional de Aplicaciones de Supercomputación (NCSA) desarrolló una herramienta de navegación gratuita y gráfica para sistemas Unix denominada Mosaic fue la primera herramienta de navegación web que incorporó algunos de los elementos estándar del interfaz gráfico de usuario (GUI) dentro de una herramienta de navegación.

En 1993, Mosaic se importó a sistemas Macintosh y Windows y a finales de ese año había cientos de servidores web. Las posibilidades de uso multimedia de las herramientas de navegación gráficas, como Mosaic, fue una de las razones del por qué la web se hizo tan popular. En 1994 los desarrolladores de Mosaic abandonaron NCSA y crearon su propia empresa, que más tarde se convirtió en Netscape Communications Corporation. En 1995; la herramienta web de Netscape, denominada Navigator, se convirtió en la herramienta más popular de la web.

En 1996, Microsoft entró en el mercado de las herramientas de la navegación con Internet Explorer. Otro desarrollo importante en 1995 y 1996 fue la posibilidad de incorporar los applets Java en las herramientas de navegación web y la adopción del lenguaje de programación Java. Los applets Java son pequeños programas escritos en el lenguaje de programación Java que se pueden incorporar en páginas web.

3 Metodología

3.1 ¿Qué es una metodología?

Es importante que el desarrollo de software esté basado en una metodología que sea capaz de incrementar su productividad y calidad. Una metodología es una secuencia de pasos para lograr una serie de tareas. Las tareas se realizan por lo general en el mismo orden todas las veces. Una metodología es un conjunto ordenado de tareas que involucran actividades, restricciones y recursos que producen una determinada salida esperada. Una metodología involucra un conjunto de herramientas y técnicas. Cualquier metodología tiene las siguientes características:

- La metodología establece todas las actividades principales.
- La metodología utiliza recursos, está sujeta a una serie de restricciones (tal como un calendario) y genera productos intermedios y finales.
- La metodología puede estar compuesta por subprocesos que se encadenan de alguna manera.
- La metodología puede definirse como una jerarquía de procesos organizada de tal modo que cada subproceso tenga su propio modelo de proceso.
- Cada actividad de la metodología tiene criterios de entrada y de salida, de modo que se conoce cuando inicia y cuando termina una actividad.
- Las actividades se organizan en una secuencia, de modo que resulta claro cuando una actividad se realiza en orden relativo a otras actividades.
- Todo metodología tiene un conjunto de principios orientadores que explican las metas de cada actividad.
- Las restricciones o controles pueden ser de aplicación a una actividad, recurso o producto.

Las metodologías son importantes porque imponen consistencia y estructura sobre un conjunto de actividades.

3.2 Tipos de Metodología

3.2.1 Modelo en cascada

El modelo clásico del proceso de desarrollo de software es el modelo en cascada, llamado así porque sus etapas se representan cayendo en cascada, desde una etapa hacia la siguiente. Este modelo es una secuencia de actividades (o etapas) que consiste en el análisis de requerimientos, el diseño, la implementación, la integración y las pruebas. Dentro del modelo en cascada una etapa de desarrollo debe completarse antes de dar comienzo a la siguiente. De esta forma, cuando todos los requerimientos del cliente han sido identificados, analizados para comprobar su integridad y consistencia y registrados en un documento de requerimientos, recién entonces el equipo de desarrollo puede seguir con las actividades de diseño del sistema. El modelo en cascada presenta una visión muy clara de cómo suceden las etapas durante el desarrollo y sugiere a los desarrolladores cuál es la secuencia de eventos que podrían encontrar.

El modelo en cascada puede ser muy útil, ayudando a los desarrolladores a diagramar lo que necesitan hacer. Su simplicidad hace que sea fácil explicarlo a los clientes que no están familiarizados con el desarrollo de software; explicita los productos intermedios que son necesarios a fin de poder comenzar la siguiente etapa de desarrollo.

El análisis de requerimientos consiste en reunir las necesidades del producto y casi siempre su salida es texto. El diseño describe la estructura interna del producto y suele representarse con diagramas y texto. Implementación significa programación. El producto de esta etapa es el código a cualquier nivel, incluido el producido por sistemas de generación automática de código, lenguajes de cuarta generación u otros. La integración es el proceso de ensamblar las partes para completar el producto.

3.2.2 Modelo del proceso en espiral

El proceso en espiral reconoce la necesidad de pasar por la secuencia de análisis de requerimientos, diseño, implementación y pruebas más de una vez. Existen varias razones, una de las más importantes es eliminar los riesgos, otra es construir una versión preliminar del producto que se pueda mostrar al cliente para obtener una retroalimentación; una razón más es evitar la integración de una base de código grande todo a la vez, como lo pide el modelo del proceso en cascada. La idea es construir cada versión sobre el resultado de la anterior, este proceso repetitivo forma una especie de trayectoria en espiral.

El proceso en espiral se ajusta al avance de los proyectos típicos; sin embargo, requiere de una administración más cuidadosa que el proceso en cascada, esto se debe al hecho de que la documentación debe ser consistente siempre que el proceso termina una iteración completa. Además con el propósito de optimizar la productividad del equipo, con frecuencia es necesario comenzar una nueva iteración antes de que la anterior haya terminado.

3.2.3 Programación extrema

La Programación Extrema (XP) (del inglés Extreme Programming) es uno de los llamados procesos o metodologías ágiles de desarrollo de software. Fue creado por Kent Beck, Ward Cunningham y Ron Jeffries. Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, equipo pequeño y plazo de entrega urgente. Consiste en un conjunto de prácticas fundamentadas en un conjunto de valores.

La programación extrema aparece para equipos pequeños que necesitan desarrollar software rápidamente, en un ambiente donde los requisitos particulares del cliente y de las circunstancias pueden cambiar de forma imprevista.

La programación extrema describe cuatro actividades básicas para el proceso de desarrollo de software: codificación, pruebas, escuchar y diseño. Las prácticas de XP están agrupadas en cuatro áreas: retroalimentación a escala fina, proceso continuo, entendimiento compartido y bienestar del programador. Los valores que actualmente son utilizados por XP son: comunicación, simplicidad, retroalimentación, coraje y respeto.

Características de XP.

La metodología se basa en:

• Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, haciendo pruebas de las fallas que pudieran ocurrir.

• Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, haciendo la metodología más flexible al cambio.

• Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

3.3 Proceso Unificado de Rational (RUP – Rational Unified Process)

El Proceso Unificado de Rational es un proceso de ingeniería de software de aproximación iterativa el cuál está bien definido y estructurado. Este proceso puede ser utilizado por equipos de trabajo grandes o pequeños.

RUP es utilizado por una amplia variedad de compañías en diferentes sectores de la industria. Los principios esenciales de RUP, son los siguientes:

- Mitigar los errores y riesgos de manera temprana y continua.
- Asegurarse del valor y calidad del producto que se le entregará al cliente.
- Permanecer atentos en la ejecución del software.
- Incorporar los cambios tempranamente en el proyecto.
- Trabajar todos como un solo equipo.
- Establecer una arquitectura ejecutable al iniciar el proyecto.
- Adoptar la calidad como una forma de vida y no como un cambio.

RUP y el desarrollo iterativo

RUP utiliza una aproximación por iteraciones, esto es, una secuencia de iteraciones. Cada iteración incluye algunas o la mayoría de las disciplinas (requisitos, análisis, diseño, implementación). Cada iteración tiene bien establecido un conjunto de objetivos y produce un trabajo parcial de la implementación del sistema final. Cada iteración sucesiva es consecuencia o es construida a través de las anteriores iteraciones para desarrollar y refinar el sistema hasta que el sistema final esté completo. Las iteraciones iniciales tienen énfasis en los requisitos, análisis y diseño. Las iteraciones posteriores tienen mayor énfasis en la implementación y en la pruebas.

RUP tiene dos estructuras o dimensiones, la estructura dinámica y la estructura estática. La estructura dinámica representa la dimensión del tiempo del proceso, la cual muestra al proceso en términos de ciclos, fases, iteraciones e hitos. La estructura estática describe los elementos del proceso tales como las actividades, disciplinas, roles. La estructura dinámica se ocupa del ciclo de vida o tiempo de un proyecto, RUP divide al proyecto en cuatro fases o etapas: Inicio, Elaboración, Construcción y Transición. Cada fase contiene una o más iteraciones, esto lo determinará el hito de cada fase, es decir el cumplimiento de los objetivos en cada fase.

3.4 Fases de RUP

3.4.1 Fase de Inicio

Dentro de la fase de inicio el propósito es comprender el sistema que se va a construir, establecer los requisitos y el alcance del sistema. Mitigar algunos de los riesgos que existan, así como reunir la información necesaria para saber cómo se va a desarrollar el proyecto o cómo no se debe desarrollar.

La mayoría de los proyectos sólo realizan una iteración en esta fase, pero en algunos casos se requiere más de una iteración para alcanzar los objetivos, entre algunas de las razones para realizar más de una iteración se encuentran:

- El sistema es muy grande y no es fácil determinar el alcance del sistema.
- No hay precedente del sistema.
- Existen varios interesados en el sistema y la relación entre ellos es compleja, lo que complica los objetivos a los que se quiere llegar.

3.4.1.1 Los cinco objetivos básicos en la fase de Inicio

1. Entender el sistema que se va a construir. Determinar la visión, el alcance del sistema, sus límites, identificar a quién va dirigido el sistema y qué es lo que se espera de él. Si existe más de un cliente, todos deben estar de acuerdo en los objetivos del sistema.

En lo referente al equipo de trabajo que realizará el proyecto, es importante que los líderes de proyecto, desarrolladores, analistas y todos los involucrados tengan en mente los mismos objetivos.

Para alcanzar este objetivo es necesario:

a) Estar de acuerdo en la visión del proyecto. Establecer los beneficios que nos proporciona la construcción del proyecto, los problemas que éste resolverá, quienes son los usuarios principales y las características de alto nivel del proyecto.

b) Describir de manera breve que va hacer el sistema. Se deben identificar y describir los actores típicos del sistema y clasificarlos en base a lo que hacen dentro del sistema y con qué recursos cuentan.

c) Identificar y describir brevemente los casos de uso. Identificar y describir cómo cada actor interactuará con el sistema. Si el actor es un humano describir las diferentes maneras en que usará el sistema. La descripción de estas interacciones típicas con el sistema es a lo que se le llama casos de uso.

2. Identificar la clave de la funcionalidad del sistema. Identificar los casos de uso más críticos para dedicar el mayor tiempo posible a estos.

a) Usualmente el arquitecto del sistema es el encargado de identificar los casos de usos más críticos y analizar las estrategias de desarrollo, los riesgos de funcionamiento y las estrategias de la seguridad de datos.

b) Es importante determinar la funcionalidad del sistema desde la perspectiva del usuario, ya que ésta cubre un área de la arquitectura que no es cubierta por ningún caso de uso crítico, también es importante entender cada área del sistema para asegurarse de tratar todos los riesgos técnicos.

Es importante identificar todos los casos de uso críticos y tener una descripción detallada de los mismos.

3. Determinar al menos una posible solución. Proponer al menos una arquitectura para la realización del sistema. Es necesario asegurarse que exista una arquitectura con la cuál se pueda construir el sistema, donde el riesgo y y el costo para construirlo sean mínimos. Como ejemplo se pueden considerar 3 opciones para la arquitectura, está se elegirá en base a la funcionalidad deseada, la compatibilidad con otras aplicaciones, los requerimientos en las aplicaciones y el mantenimiento.

Se sugiere que se realicen las siguientes preguntas para ver qué opción es la más viable.

