Universidad de las "Ciencias Informáticas"
Facultad **3**

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Análisis y Diseño del Módulo Entrada de Datos para el Software de Automatización del Convenio Integral de Cooperación Cuba-Venezuela.

Autores:

Reynier Peña Ramírez

Michael Raul Galiano Sierra

Tutor:

Ing. Juan Carlos Montané Izaguirre

Ciudad de La Habana, Cuba

Mayo, 2008

Un hombre inteligente es aquel que sabe ser tan inteligente como para contratar gente más inteligente que él

John F. Kennedy

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la infraestructura productiva de la Universidad de las Ciencias Informáticas; así como a la facultad 3 de dicho centro de hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Michael Raúl Galiano Sierra

_____

(Autor)

Reynier Peña Ramírez

_____

(Autor)

Juan Carlos Montané Izaguirre

_____

(Tutor)

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado **Análisis y Diseño del Módulo Entrada de Datos para el Software de Automatización del Convenio Integral de Cooperación Cuba-Venezuela** , fue realizado en La Universidad de las Ciencias Informáticas (UCI). Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

 Totalmente

 Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____

_

_____

_

_____

_

_____

_

_____

_

_____

_

_____

Como resultado de la implantación de este trabajo se reportará un efecto económico considerable. Para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

Agradecimientos De Reynier

A todo el que confió en mí.

A todo el que me ayudó, en algún momento.

A los que me tendieron su mano, y a los que me la negaron también.

A todos, gracias.

Agradecimientos De Michael

A todos los que nunca perdieron la fe en mí.

A todo aquel que me brindo apoyo en mi vida.

A los que me dieron la espalda, porque también me dieron fuerza.

A todos, gracias.

Dedicatoria De Reynier

Primero que todo a Dios por regalarme la vida y darme la posibilidad hoy de dedicar este trabajo.

A mis padres, mis hermanos, mi cuñado, especialmente mis sobrinas por ser las persona que más quiero; por la vida, el amor, la educación que me han dado.

Por mostrarme el camino correcto, por darme el apoyo y la seguridad de que podía lograrlo.

A ustedes le debo todo lo que soy y mucho más, por tanta paciencia y dedicación, este es mi regalo.

A mi abuela Ízamela y mi abuelo Gastón, por darme tantos consejos y su sabiduría.

A mi Tía Vildemia, por ser tan buena y comportarse como mi segunda madre defendiéndome en todos los momentos, una de las personas que más he querido.

A mi Tía Yenny y mi Tío Pepe por quererme igual que a un hijo.