- ¿Qué otros sistemas similares se han construido, y que tecnología y arquitectura se utilizó en ellos? ¿Cuál fue su costo?
- En particular para una tecnología en evolución existente, ¿la actual arquitectura es satisfactoria o necesita ser desarrollada?
- ¿Qué tecnologías se tendrían que utilizar en el sistema?, ¿Se necesita adquirir una nueva tecnología?, ¿Cuáles son los riesgos y los costos asociados con ésto?

- ¿Qué software es necesario para el desarrollo del sistema?, ¿Puede ser adquirido?, ¿Se puede utilizar el software utilizado en otro proyecto?, ¿Cuáles son los costos estimados?

En algunos casos es necesario poner en ejecución algunos de los elementos clave de las arquitecturas posibles, para el mejor entendimiento de los riesgos y opciones que se tienen.

4. Identificar los costos, el tiempo requerido y los riesgos asociados al proyecto.

Entender qué se va construir es esencial, pero determinar cómo se va construir y el costo que éste implica también es crucial. Para determinar si se continúa con la construcción de un proyecto, es necesario saber cuánto costará. La mayoría de los costos están relacionados con los recursos que se necesitan, así como el tiempo que llevará terminar el proyecto.

Con la combinación de estos conocimientos, la comprensión de la funcionalidad requerida y la valoración de los usuarios, es posible construir un documento de modelo de negocio para el proyecto. El documento de modelo de negocio indica el valor económico del proyecto, expresándolo en términos cuantitativos. El documento de modelo de negocio es el instrumento para conseguir el financiamiento adecuado para el proyecto. En algunos casos ya se ha fijado un presupuesto para el proyecto, por lo que se determina lo que se puede entregar y el tiempo.

5. Decidir que proceso se va a seguir y las herramientas a utilizar.

Es importante que el equipo de trabajo comparta la misma idea de cómo se desarrollará el proyecto, es decir cuál es el proceso a seguir. El proceso a seguir debe evitar gastos innecesarios, así como satisfacer las necesidades específicas del mismo. Las decisiones se pueden ir tomando conforme avanza el proyecto, pero para proyectos grandes es necesario tomar un tiempo para la elección del proceso. Una vez que se eligió el proceso, se escogen las herramientas a utilizar.

3.4.1.2 Hito del proyecto en la fase de Inicio

Al finalizar la fase de la inicio, se realiza un hito (revisión de los objetivos de la fase) para examinar los objetivos del ciclo de vida del proyecto para saber si los objetivos han sido alcanzados. La revisión de los objetivos, considera los siguientes criterios de evaluación:

- El interesado en el sistema está de acuerdo con el costo y el tiempo estimado.
- Estar de acuerdo en los requisitos que se han identificado y la manera en que se han entendido.
- Estar de acuerdo con los riesgos iniciales que se han identificado, así como con la manera de mitigarlos.

El proyecto puede ser abortado o reconsiderado si no se han cumplido los objetivos.

3.4.2 Fase de Elaboración

La elaboración es la segunda fase del ciclo de vida de RUP. Las ventajas principales en las iteraciones dentro de esta fase son: atender los riesgos importantes, construcción de un esqueleto o bosquejo de la arquitectura del sistema, refinar y desarrollar los planes del proyecto que fueron producidos en la fase de inicio. Estos planes seguirán revisándose a lo largo del desarrollo del proyecto.

La fase de elaboración más que describir las actividades, se preocupa por los objetivos que se quieren lograr y dará una guía de cómo alcanzarlos. Esto ayuda a centrarse en las actividades esenciales.

La meta de esta fase es definir y establecer la arquitectura del sistema. Los objetivos de la elaboración son cuatro y cada uno se enfoca a los riesgos asociados con los requisitos, a la arquitectura, a los costos, al tiempo y finalmente los riesgos relacionados con el proceso y las herramientas de desarrollo. Al asegurar la correcta atención a los riesgos antes mencionados, se facilita la fase de construcción minimizando los riesgos en ésta.

3.4.2.1 Objetivos de la fase de Elaboración

1. Obtener una comprensión más detallada de los requerimientos.

Durante la elaboración, se desea tener una comprensión extensa de la mayoría de los requisitos, puesto que la mayor parte se describen brevemente en la fase de inicio. Esto permitirá crear un plan más detallado. Finalmente se desea obtener una comprensión profunda de los requisitos más críticos para validar que la arquitectura ha cubierto todas las bases, algo que solo se puede alcanzar a través de la implementación parcial de estos requerimientos.

Al finalizar la fase de inicio, se debe haber producido un buen documento de visión, así como una descripción detallada del veinte por ciento de los casos de uso o la mayoría de los esenciales. También se debe tener una breve descripción de los casos de uso restantes.

Para el final de la elaboración, es preciso terminar la descripción de la mayoría de casos de uso. Algunos casos del uso pueden ser muy simples o similares a otros, pero están relacionados con otros datos, que se pueden posponer hasta la fase de construcción. Se debe crear un prototipo de la interfaz de usuario para los casos de uso importantes, esto ayuda a que los clientes comprendan la importancia de los casos de uso. Se recomienda probar cada caso de uso con un usuario, usando el prototipo de la interfaz para aclarar cuál será la experiencia y de esta manera obtener información que sirva de retroalimentación. Al detallar los casos de uso es probable encontrar nuevos casos de uso, los cuáles se agregan después. Es importante actualizar constantemente el glosario. Al final de la elaboración, se habrá detallado la mayor parte de los requisitos (probablemente cerca del 80 por ciento).

2. Diseño, implementación, validación y establecimiento de la arquitectura.

Se desea diseñar, implementar y probar el esqueleto de una estructura del sistema. La funcionalidad en el nivel de uso no será completa, pero como la mayoría de las interfaces entre los bloques de construcción se ponen en ejecución durante la elaboración, se puede compilar y probar la arquitectura. Se pueden tomar decisiones críticas del diseño, incluyendo la selección de tecnologías, componentes principales, y sus interfaces. Para cumplir con este objetivo es importante tomar en cuenta los bloques de construcción más importantes del sistema, sus interfaces, para tomar decisiones de construir, comprar o reutilizar algunos bloques de construcción, es importante hacer una descripción de cómo estos bloques de construcción influyen en el tiempo. En RUP para validar una arquitectura es necesario tener una arquitectura ejecutable que pueda ser probada para verificar que atienda las necesidades y que constituya la base apropiada sobre la cual se pueda implementar el resto del sistema.

Ejecutar una arquitectura es poner en práctica parcialmente el sistema construido para demostrar que el diseño arquitectónico podrá apoyar la funcionalidad dominante y, más importante, para exhibir las características adecuadas en términos de funcionamiento, rendimiento de procesamiento, capacidad, confiabilidad, escalabilidad y otros. Establecer una arquitectura ejecutable permite que la capacidad funcional completa del sistema sea construida sobre una base sólida durante la fase de la construcción, sin el miedo de fractura. La arquitectura ejecutable se construye como prototipo evolutivo, con una alta probabilidad de que satisfaga los requerimientos del sistema.

Al finalizar la fase de la elaboración, se debe establecer la arquitectura, lo que significa que se puede tomar la arquitectura como una referencia estable para construir el resto del sistema. Al llegar a este punto, es posible modificar la arquitectura sólo si se tiene una buena razón. Esta base proporciona una cierta estabilidad para el equipo de desarrollo. Mientras más grande sea el equipo de trabajo y más compleja la arquitectura, no se puede tener la libertad de realizar cambios.

Al finalizar la etapa de inicio se produjo una arquitectura o por lo menos se identificó una que permitiría que se construyera el sistema con una cantidad razonable de riesgo y un costo razonable. También se tiene una idea de los riesgos que se enfrentarán, especialmente en las áreas de la adquisición de la tecnología y elementos reutilizables, tales como el marco arquitectónico y el software empaquetado.

Al iniciar la etapa de elaboración se debe tener bastante claro qué sistema se va a construir, así como la arquitectura a utilizar, si se ha utilizado con anterioridad la arquitectura propuesta, se debe aprovechar para avanzar, si no es así se deben identificar los bloques arquitectónicos principales, es decir los subsistemas y los componentes principales.

Para cada subsistema o componente identificado, es necesario describir sus capacidades dominantes, para conocer sus interfaces con el resto del sistema. Un subsistema corresponde a un componente o colección de componentes. Se deben utilizar los casos de uso arquitectónicos significativos para manipular la arquitectura. En la etapa de inicio se identificaron quizá el 20 o 30 por ciento de los casos de uso que son críticos para el sistema, éstos probablemente son importantes para la manipulación de la arquitectura.

Es necesario identificar ciertos elementos en los requerimientos, que posiblemente no sean funcionales e incluso sean desconocidos y riesgosos y después identificar los casos de uso que los representan. La implementación de estos casos de uso nos lleva a confrontar y resolver los riesgos.

Finalmente se deben identificar otros casos de uso, que aunque no sean críticos o no impliquen cambios técnicos, ayuden a tener una mejor visión de partes del sistema que aún no han sido cubiertas, de modo que al final de la elaboración, se tenga la arquitectura deseada.

Diseño de los casos de uso críticos

La representación del diseño de un caso de uso se llama realización del caso de uso. Se puede dividir este trabajo en una sección de análisis y una sección de diseño. A continuación una descripción de los cinco pasos más esenciales al producir una realización de un caso de uso.

1. Realizar un análisis preliminar de los objetos involucrados en la colaboración.
2. Especificar la responsabilidad de las clases en cada caso de uso.
3. Detallar el análisis de las clases. Esto ayuda a entender la responsabilidad del análisis de cada clase. Revisar el modelo de análisis para asegurarse de que dos clases no se traslapen y que su relación tiene sentido.

4. Diseñar los casos de uso. Especificar en qué orden y cómo cada clase diseñada se relaciona con las otras para proporcionar partes importantes de la arquitectura usadas en la funcionalidad del caso de uso. De esta manera repartir la funcionalidad de un caso de uso en un sistema de elementos donde se comunican el uno con el otro, puede ser usado para un sistema orientado a objetos o no.
5. Refinar las clases de análisis en clases de diseño. Detallar cada clase de diseño especificando las operaciones y atributos, revisar el modelo del diseño para asegurarse que no exista duplicidad de funcionalidad entre varias clases y que las relaciones entre ellas tengan sentido.

Consolidación y empaquetadura de las clases identificadas

El siguiente paso es agrupar las clases identificadas en subsistemas apropiados. El equipo arquitectónico habrá identificado ya algunos de los subsistemas. Algunas pautas que se siguen para empaquetar las clases son:

- Localizar el impacto de los futuros cambios agrupando las clases que sea posible cambiar en el mismo subsistema.
- Forzar a que se cumplan ciertas reglas de visibilidad. No se desea empaquetar clases de diferentes capas en el mismo subsistema.

Asegurarse de que la arquitectura cubra los casos de uso críticos

Un objetivo importante en la construcción de la arquitectura ejecutable es asegurarse de que incluye los casos de uso de todas las áreas importantes del sistema. Una vez que se haya consolidado el empaquetado y se hayan detallado los casos de uso, se necesita confirmar que todas las áreas del sistema estén cubiertas. Si, al final de la elaboración se descubren áreas “no examinadas”, se deben identificar escenarios adicionales para asegurar la cobertura de la arquitectura. Esto es parte de la estrategia de mitigación de riesgos para reducir al mínimo cambios inesperados más adelante.

Identificar los mecanismos arquitectónicos

Los mecanismos arquitectónicos representan soluciones concretas comunes a los problemas encontrados con frecuencia. Son patrones arquitectónicos, que proporcionan soluciones estándar a los problemas tales como recolección de basura, almacenaje persistente, comunicación con los sistemas externos con cierto protocolo, etc.

3. Mitigar los riesgos esenciales, determinación del tiempo de término y estimación de los costos de manera más exacta.

Durante la elaboración se tratan los riesgos importantes. La mayoría serán tratados como resultado de detallar los requerimientos, diseñar, implementar, y de probar la arquitectura en ejecución. También se refina y detalla el plan para el proyecto .

Planeación del proyecto y estimación de los costos.

Hacia el final de la elaboración, se debe tener información más exacta que permita actualizar la valoración del proyecto y los costos.

- Se debieron detallar los requerimientos para así entender qué sistema se va a construir.
- Se debió implementar la estructura de un esqueleto (arquitectura ejecutable) del sistema, lo que significa que se han solucionado la mayoría de los problemas más difíciles.
- Se debieron mitigar la mayoría de los riesgos, lo que ayuda a tener una mejor visión de la estimación del tiempo y el costo del proyecto
- En este momento se conoce la efectividad del equipo de trabajo, las herramientas y la tecnología actual porque se han utilizado los tres para llevar a cabo al menos una vez un ciclo de vida en la elaboración.

4. Refinar el desarrollo.

Refinar el proceso que se definió para la fase de inicio refleja las lecciones aprendidas. También continúa la implementación de las herramientas de desarrollo de software que se necesitan para el proyecto.

3.4.2.2 La elaboración y las iteraciones

En la fase de la elaboración, muchos riesgos son mitigados produciendo una arquitectura ejecutable, es decir, un subconjunto de los aspectos más esenciales del sistema que permiten que se demuestren las capacidades dominantes y por lo tanto se puede afirmar que los riesgos están eliminados. Si se ha construido un sistema anteriormente con la misma tecnología que se está utilizando en el actual proyecto, es probable que se logre alcanzar el objetivo de esta fase en una sola iteración, porque los riesgos serían mínimos. Es posible utilizar soluciones anteriores y de esta manera el progreso se agiliza. Pero si no se cuenta con la experiencia, el sistema es muy complejo o si se está utilizando una nueva tecnología, se pueden necesitar dos o tres iteraciones para conseguir la arquitectura óptima y mitigar los riesgos dominantes.

Primera iteración en la elaboración

- Diseñar, implementar, y realizar una prueba, de una pequeña cantidad de escenarios críticos que sirve para identificar qué tipo de arquitectura se necesita.
- Identificar, implementar, y realizar una pequeña prueba del sistema inicial de los mecanismos arquitectónicos.
- Realizar un diseño lógico preliminar de la base de datos.
- Detallar el flujo de acontecimientos complicados de la mitad de los casos de uso que se propone detallar en la elaboración, la prioridad debe ser en orden decreciente.
- Realizar las suficientes pruebas para asegurarse que los riesgos arquitectónicos están mitigados.

Segunda iteración en la elaboración

- Corregir los errores de la iteración anterior.
- Diseñar, implementar, y realizar la prueba de los panoramas arquitectónicos restantes. Es importante que en esta iteración se asegure la arquitectura.
- Identificar, implementar y probar los mecanismos arquitectónicos restantes.
- Diseñar e implementar una versión preliminar de la base de datos.
- Detallar la segunda mitad de los casos de uso que necesitan ser especificados en la elaboración.

- Probar, validar y refinar la arquitectura de manera que sea estable. Es posible realizar modificaciones a la arquitectura, pero lo más recomendable es que no se haga, excepto para resolver problemas críticos. Si no se puede alcanzar un buen grado de estabilidad en la arquitectura, se debe agregar otra iteración a la elaboración. Es probable que se retrase el proyecto, pero los costos son mínimos.

3.4.2.3 Hito del proyecto en la fase de elaboración

Al final de la fase de elaboración en el hito, se examinan detalladamente los objetivos del sistema y el alcance, la arquitectura elegida y la solución de la mayoría de los riesgos. Si el proyecto no cumple con lo antes mencionado, es necesario reconsiderarlo. La revisión del hito del ciclo de vida en la arquitectura considera los siguientes criterios de evaluación:

- ¿La visión y los requerimientos del proyecto son estables?
- ¿La arquitectura es estable?
- ¿La prueba y la evaluación de prototipos ejecutables han demostrado riesgos y se han resuelto?
- ¿Los planes de la iteración para la construcción son lo suficientemente detallados para permitir proceder con el trabajo?
- ¿Los planes de la iteración para la fase de construcción están basados en estimaciones creíbles?
- ¿Todos los clientes están de acuerdo en la visión actual, según lo definido en el documento de visión?
- ¿Son los gastos reales contra los gastos previstos aceptables?

3.4.3 Fase de Construcción

La elaboración se finaliza con el lanzamiento interno de la arquitectura ejecutable, que permitió tratar los riesgos técnicos más importantes, implementando y validando código en ejecución real. Durante la fase de construcción, hay que centrarse en refinar los requerimientos, el diseño, la implementación y hacer las pruebas.

En la fase de la construcción es donde más tiempo se consume. Aunque durante la elaboración se trataron la mayor parte de los riesgos, en la construcción aparecen nuevos riesgos que se tienen que resolver.

3.4.3.1 Objetivos de la fase de Construcción

La fase de construcción trata sobre el desarrollo eficiente en costo de un sistema completo, es decir, una versión operacional del sistema. Los objetivos de la etapa de construcción son los siguientes:

1. Reducir los costos de desarrollo y alcanzar un cierto grado de paralelismo en el trabajo de los equipos de desarrollo. Optimizar los recursos y evitar trabajar dos veces en una misma cosa.

2. Organizar en base a la arquitectura. Una de las muchas ventajas de una arquitectura robusta es que divide claramente las responsabilidades del sistema en subsistemas bien definidos, y los individuos pueden centrarse sobre todo en sus subsistemas asignados. Los desarrolladores necesitan entender en su totalidad el sistema, pero pueden centrarse sobre todo en un subsistema asignado a ellos. La organización en base a la arquitectura también ayuda con la comunicación. La comunicación cara a cara es normalmente la forma más eficaz de comunicación, pero cuando los proyectos crecen, el método cara a cara deja de ser eficiente. La comunicación se puede mejorar, teniendo un equipo responsable de la arquitectura y varios equipos de los subsistemas con un responsable cada uno.

3. Asegurarse del progreso continuo. Se necesitan establecer metas a corto plazo y demostrar continuamente que se han alcanzado. Las pautas para asegurar el éxito son las siguientes:

- Crear un equipo con una misión. Es decir se debe tener equipos multidisciplinarios, donde cada miembro del equipo se sienta responsable de la aplicación que realiza y del progreso que el equipo va obteniendo.
- Fijar claramente las metas realizables para los desarrolladores. Cada desarrollador debe tener muy claro los objetivos a lograr en una iteración dada.
- Demostrar continuamente y probar el código. Medir el progreso, revisando, compilando y probando el código.

4. Desarrollar de manera iterativa un sistema completo que esté listo para el usuario. Desarrollar la primera versión operacional del sistema (lanzamiento beta) describiendo los casos de uso restantes y otros requerimientos, completando los detalles del diseño, completando la implementación y probando el software. Determinar si el software, los sitios y los usuarios están listos para que la aplicación pueda ser utilizada.

3.4.3.2 La construcción y sus iteraciones

El número de iteraciones que se requiere para la fase de construcción varía dependiendo del proyecto, pero en la mayoría de los casos la fase de construcción necesita más iteraciones que cualquier otra fase. La planeación de las iteraciones es conducida en su mayoría por los casos de uso que deben ser implementados. Se desean implementar los casos de uso que son los más esenciales para los clientes, así como también los que están asociados con los riesgos técnicos. Especialmente en la primera iteración y tal vez en la segunda iteración, se debe comenzar con la implementación parcial de los casos de uso, para terminar rápidamente con tantos riesgos como sea posible, así como obtener una implementación razonable antes de detallar los casos de uso. Una vez que se ha decidido cuales casos de uso implementar por lo menos parcialmente, se deben identificar cuales componentes se necesitan para proporcionar la funcionalidad de los casos de uso; éstos son los componentes que se deben diseñar, implementar y probar dentro de esa iteración. Esta identificación proporciona el tiempo requerido para la implementación de los casos de uso, basado en los recursos disponibles y el alcance del trabajo que se necesita para la iteración dada.

3.4.3.3 Preparación del lanzamiento de la versión Beta

Prepararse para el lanzamiento de la versión beta significa un prelanzamiento al final de la fase de construcción. Lo que significa hacer pruebas a nuestro sistema. De ser posible las pruebas deben hacerse por las personas que harán uso del sistema. El prelanzamiento de la versión beta responde a dos propósitos: Primero, se prueba la aplicación, implementado el sistema, y en segundo lugar, proporciona una inspección previa al lanzamiento final.

El encargado del lanzamiento necesita manejar la prueba de la versión beta del sistema para asegurarse de que ambos propósitos se cumplan. Es necesario incluir instrucciones de instalación, manual de usuario y el material de capacitación o no se conseguirá la retroalimentación de los usuarios de la versión beta.

3.4.4 La fase de Transición

El objetivo de la fase de la transición es asegurarse de que el software cumpla completamente con las necesidades de los usuarios. En la fase de la transición se realizan normalmente una o dos iteraciones que incluyan la prueba del sistema en preparación para el lanzamiento y realizar las modificaciones de menor importancia basadas en la retroalimentación del usuario.

3.4.4.1 Objetivos de la fase de Transición

La fase de transición tiene los siguientes objetivos.

1. Hacer pruebas a la versión Beta del sistema y verificar que cumple con las necesidades del usuario.
2. Capacitar a los usuarios y administradores del sistema y asegurarse que cuentan con la información necesaria para el correcto manejo del sistema.
3. Preparar el software necesario para la instalación de la versión final, hacer un respaldo de la nueva versión del sistema, en caso de que se haya tenido una versión anterior. Inclusive adquirir nuevo hardware si es necesario.
4. Tener lista la versión final, es decir el software, así como los documentos, licencias y manuales. Se debe garantizar que todos estén aprobados en el momento de la entrega final.
5. Realizar una prueba de aceptación final, antes de la entrega final. El objetivo de esta prueba es verificar que el software está listo para realizar las funciones y tareas para las que fue construido.
6. Al finalizar un proyecto, es conveniente analizar y documentar como fue el desarrollo del proyecto para mejorar los resultados de los proyectos futuros a través de la experiencia adquirida.

4 Desarrollo

En este capítulo se describe el desarrollo llevado con la metodología elegida (Proceso Unificado). Las cuatro etapas que conforman esta metodología son: inicio, elaboración, construcción y transición. En cada etapa se lleva a cabo una serie de iteraciones. Se decidió utilizar esta metodología porque es un proceso bien definido, iterativo, estructurado dinámicamente que define quién es el responsable de cada tarea, cómo se hacen y cuando hacerlas. Su estructura dinámica permite establecer el ciclo de vida del proyecto. La metodología puede ser utilizada por equipos pequeños y grandes además de que en el diseño y documentación utiliza UML (Unified Modeling Language), el cual es ampliamente utilizado en la actualidad.

Cabe mencionar que el proceso unificado es constantemente actualizado y que ofrece una amplia variedad de herramientas para poder integrarlas al desarrollo del software, lo que implica un mayor conocimiento de tecnología. El proceso unificado es una de las metodologías más importantes para lograr un grado de certificación en el desarrollo de software, siendo utilizado en diversos sectores de la industria como lo son las telecomunicaciones, transportación, industria aeroespacial, militar, manufactura, servicios financieros, etc.

4.1 Inicio

En esta etapa, el propósito es comprender el sistema que se va a construir, establecer los requisitos y el alcance del sistema. Mitigar los riesgos que existan, así como reunir la información necesaria para saber como se va a desarrollar el proyecto. En esta etapa se realizó una sola iteración.

Iteración 1

Las actividades realizadas durante la primera iteración en la etapa de inicio son:

4.1.1 Establecer horarios

Es muy importante que se establezca un horario de trabajo para el desarrollo del proyecto, ya que esto permite estimar el tiempo de las actividades o tareas a realizar en el proyecto, así como el tiempo estimado de la entrega del proyecto.

4.1.2 Crear directorio de participantes

Las personas involucradas en el proyecto, deben tener una excelente comunicación, por lo que es necesario obtener los datos necesarios para localizarlos, como lo es el teléfono y el correo electrónico.

4.1.3 Establecer agenda de reuniones semanales

La comunicación es esencial dentro del proceso unificado y la mitigación temprana de los riesgos. Las reuniones semanales permiten observar el avance del proyecto, así como las posibles dificultades que se pueden presentar y de esta manera dar solución temprana.

4.1.4 Generar el documento de visión

Este documento incluye los beneficios y oportunidades que se obtendrán al desarrollar el proyecto, así como los problemas que resolverá.

Documento de visión SIIPREG

Nombre del proyecto:	Consolidación del SIIPREG
Líder de proyecto:	León Amaro Juana Yadira
Director de tesis:	Saavedra Hernández Honorato
Persona que elaboró:	León Amaro Juana Yadira
Archivo Diagrama de Gantt:	<Archivo de Planner (u otro) correspondiente a este proyecto>

Introducción

Objetivos del proyecto

- Corrección de los errores de la última versión del Siipreg.
- Implementación de los cambios pedidos por el PUEG.

Descripción breve del proyecto

El proyecto a realizar tiene como antecedente dos versiones. Para este proyecto se tomará como punto de inicio la última versión del SIIPREG, es decir, se trabajará con la base de datos ya existente y se hará la implementación de nuevas pantallas, formularios, nuevos elementos que otorguen mayor eficiencia al sistema para la mejor difusión de los programas, centros, proyectos y personas relacionadas con los estudios de género.

Alcances del proyecto

- Los delimitantes del sistema son las impuestas por la versión dos del SIIPREG que es la que actualmente se encuentra en funcionamiento.
- Mejora de la interfaz gráfica.
- Se mostrará más información y mejor orden en la presentación de los datos.
- Se mostrarán más criterios de búsqueda.
- Los campos de fechas limites y de entrevistas pueden quedar vacíos cuando no se tenga la información.
- Se debe mostrar la fecha de actualización de cada uno de los registros.
- Se corregirán errores de búsqueda y captura.

Resultado final esperado

Se espera obtener un sistema eficiente y de calidad que cumpla con las necesidades del usuario.

Justificación y Beneficios

Se corregirán diversos problemas de la última versión del SIIPREG, tales como problemas con búsquedas o referentes a la administración, es decir, altas, bajas o cambios de datos ya sea de personas, instituciones o proyectos. Los beneficios que se obtendrán se relacionan con la eficiencia del sistema y el correcto funcionamiento.

Definir estándar de documentación

El tipo de letra para los documentos es Times New Roman del número 12, en la cabecera el documento deberá llevar en el margen superior izquierdo Facultad de Ingeniería, UNAM, abajo Laboratorio de Multimedia e Internet, en el margen superior derecho el nombre del documento. El pie de página llevará el número de página y el total, por ejemplo: Página 1 de 2.

Definir plan de configuración

Los archivos se guardarán en el servidor del Laboratorio de Multimedia e Internet de la Facultad de Ingeniería, en este caso en el directorio /home/yadirala/siipreg_consolidacion, la persona responsable será Yadira León Amaro.

Revisión de la situación actual

- Resumen de la evaluación de los sistemas existentes.

Actualmente se cuenta con la versión 2 del SIIPREG, la cuál está funcionando. Para este proyecto se realizarán las mejoras o cambios pedidos por el PUEG.

- Resumen de la evaluación de sistemas similares

En este caso el proyecto a desarrollar, tiene como antecedente dos versiones, es decir SIIPREG versión 1 y SIIPREG versión 2. Para realizar este proyecto, se tomará como base la última versión, por lo que se requirió un aprendizaje previo de las diferentes herramientas de programación utilizadas.

Definir Requerimientos

Necesidades de Información:

El usuario necesita consultar información de personas, centros o instituciones y proyectos relacionados con los estudios de género, así como la administración de los mismos. El sistema tiene diferentes módulos, los cuales pueden ser utilizados dependiendo del tipo de usuario.

Requerimientos de Software para la aplicación

Para el funcionamiento de este proyecto se tienen las siguientes dependencias:

- Servidor Web que soporte PHP versión 4.
- Mahejador de bases de datos PostgreSQL.
- Librerias Smarty y Pear para PHP.
- Navegador Web con Javascript activado.

Requerimientos de Hardware para la aplicación

Actualmente se tiene:

- Un servidor con sistema operativo Red Hat Linux con 2 procesadores de 2 GHz con 2 GB de RAM y 140 GB de disco duro.

4.1.5 Identificar casos de uso principales

Un caso de uso es una descripción de la secuencia de las interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa con un diagrama de casos de uso, mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema. Para documentar los casos de uso se utilizó UML. Los actores identificados son cuatro: visitante, administrador, usuario_pueg y usuario_pueg+.

Actor "Visitante"

Es el actor (Figura 4.1), que sólo puede consultar, buscar, guardar, imprimir y enviar un correo de información referente a centros o programas relacionados con los estudios de género.

Caso de Uso "Consultar información"

Este caso de uso permite que los actores consulten información de la red de centros y programas. Este caso de uso se utiliza para definir los siguientes casos de uso por extensión: "Buscar", "Imprimir", "Guardar" y "Enviar un correo a los centros y programas relacionados con los estudios de género".

Caso de Uso "Enviar correo a centro o programa"

Este caso de uso permite que los actores envíen un correo, siempre y cuando la institución tenga una cuenta de correo y se encuentren registrados dentro de la red de centros y programas.

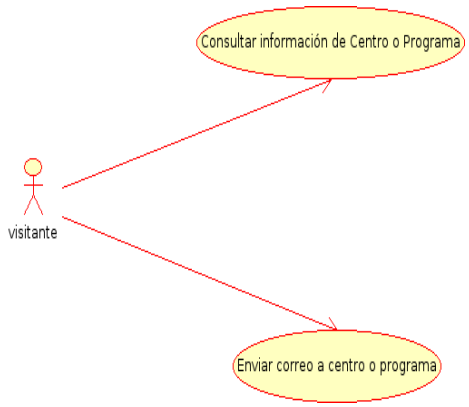

Figura 4.1 Actor "visitante"

Actor "Administrador"

Es el actor (Figura 4.2) que puede emplear todos los caso de uso, pero principalmente los que se relacionan con la administración del sistema.

Caso de Uso "Administración del sistema"

El sistema deberá permitir al administrador emplear este caso de uso. Este caso de uso permite que el actor administre la información relacionada con los estudios de género, es decir, hacer altas, bajas y modificaciones de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos. Así como también consultar los nombres de usuario por tipo de cuenta y los catálogos.

Caso de Uso "Consultar información de estudios de género"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso. Este caso de uso permite que los actores consulten información relacionada con los estudios de género.

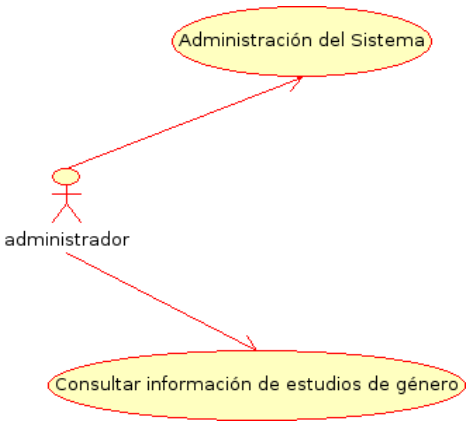

Figura 4.2 Actor "administrador"

Actor "Usuario_pueg"

Es el actor (Figura 4.3) que emplea casi todos los casos de uso, excepto dos: "Administración del Sistema" y "Consultar datos personales UNAM".

Caso de Uso "Consultar información de personas"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso. Este caso de uso sirve para consultar la información detallada de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. En este caso de uso se detectó un error en el módulo UNAM al consultar los antecedentes académicos no se presentan en orden jerárquico, el orden que deben de tener es el siguiente: Educación media superior, técnico, licenciatura,diplomado,especialidad, maestría, doctorado y posdoctorado.

Caso de Uso "Consultar información de instituciones "

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso. Este caso de uso sirve para consultar la información detallada de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Se toman en cuenta sin importar su ubicación geográfica.

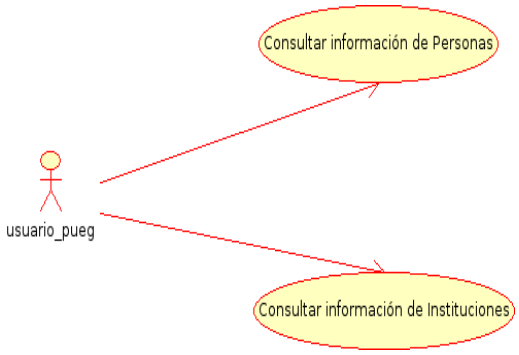

Figura 4.3 Actor "usuario_pueg"

Actor "Usuario_pueg+"

Es el actor (Figura 4.4) que emplea todos los casos de uso excepto el de "Administración del Sistema".

Caso de Uso "Consultar información de personas"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso. Este caso de uso sirve para consultar la información detallada de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente.

Caso de Uso "Consultar información de instituciones"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso. Este caso de uso sirve para consultar la información detallada de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género, se toman en cuenta sin importar su ubicación geográfica.

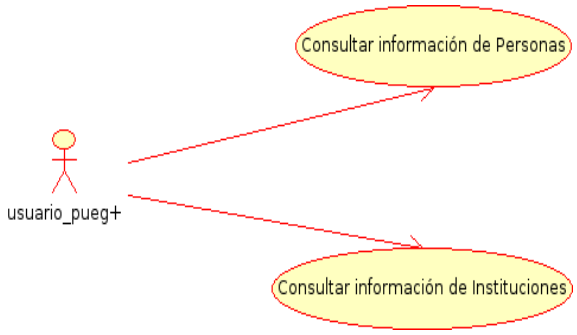

Figura 4.4 Actor "usuario_pueg+"

Caso de Uso "Consultar información de estudios de género"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten información relacionada con los estudios de género.

Caso de Uso "Consultar Información de centro o programa"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten información de la red de centros y programas. A través de este caso de uso se pueden definir los siguientes casos de uso: "Buscar", "Imprimir", "Guardar" y "Enviar un correo a los centros y programas relacionados con los estudios de género".

Caso de Uso "Consultar información UNAM"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género. De aquí se derivan seis casos de uso, en donde los actores pueden tener restricción hacia alguno de ellos.

Caso de Uso "Consultar datos personales UNAM"

El sistema deberá permitir al administrador y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten los datos personales como es el estado civil, el domicilio, fecha de nacimiento, nacionalidad del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Consultar datos institucionales en la UNAM"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten datos del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Consultar antecedentes académicos"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten antecedentes académicos del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Consultar actividades académicas relacionadas con los estudios de género"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten actividades académicas relacionadas con los estudios de género del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Consultar actividades profesionales relacionadas con los estudios de género"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten actividades profesionales relacionadas con los estudios de género del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Consultar toda la información"

El sistema deberá permitir al administrador y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten toda la información de los cuatro casos de uso que se derivan del caso de uso "Consultar información de UNAM del personal académico que labora en la UNAM" el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Consultar información de directorio"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten la información detallada de dos fuentes de datos en el PUEG, la primera es de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. La segunda es de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género, éstas también se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso "Consultar información de personas"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso sirve para consultar la información detallada de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente.

Caso de Uso "Consultar información de instituciones"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso sirve para consultar la información detallada de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género, se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso "Consultar información de financiadoras"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

En este caso de uso se encontró el siguiente error: en el módulo administración en financiadoras no se despliega la información de país, estado y ciudad. También se encontró que en el módulo financiadoras al consultar los datos sobre alguna financiadora, los datos sobre el financiamiento no aparecen.

Caso de Uso "Consultar información de agenda"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.5). Este caso de uso permite que los actores consulten la información de todos los proyectos y actividades que el PUEG tiene calendarizados. En este caso de uso se encontró el siguiente error: en el módulo agenda al consultar los datos de una tarea, la fecha de término es la misma que la de inicio.

Figura 4.5 Caso de uso "Consultar Información de estudios de género"

Caso de Uso "Buscar general"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso permite que los actores busquen información relacionada con los estudios de género.

Caso de Uso “Buscar centro o programa”

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso permite que los actores busquen información de la red de centros y programas.

Caso de Uso “Buscar UNAM”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso permite que los actores busquen información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso “Buscar agenda”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso permite que los actores busquen la información de todos los proyectos y actividades que el PUEG tiene calendarizados.

En este caso de uso se encontró el siguiente error: cuando se requiere buscar un proyecto y se sabe la fecha del proyecto y de la tarea, envía el mensaje de que no existen datos, de igual manera este mensaje aparece al seleccionar búsqueda por todos los proyectos y por fecha para todas las tareas.

Caso de Uso “Buscar financiadora”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso permite que los actores busquen información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

Caso de Uso “Buscar directorio”

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso permite que los actores busquen la información detallada de dos fuentes de datos en el PUEG, la primera es de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. La segunda es de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Éstas también se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso "Buscar personas"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso sirve para buscar la información detallada de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente.

Caso de Uso "Buscar instituciones"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.6). Este caso de uso sirve para buscar la información detallada de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Se toman en cuenta sin importar su ubicación geográfica.

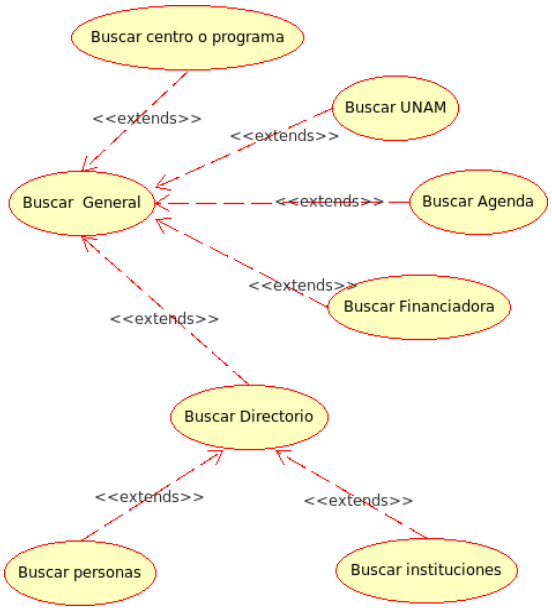

Figura 4.6 Caso de uso "Buscar General"

Caso de Uso "Administración del sistema"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.7). Este caso de uso permite que el actor administre la información relacionada con los estudios de género, es decir, hacer altas, bajas y modificaciones de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos. Así como también se pueden consultar los nombres de usuario por tipo de cuenta y los catálogos.

Caso de Uso "Agregar"

El sistema deberá permitir al administrador emplear este caso de uso. Este caso de uso (Figura 4.7) permite que el actor realice altas de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos.

Caso de Uso "Borrar"

El sistema deberá permitir al administrador emplear este caso de uso. Este caso de uso (Figura 4.7) permite que el actor realice bajas de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos.

Caso de Uso "Modificar"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.7). Este caso de uso permite que el actor realice cambios de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos.

Caso de Uso "Consultar claves de usuario"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.7). Este caso de uso permite que el actor consulte los nombres de usuario por tipo de cuenta.

Caso de Uso "Consultar catálogos"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.7). Este caso de uso permite que el actor consulte los catálogos existentes.

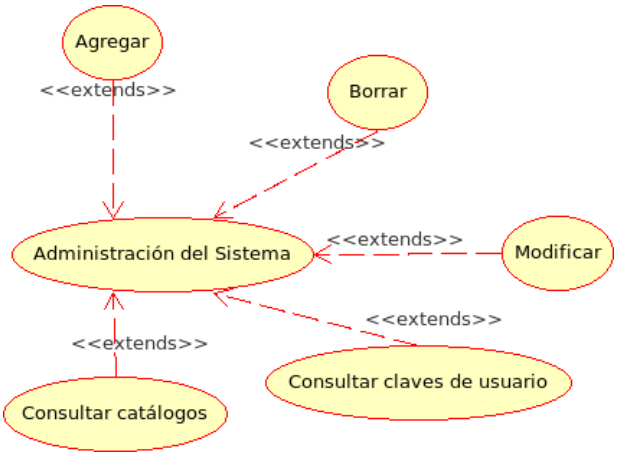

Figura 4.7 Caso de uso "Administración del sistema"

Caso de Uso "Agregar general"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos.

Caso de Uso "Agregar persona"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de personas. En este caso de uso se pide el siguiente cambio: en el módulo de administración, al agregar una persona, se puede modificar, agregar o borrar alguna investigación a una persona y muestra una pantalla la cual tiene opciones para poner la fecha de inicio y de término. Se requiere eliminar este rubro.

También en este caso de uso se pide que en el módulo de administración, al agregar personas se pide la fecha de nacimiento. Si no se tiene el dato de la fecha de nacimiento se debe poder dejar en blanco.

Caso de Uso "Agregar proyecto"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de proyectos.

En este caso de uso se encontró el siguiente error: en el módulo administración, en proyectos no permite dar de alta nuevos proyectos en la pantalla. Aparece el aviso error “No se realizó la operación”.

Caso de Uso “Agregar catálogos”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de catálogos.

En este caso de uso se encontró el siguiente error: en el módulo administración en el catálogo de títulos, no agrega los títulos.

Caso de Uso “Agregar financiadora”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de financiadoras.

Los cambios pedidos en este caso de uso en el módulo administración en financiadoras son: los campos de fechas límite y de entrevistas deben aceptar “No aplica” cuando no se tenga la información y para el campo de filosofía agregar más espacio para caracteres, de 1000 a 1500.

Caso de Uso “Agregar financiamiento”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de financiamientos.

En este caso de uso se encontró el siguiente error: en el módulo administración en financiamientos al querer agregar un financiamiento no se despliega la dependencia financiera.

Caso de Uso “Agregar dependencia”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de dependencias.

Caso de Uso “Agregar institución”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de instituciones.

Caso de Uso "Agregar tarea"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.8). Este caso de uso permite que el actor realice altas de tareas.

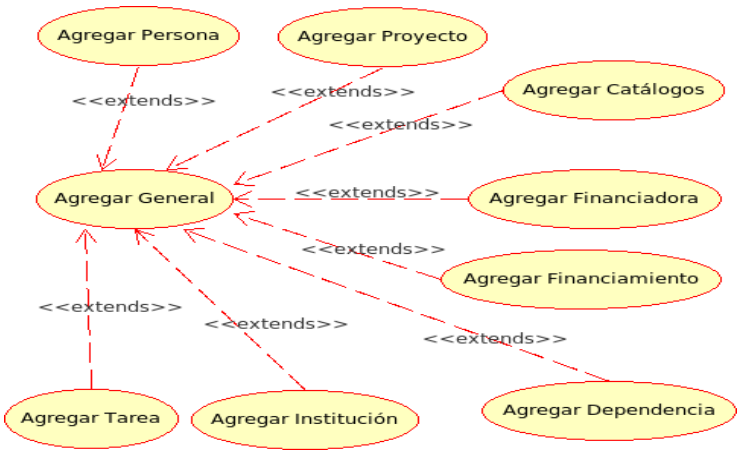

Figura 4.8 Caso de uso "Agregar general"

Caso de Uso "Modificar general"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios en los datos existentes de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos.

Caso de Uso "Modificar persona"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de personas.

En este caso de uso se pide el siguiente cambio: en el módulo de administración, al modificar una persona, se le puede modificar, agregar o borrar alguna investigación y muestra una pantalla la cual tiene opciones para poner la fecha de inicio y de término. Se requiere eliminar este rubro.

Caso de Uso "Modificar proyecto"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de proyectos.

En este caso de uso se encontró el siguiente error: en el módulo administración, en proyectos no permite modificar nuevos proyectos en la pantalla. Aparece el aviso "Error no se realizó la operación".

Caso de Uso "Modificar catálogo"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de catálogos.

Caso de Uso "Modificar financiadora"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de financiadoras.

Caso de Uso "Modificar financiamiento"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de financiamientos.

Caso de Uso "Modificar dependencia"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de dependencias.

Caso de Uso "Modificar institución"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de instituciones.

Caso de Uso "Modificar tarea"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.9). Este caso de uso permite que el actor realice cambios de los datos ya existentes de tareas.

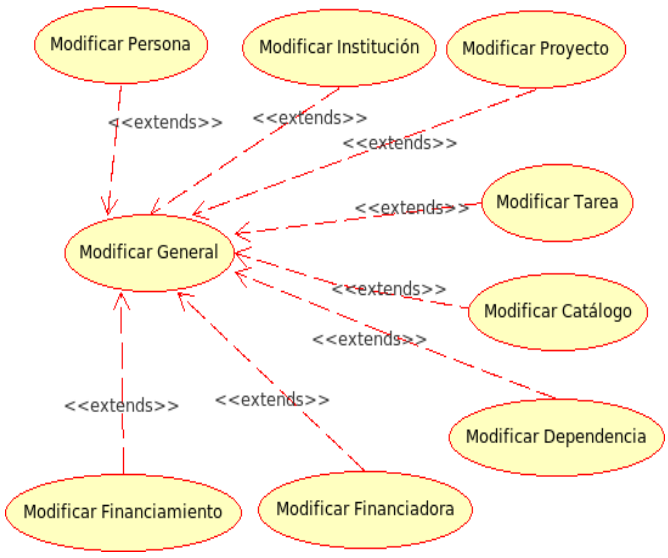

Figura 4.9 Caso de uso "Modificar general"

Caso de Uso "Borrar general"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos.

Caso de Uso "Borrar persona"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de personas.

Caso de Uso "Borrar proyecto"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de proyectos.

Caso de Uso "Borrar catálogo"

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de catálogos.

Caso de Uso “Borrar financiadora”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de financiadoras.

Caso de Uso “Borrar financiamiento”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de financiamientos.

Caso de Uso “Borrar dependencia”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de dependencias.

Caso de Uso “Borrar institución”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de instituciones.

Caso de Uso “Borrar tarea”

El sistema deberá permitir al administrador emplear este caso de uso (Figura 4.10). Este caso de uso permite que el actor realice bajas de tareas.

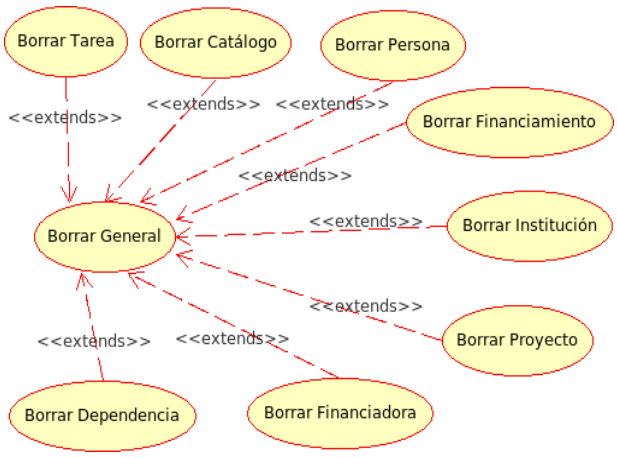

Figura 4.10 Caso de uso "Borrar General"

Caso de Uso "Imprimir general"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso permite que los actores impriman información relacionada con los estudios de género.

Caso de Uso "Imprimir centro o programa"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso permite que los actores impriman información de la red de centros y programas.

Caso de Uso "Imprimir UNAM"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso permite que los actores impriman información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Imprimir financiadora"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso permite que los actores impriman información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

Caso de Uso "Imprimir directorio"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso permite que los actores imprimir la información detallada de dos fuentes de datos en el PUEG, la primera es de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. La segunda es de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Éstas también se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso “Imprimir persona”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso sirve para imprimir la información detallada de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente.

Caso de Uso “Imprimir institución”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso sirve para imprimir la información detallada de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso “Imprimir agenda”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.11). Este caso de uso permite que los actores guarden la información de todos los proyectos y actividades que el PUEG tiene calendarizados.

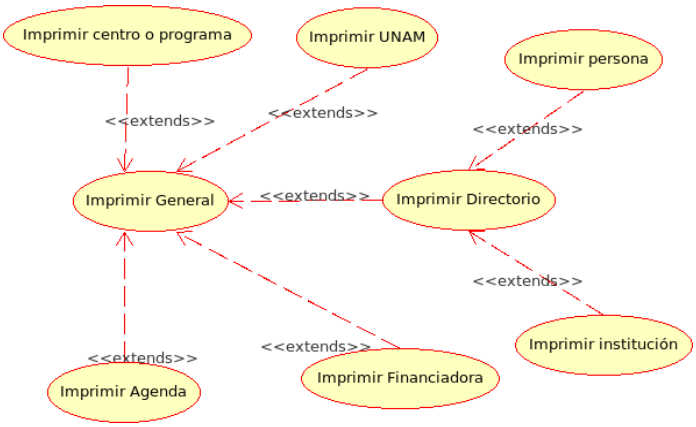

Figura 4.11 Caso de uso "Imprimir general"

Caso de Uso “Guardar general”

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso permite que los actores guarden información relacionada con los estudios de género.

Caso de Uso "Guardar centro o programa"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso permite que los actores guarden información de la red de centros y programas.

Caso de Uso "Guardar UNAM"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso permite que los actores guarden información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso "Guardar directorio"

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso permite que los actores guarden la información detallada de dos fuentes de datos en el PUEG, la primera es de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. La segunda es de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Éstas también se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso "Guardar personas"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso sirve para guardar la información detallada de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente.

Caso de Uso "Guardar instituciones"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso sirve para guardar la información detallada de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso “Guardar agenda”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso permite que los actores guarden la información de todos los proyectos y actividades que el PUEG tiene calendarizados.

Caso de Uso “Guardar financiadoras”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.12). Este caso de uso permite que los actores guarden información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

Figura 4.12 Caso de uso "Guardar general"

Caso de Uso “Enviar correo general”

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13).

Este caso de uso permite que los actores envíen un correo para obtener información relacionada con los estudios de género.

Caso de Uso “Enviar correo centro o programa”

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13).

Este caso de uso permite que los actores envíen correos para obtener información de algún centro o programa.

Caso de Uso “Enviar correo UNAM”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13).

Este caso de uso permite que los actores envíen un correo para obtener información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Caso de Uso “Enviar correo financiadoras”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13).

Este caso de uso permite que los actores envíen un correo para obtener información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

En este caso de uso se encontró el siguiente error: en el módulo financiadoras al intentar enviar un correo aparece el aviso "No existen datos”.

Caso de Uso “Enviar correo directorio”

El sistema deberá permitir al visitante, administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13).

Este caso de uso permite que los actores envíen un correo al personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente y a todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Éstas también se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso “Enviar correo instituciones”

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13). Este caso de uso sirve para enviar un correo a todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género. Se toman en cuenta sin importar su ubicación geográfica.

Caso de Uso "Enviar correo personas"

El sistema deberá permitir al administrador, usuario_pueg y usuario_pueg+ emplear este caso de uso (Figura 4.13). Este caso de uso sirve para enviar un correo a todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente.

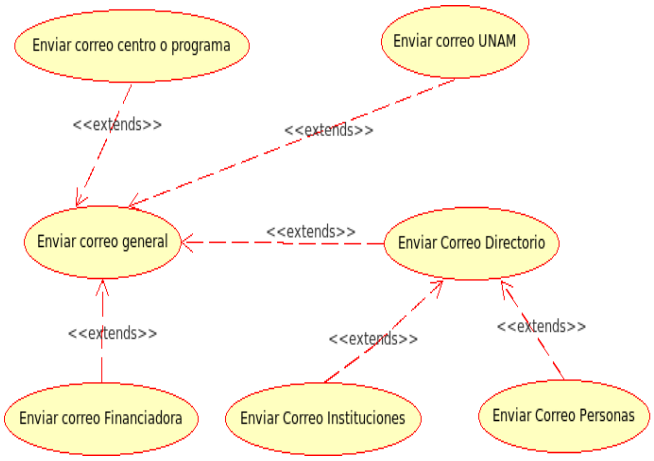

Figura 4.13 Caso de uso "Enviar correo general"

4.1.6 *Identificar soluciones posibles*

Para las modificaciones y cambios en esta versión del SIIPREG, se trabajó con el mismo lenguaje de programación y librerías. No se implementaron otras estructuras de datos ni se hicieron modificaciones a la arquitectura principal.

Para tener un control adecuado de los problemas a solucionar, se utilizó Bugzilla. Bugzilla es una herramienta de seguimiento de errores (*bugs*) desarrollada por la organización Mozilla.

Al estar basada en web y ser de código abierto, se ha convertido en la herramienta de seguimiento de errores elegida por muchos proyectos. Los errores pueden ser enviados por todos los desarrolladores y pueden ser asignados a un desarrollador en particular. Cada error puede estar en diferentes estados, así como ir acompañado de notas del usuario o ejemplos de código que ayuden a corregir el error.

La noción de error en Bugzilla es muy general; por ejemplo, Mozilla.org lo utiliza también para registrar las peticiones de nuevas funcionalidades. La manera de corregir los errores encontrados en SIIPREG, es haciendo pruebas al sistema dentro del módulo donde fue reportado el error, se detecta el archivo donde se encuentra el error.

4.1.7 Generar el glosario

Administración.- Este módulo sirve para administrar el sistema, es decir, hacer altas, bajas y modificaciones de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos. Así como también se pueden consultar los nombres de usuario por tipo de cuenta y los catálogos.

Agenda.- Este módulo sirve para consultar la información de todos los proyectos y actividades que el PUEG tiene calendarizados, bajo una clasificación muy detallada de cada uno de ellos para poder identificarlos de forma inmediata cuando así se requiera.

Directorio.- Este módulo sirve para consultar la información detallada de dos fuentes de datos en el PUEG, la primera es de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. La segunda es de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género, éstas también se toman en cuenta sin importar su ubicación geográfica.

Financiadoras.- Este módulo sirve para consultar la información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

Redes.- Este módulo sirve para obtener la información que contienen las Redes Nacionales y Latinoamericanas sobre Centros y Programas de Estudios de la Mujer y de Género de Instituciones de Educación Superior.

UNAM.- Este módulo sirve para consultar la información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

4.1.8 Generar el plan de desarrollo de proyectos

El plan de desarrollo se generó con planner.

1 **Ciclo 1**

1.1 Inicio.

1.1.1 Iteración 1.

1.1.1.1 Establecer horarios.

1.1.1.2 Crear directorio de participantes (correos, teléfonos).

1.1.1.3 Establecer agendas de reuniones semanales.

1.1.1.4 Generar el documento de visión.

1.1.1.5 Identificar casos de usos principales.

1.1.1.6 Identificar soluciones posibles.

1.1.1.7 Construir prototipo inicial.

1.1.1.8 Generar el glosario.

1.1.1.9 Diseño inicial de la base de datos.

1.1.1.10 Generar el plan de desarrollo de proyectos (SDP).

1.1.1.11Generar la lista de riesgos.

1.1.1.12 Selección de proceso a seguir y herramientas.

1.1.1.13 Definir estándar de documentación.

1.1.1.14 Definir plan de configuración.

1.1.1.15 Generar el documento de modelo negocio.

1.1.1.16 Obtener la aprobación de la dirección y del usuario.

1.1.1.17 Generar documento de propuesta de tesis.

1.1.2 Hito de objetivos del ciclo de vida (LCO).

1.2 Elaboración.

1.2.1 Iteración 1.

1.2.1.1 Completar requerimientos.

1.2.1.2 Dar prioridad a los casos de uso.

1.2.1.3 Actualizar el plan de trabajo

1.2.1.4 Mitigar los riesgos.

1.2.1.5 Diseñar, implementar y validar la arquitectura base.

1.2.1.6 Diseñar el plan de instalación (deployment).

1.2.1.7 actualizar el documento de modelo negocio.

1.2.1.8 Instalar las herramientas de desarrollo.

1.2.1.9 Implementar plan de configuración.

1.2.1.10 Construir prototipo de la interface de usuario.

1.2.1.11 Actualizar el glosario.

1.2.2 Iteración 2.

1.2.2.1 Asegurar cobertura de la arquitectura.

1.2.2.2 Completar el diseño de la base de datos.

1.2.2.3 Probar las funciones faltantes de la arquitectura.

1.2.2.4 Verificar desempeño.

1.2.2.5 Verificar requerimientos no funcionales.

1.2.2.6 Verificar las tareas para la fase de construcción.

1.2.3 Hito de arquitectura del ciclo de vida (LCA).

1.3 **Construcción.**

1.3.1 Iteración 1.

1.3.1.1 Actualizar lista de riesgos.

1.3.1.2 Probar el plan de configuración.

1.3.1.3 Buscar paralelismo.

1.3.1.4 Refinar y actualizar la arquitectura.

1.3.1.5 Implementar 50% de lafuncionalidad.

1.3.1.6 Probar implementación (módulos e integración)

1.3.2 Iteración 2.

1.3.2.1 Actualizar lista de riesgos.

1.3.2.2 Implementar 50% de lafuncionalidad.

1.3.2.3 Probar implementación (módulos e integración).

1.3.2.4 Implementar plan de instalación(deployment).

1.3.2.5 Generar manual de usuario.

1.3.2.6 Generar manual de administrador.

1.3.2.7 Generar plan de capacitación de usuarios.

1.3.2.8 Liberar versión Beta.

1.3.3 Hito de capacidad de operación inicial (IOC).

1.4.1 **Transición**

1.4.1.1 Iteración 1.

1.4.1.2 Beta test.

1.4.1.3 Capacitar usuarios y administradores.

1.4.1.4 Preparar sitio de instalación.

1.4.1.4 Convertir bases de datos.

1.4.1.5 Tener lista la versión final (software, documentos, licencias y manuales).

1.4.1.6 Comunicar al cliente que los objetivos han sido completados.

1.4.2 Hito de liberación de producto (PR).

4.1.9 Generar la lista de riesgos

1. No tener conocimiento del Lenguaje de Modelado Unificado (UML) que es un lenguaje gráfico para representar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar conceptos como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son clases en un lenguaje determinado, esquemas de bases de datos y componentes de software reusables.
2. No tener conocimientos del manejo de todas las herramientas de Umbrello para la realización de los casos de uso.
3. Probar la metodología del proceso unificado, por lo cual todavía no se tiene un conocimiento adecuado de la organización de las actividades.

4.1.10 Selección de proceso a seguir y herramientas

El proceso o metodología es el proceso unificado, las herramientas utilizadas son de software libre.

Las herramientas utilizadas son:

- Pear
- Smarty
- Umbrello
- Planner

Pear.- Es una herramienta, que permite de manera eficaz trabajar con la base de datos del sistema, ayuda a trabajar con sentencias cortas y menos complicadas que si se trabajara sólo con SQL.

Smarty.- Es una herramienta que trabaja a base de plantillas de html, lo cual permite no repetir código de html y sólo basta con llamar a la plantilla requerida y de esa manera optimizamos código y trabajo. Smarty require cuatro directorios llamados 'templates/', 'templates_c/', 'configs/' y 'cache/'.

Umbrello.- Umbrello UML Modeller es una herramienta de diagramas que ayuda en el proceso del desarrollo de software. Umbrello UML Modeller facilita la creación de un producto de alta calidad, en este caso se utilizó para la creación de los diagramas de casos de uso.

Planner.- Es un programa de software libre de gestión de proyectos que permite crear planes para un proyecto y seguir su progreso. Es un programa muy útil para la generación de diagramas de Gantt y sus recursos asociados. Está programado en Java y corre en entornos Windows y Linux.

Lista de características de Planner

- Diagramas de Gantt.
- Gestión de recursos.
- Gestión de tareas.
- Calendario.
- Costos de proyecto.

4.3 Elaboración

La elaboración es la segunda fase del ciclo de vida del proceso unificado, las tareas principales dentro de esta fase son: atender los riesgos importantes, construcción de un esqueleto o bosquejo de la arquitectura del sistema, refinar y desarrollar los planes del proyecto que fueron producidos en la fase de la inicio.

Iteración 1

4.2.1 Mitigar los riesgos

En la fase de inicio, se generó una lista de riesgos, a los cuáles se les dio solución a través del aprendizaje de UML, Umbrello y el proceso unificado. En lo que respecta a los cambios que requiere el PUEG, se obtuvo un documento con los cambios que solicitaban.

4.2.2 Diseñar, implementar y validar la arquitectura base

Arquitectura del sistema

Descripción General

El Sistema de Información Institucional y de Personas Relacionadas con los Estudios de Género (SIIPREG), es una base de datos que reúne información de Centros y Programas de Estudios de Género de Instituciones de Educación Superior (Módulo Redes) de México y América Latina, del personal académico de la UNAM que trabaja desde la perspectiva de género (Módulo UNAM), así como de otras Personas e Instituciones (Módulo Directorio) vinculadas a estas temáticas. El SIIPREG es una importante herramienta de trabajo para el PUEG y un instrumento de vinculación, difusión y promoción para quienes se encuentran registradas en el mismo.

Los objetivos fundamentales del SIIPREG son difundir el trabajo de las organizaciones y personas que trabajan desde la perspectiva de género, promover la vinculación entre ellas y contactar a especialistas en distintas temáticas para participar en actividades académicas del PUEG o de otras instituciones, así como en entrevistas y programas de distintos medios de comunicación.

Dependencias

El sistema funciona:

- Servidor web con PHP 5 o mayor
- PEAR y los módulos DB, DB_DataObject, Date, Mail, Mail_Mime.
- Manejador de bases de datos PostgreSQL.
- Servidor de correo.
- Conexión a Internet.

Módulos

Redes.- Este módulo sirve para obtener la información que contienen las Redes Nacionales y Latinoamericanas sobre Centros y Programas de Estudios de la Mujer y de Género de Instituciones de Educación Superior.

UNAM.- Este módulo sirve para consultar la información del personal académico que labora en la UNAM el cual realiza actividades relacionadas con los estudios de género.

Agenda.- Este módulo sirve para consultar la información de todos los proyectos y actividades que el PUEG tiene calendarizados, bajo una clasificación muy detallada de cada uno de ellos para poder identificarlos de forma inmediata cuando así se requiera.

Directorio.- Este módulo sirve para consultar la información detallada de dos fuentes de datos en el PUEG, la primera es de todo el personal que labora en el PUEG tomando en cuenta todas las regiones geográficas en donde éste se haga presente. La segunda es de todas las instituciones que están ligadas al PUEG debido a su interés en los proyectos de género, éstas también se toman en cuenta sin importar su ubicación geográfica.

Financiadoras.- Este módulo sirve para consultar la información de las instituciones que financian los proyectos que el PUEG realiza basados en el estudio del género.

Administración.- Este módulo sirve para administrar el sistema, es decir, hacer altas, bajas y modificaciones de instituciones, dependencias, financiadoras, personas, proyectos, tareas, catálogos y financiamientos. Así como también se puede consultar los nombres de usuario por tipo de cuenta y los catálogos.

Descripción de la Arquitectura

Descripción de la Base de Datos

La base de datos se construyó en PostgreSQL, el modelado de la base de datos se realizó con PowerDesigner, que es una herramienta de modelado. Las principales tablas que maneja la base de datos son: Personas, Dependencias, Instituciones, Proyectos, Tareas, Financiadoras y Finaciamientos. La base de datos trabaja con los siguientes catálogos: Actividad global, Estado civil, Grupo de trabajo, Tema de género, Tipo de enfoque, Tipo de investigación, Línea de Investigación, Área, Estado tarea, Temática, Tipo de financiadora, Tipo de proyecto, Ciudad, Estatus, Pais, Tipo de actividad, Tipo de financiamiento, Título, Estado, Grado académico, Región geográfica, Tipo de actividad, Tipo de institución.

El manejo de las bases de datos se hace a través de PEAR, que es una herramienta que permite utilizar el lenguaje SQL de una manera más sencilla, ya que las setencias son más cortas.

Descripción de las clases

DB_DataObject

Esta clase es de PEAR y con ella se crean, todos los objetos necesarios para manejar la base de datos, un objeto por tabla existente en la base de datos.

Clase Usuario

Esta clase es utilizada para representar un usuario del sistema, el nombre de este archivo es usuario.class.php y requiere de dos archivos DataObjects2.php y DataObjects/Persona.php.

Clase navegación

Esta clase es utilizada para representar la navegación a través del sistema. El archivo es navegacion.class.php.

Clase pila

Esta clase, como su nombre lo indica, es una pila que permite almacenar y recuperar datos mediante las operaciones push y pop.

Uso de las Plantillas

Smarty es un motor de plantillas para PHP. Más específicamente, esta herramienta facilita la manera de separar la aplicación lógica y el contenido en la presentación. Las plantillas están hechas en lenguaje html lo cual permite no repetir código y sólo basta con llamar a la plantilla requerida y de esa manera se ahorra código y trabajo. Smarty require cuatro directorios llamados 'templates/', 'templates_c/', 'configs/' y 'cache/'. Los archivos se crean en un editor de texto, se guardan en el directorio templates con la extensión tpl. Todas las plantillas requieren cargar el archivo siipreg2.css para mantener un mismo estilo gráfico.

Plantilla menu.tpl

El archivo menu.tpl es una plantilla que dibuja un menú a través de una tabla. En este archivo también se utilizan las etiquetas Map, Img y Area. La imagen utilizada en la cabecera es menu-omega.jpg, en esta tabla se encuentran una liga por cada módulo del sistema. Dentro de esta plantilla se utiliza otra plantilla llamada "pie_pagina_vacio.tpl, que como su nombre lo indica, es el pie de página.

Plantilla busqueda.tpl

El archivo busqueda.tpl es una plantilla que dibuja la cabecera y el pie de página en todas las páginas programadas para la búsqueda. Esta plantilla incluye las plantillas cabecera.tpl y pie_pagina_vacio.tpl. En este archivo se verifica la fecha y además se dibuja una tabla con los botones Restaurar y Búsqueda.

Plantilla lista.tpl

El archivo lista.tpl es una plantilla que dibuja la cabecera y el pie de página en todas mis páginas programadas para desplegar la lista de la búsqueda, esta plantilla incluye las plantillas cabecera.tpl y pie_pagina_vacio.tpl. En este archivo se dibuja una tabla con los botones Enviar correo, Consultar,Guardar, Imprimir y Regresar.

Plantilla datos.tpl

El archivo datos.tpl es una plantilla que dibuja la cabecera y el pie de página en todas mis páginas programadas para desplegar la lista de la consulta, esta plantilla incluye las plantillas cabecera.tpl y pie_pagina_vacio.tpl. En este archivo se dibuja una tabla con los botones Guardar, Imprimir y

Regresar.

Plantilla cabecera.tpl

El archivo cabecera.tpl es una plantilla que dibuja la cabecera, es decir la imagen de la cabecera (Figura 4.14), así como las imágenes de los botones que llevan a los diferentes módulos.

Figura 4.14 "Cabecera"

Plantilla pie_de_pagina_vacio.tpl

El archivo pie_de_pagina_vacio.tpl es una plantilla que dibuja el pie de página de las páginas del sistema, es decir la imagen del pie de página de las páginas del sistema (Figura 4.15).

Figura 4.15 "Pie de página"

Plantilla login_invalido.tpl

El archivo login_invalido.tpl es una plantilla que da el aviso que el login proporcionado es inválido para el sistema.

Plantilla permiso_invalido.tpl

El archivo permiso_invalido.tpl es una plantilla que da el aviso que el tipo de cuenta proporcionada es inválido para consultar el módulo requerido.

Plantillas para formularios

Para ejemplificar el uso de las plantillas utilizaremos los archivos personas.php y bus_agenda.php, el primero utiliza las primeras cinco plantillas que a continuación se mencionan y el segundo sólo las dos últimas.

Plantilla llenar.tpl

Plantilla fecha.tpl

Plantilla intercambio_java.tpl

Plantilla intercambio_java_elegir.tpl

Plantilla select.tpl

Plantilla calendario.tpl

Plantilla select.tpl

Es una plantilla que dibuja una lista de selección, por ejemplo esta plantilla es utlizada en personas.php.

Plantilla llenar.tpl

Para esta plantilla se utiliza una tabla, una caja de tipo texto, una lista de selección.

Plantilla calendario.tpl

Para esta plantilla se utiliza una tabla , tres botones del tipo radio y se utiliza la función de html_select_date para crear un menu de fechas.

Plantilla fecha.tpl

Para esta plantilla se utiliza una tabla y se utiliza la funcion de html_select_date para crear un menú de fechas.

Pantilla intercambio_java.tpl

Para esta plantilla se utiliza una tabla, una caja de tipo texto y una lista de selección

intercambio_java_elegir.tpl

Esta plantilla cuenta con los mismos elementos de intercambio_java.tpl.

Cabeceras y archivos incluidos

siipreg2.php

Es necesario incluir o cargar el archivo siipreg2 para la inicialización de la aplicación. Al terminar de ejecutar este script ya existirá la variable $smarty, la conexión a la base de datos se podrá utilizar con DB_DataObject de PEAR, se habrá verificado que el usuario ha dado un login y password

correctos y se podrán usar las variables de sesión $usuario_sesion y $navegacion_sesion.

El archivo siipreg2.php incluye los siguientes archivos:

DB/DataObject.php

clases/navegacion.class.php

conexion_bd.php

clases/pila.class.php

verifica_usuario.php

Smarty.class.php

utilerias.php

conexion_bd.php

El archivo conexion_bd.php es el código usado para cargar y almacenar la configuración de DataObjects.

verifica_usuario.php

El archivo verifica_usuario.php, crea una sesion(una sesion es la secuencia de páginas que un usuario visita en un sitio web. Desde que entra al sitio(SPHP, 2008)) y verifica si no existe la variable smarty, así como la sesión y despliega la plantilla login-invalido.tpl, dentro de este archivo se cargan los archivos:

Smarty.class.php

clases/usuario.class.php

login_invalido.tpl

correo.php

El archivo correo.php, ayuda al sistema a obtener el correo de la persona, institución, financiadora o cualquier otro dependiente del módulo que se esté visitando. Dentro de este programa se utilizan los siguientes archivos:

siipreg2.php

DataObjects/Persona.php

DataObjects/Dependencia.php

DataObjects/Institucion.php

Mail/RFC822.php

correo.tpl

no_datos.tpl

menu.php

Este archivo trabaja con sesiones a través de la clases navegación y usuario, verificando si ya existe el usuario o no, en caso de no existir se crea la sesión y se verifica el login y password. Si son correctos, se despliega la plantilla menu.tpl , en caso contrario de la plantilla que se despliega es la de login_invalido.tpl.

siipreg2.css

Es una hoja de estilo que se debe cargar en las plantillas que se elaboren, ya que en ella se encuentran establecidos los márgenes, los tipos de letra, así como el color del texto.

siipreg2.ini

Este archivo se conecta a la base de datos, define donde se encuentran los scripts del sistema y las tablas con las queremos trabajar.

utilerias.php

Este archivo contiene las funciones básicas que se pueden utilizar en cualquier script.

Descripción del módulo de redes (como ejemplo para todos los otros módulos)

bus_redes.php

El archivo bus_redes.php, es el código que permite buscar información acerca de la red de centros y programas.

Para tener acceso a los diferentes módulos, es necesario que se verifique el tipo de cuenta, por lo que se usa la función verificar_permisos, que se encuentra dentro del archivo utilerias.php. Si el tipo de cuenta, tiene los permisos para entrar al módulo, se utilizará la información, de lo contrario se despliega la plantilla permiso_invalido.tpl.

lis_redes.php

El archivo lis_redes.php, es el código que permite desplegar la información de los nombres de los Centros o Programas de Estudios de la Mujer y de Género de Instituciones de Educación Superior. Si estos centros o programas pertenecen a una institución y a su vez a alguna dependencia y el país donde esta el centro o programa.

dat_redes.php

El archivo dat_redes.php, es el código que permite desplegar la información más detallada de cada Centro y Programa relacionados con los estudios de género.

Iteración 2

4.2.3 Completar el diseño de la base de datos

La base de datos sufrió algunas modificaciones:

- Eliminación de tablas:

  Zona

  Region_postal

  Centro

  Tematica_centro

  Activdadg_centro

- Creación de Nuevas Tablas

  Tematica_dependencia

Actividadg_dependencia

- Modificaciones al agregar campos a las tablas existentes:

Actividad_personal, se le agregaron los campos id_tema, id_doc, id_per, id_profesional, id_servicio, id_investigacion; además al campo nombre se le aumentó la capacidad del número de caracteres de 100 a 150.

Area, se le agregó el campo id_linea.

Cat_ciudad, se le agregaron los campos id_pais, id_edo.

Cat_estado, se le agregó el campo id_pais.

Cat_grado, se le agregó el campo orden.

Contacto, se le agregaron los campos id_ins, id_depn.

Departamento, se le agregaron los campos id_depn, id_ins, id_per.

Dependencia, se le agregaron los campos id_ins, id_per, id_pais, id_estado, id_ciudad, email_depn, pag_web_depn, cargo_cen, biblioteca_depn, horario_depn, madre_depn, es_centro, y en lugar del campo lada_depr se cambio por lada_depn.

Docencia, se le agregaron los campos id_tipo_mat, id_ins, id_depn.

Financiadora, esta tabla tenía el nombre de Inst_finra, se le agregaron los campos id_ins, id_depn, id_tipo_finra, y al campo filosofia_ins se le aumento la capacidad de caracteres era de 200 aumentó a 1500.

Financiamiento, se le agregaron los campos id_ins, id_depn, id_finto.

Institucion, se le agregaron los campos id_pais, id_edo, id_ciudad, id_tipo_ins.

Investigacion, se le agregaron los campos id_estatus, id_tipo_inv.

Persona, se le agregaron los campos id_pais, id_edo, id_ciudad, id_edo_civ, pert_registro.

Proyecto, se le agregaron los campos id_estatus, id_tipo_pro, id_act_pro, pro_id_pro, id_area, num_participantes.

Tarea, se le agregaron los campos id_pro, id_estado_tarea

Titulo, se le agregó el campo id_grado.

4.3 Construcción

Durante la fase de construcción, hay que centrarse en refinar los requerimientos, el diseño, la implementación y las pruebas. En la fase de construcción se realizaron dos iteraciones.

Iteración 1

Se implementaron las soluciones para resolver los errores del sistema.

- Modificaciones a la base de datos, agregar campos, extender la capacidad del número de caracteres en algunos campos.
- Modificaciones de algunas funciones en PHP.
- Verificación de variables.
- Modificaciones en algunas plantillas (archivos tpl).
- Modificación al código en Javascript.
- Se modificaron algunos ciclos en los archivos requeridos.
- Modificaciones en algunas consultas de Pear.
- Modificaciones en el archivo siipreg.ini.

Iteración 2

Se probó el sistema módulo por módulo del SIIPREG tomando como guía la lista de casos de uso.

4.4 Transición

En esta fase se instaló la versión tres del SIIPREG en las instalaciones del PUEG en la Torre de Humanidades II. Se hizo un respaldo de los archivos ya modificados del sistema SIIPREG, que se tienen en el servidor mmedia3 del Laboratorio de Multimedia e Internet. Este respaldo se dejó instalado en el servidor generounix.pueg.unam.mx, así como también la versión modificada de la base de datos.

5 CONCLUSIONES

Al manejar grandes cantidades de información es importante que las empresas e instituciones la administren correctamente para poder utilizarla de forma rápida y sencilla. Como consecuencia, es necesaria la creación y el desarrollo de sistemas de cómputo para este propósito que cuenten con una interfaz fácil de utilizar. El acceso a la información se hace más fácil cuando los sistemas que se desarrollan se puede utilizar a través de Internet, ya que lo puede utilizar cualquier persona sin importar en donde se encuentre ubicado geográficamente. Por esta razón el PUEG almacenó la información en una base de datos única que puede consultarse desde Internet. El sistema SIIPREG permite hacer consultas de personas, instituciones, dependencias, financiadoras y proyectos de forma rápida y sencilla.

Partiendo de los objetivos planteados, donde se estableció que se debían corregir fallas del SIIPREG y que se requería de nuevas funciones en el sistema, se llevaron a cabo las correcciones, los cambios y se hizo una revisión minuciosa de todo el sistema para que la nueva versión quedara exenta de errores al realizar las consultas. Los errores que se corrigieron fueron generalmente en la búsqueda de información ya que el sistema mostraba avisos de error o los resultados no eran correctos.

La corrección de las fallas encontradas en la versión dos y la implementación de los cambios en el sistema, ayudan a que la versión tres del SIIPREG que ya está en operación, cumpla con el objetivo primordial de ser una herramienta de vinculación, promoción y difusión para quienes se encuentran registradas en el mismo. También permite establecer comunicación con las personas que están interesadas en los estudios de género y son ajenas al grupo ya registrado en el sistema SIIPREG.

Es también importante la seguridad de la información. Para aumentar la seguridad se pueden desarrollar sistemas de información automatizados. El acceso a la información puede estar controlado por un complejo sistema de contraseñas o limitado a ciertas áreas o personal. Si está bien protegido, es difícil hacer mal uso de la información. El SIIPREG está diseñado de tal forma que la información contenida en él se encuentra restringida ya que para utilizarla existen diferentes tipos de usuarios, que dependiendo del tipo al que pertenezcan podrán hacer uso de la información contenida en los diversos módulos del SIIPREG.

De la experiencia del desarrollo del sistema se puede concluir que es importante utilizar una metodología para el desarrollo de software porque ayuda a la mejor comprensión del alcance del sistema, la planificación de los tiempos, los recursos, así como a reconocer los riesgos. Durante la corrección del SIIPREG se utilizaron herramientas de programación que facilitaron el proceso de desarrollo y permitieron a los participantes aumentar las capacidades de desarrollo de sistemas web.

Los integrantes del PUEG utilizan la herramienta diariamente y están ya interesados en nuevas funcionalidades que serán motivo de trabajo futuro.

6 Bibliografía

(Date,1986) C. J. Date, “Introducción a los sistemas de bases de datos relacionales”, versión en español de Jaime Malpica, Addison-Wesley Iberoamericana 1986.

(TBD, 2007) “Bases de datos, tipos de bases de datos” http://es.wikipedia.org/wiki/Base_de_datos, Último acceso: 9 de agosto de 2007.

(Millar,1997) Millar David,“Desarrollo multimedia para internet la mejor fuente de información para la implementación multimedia en el web”, traducción Parra Pérez Beatriz, Madrid Anaya-Multimedia 1997.

(Mendelzon,2000) Mendelzon Alberto, Mendelzon Ale, “Introducción a las bases de datos relacionales”, Buenos Aires, Prentice Hall 2000.

(VBD,2007) “Ventajas del uso de las bases de datos”,

http://www.monografias.com/trabajos11/basda/basda.shtml, último acceso 28 de agosto de 2007.

(Pressman,5) Pressman Roger S., “Ingeniería del software, un enfoque práctico”, traducción de Ojeda Rafael, Morales Isabel, Yagûe Virgilio, Sánchez Salvador, Mc Graw Hill 5ª edición.

(Gardarin,1990) Gardarin Georges “Bases de Datos”, traducido por García F. Mateo, Madrid, Paraninfo S.A., 2a edición 1990.

(Kort,1987) Kort Henry F., Silberschatz Abraham, “Fundamentos de bases de datos”, traducción Escalona García Roberto, Mc Graw-Hill 1987.

(Ceballos, 2004) Ceballos Sierra Francisco Javier, “Programación orientada a objetos con C++”, México, Alfaomega 2004.

(Greiff, 1994) Greiff W. R. “Paradigma vs Metodología, El Caso de la POO (Parte II).” *Soluciones Avanzadas*.

(Booch,1996) Booch Grady,“Análisis y diseño orientado a objetos con aplicaciones”, versión española de Cueva Lovelle Juan Manuel, Cernuda del Río Agustín; Argentina, Ed. Addison Wesley 1996.

(VPOO, 2007) Ventajas de la Programación orientada a objetos sobre otras formas de programación, último acceso 12 de septiembre de 2007. http://www.elrinconcito.com/articulos/POO/POO.htm

(UML, 2007) Lenguaje Unificado Modelado

http://www.clikear.com/manuales/uml/index.asp, último acceso 12 de septiembre de 2007.

(Soft/Est UML, 2007) “Software gratuito para modelar en UML”

http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado#Software_gratuito_para_modelado_en_UML, último acceso 12 de septiembre de 2007.

(Fairley,1994) Fairley Richard E., “Ingeniería de Software”, traducción Sánchez Aguilar Antonio, Flores Suárez Pedro Luis, México, Mc Graw-Hill 1994.

(Lewis, 1994) Lewis G."What is Software Engineering?" DataPro (4015). Feb 1994.

(Cota,1994) Cota A."Ingeniería de Software, soluciones avanzadas”. Julio de 1994

(IS, 2007) “Ingeniería del Software” , http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software, último acceso 12 de septiembre de 2007.

(IIS, 2007), “Introducción a la Ingeniería del Software”, http://www.um.es/docencia/barzana/IAGP/IAGP2-Ingenieria-software-introduccion.html, último acceso 12 de septiembre de 2007.

(Jacobson,1992) Jacobson, I. “Object-Oriented Software Engineering; A Use Case Driven Aproach”, ACM Press. Adison-Wesley Publishing. Co. U.S.A.

(MOO, 2008)”Modelo orientado a objetos” ,http://es.wikipedia.org/wiki/Base_de_datos, último acceso 15 de enero de 2008.

(SPHP, 2008)”Sesiones en php”, ”http://www.webestilo.com/php/php12a.phtml, último acceso 15 de enero de 2008.

(CERN, 2008) “ Organización Europea para la Investigación Nuclear“, http://es.wikipedia.org/wiki/CERN

(LEMAY, 1998) Lemay Laura, “Aprendiendo HTML para Web en una semana”, traducido por Gutiérrez Jorge Luis, México 3a edición, Prentice Hall 1998.

(LÓPEZ, 2000) López González Angel, Novo López Alejandro, “Protocolos de Internet, Diseño e implementación en sistemas UNIX”, México Alfaomega,S.A. 2000.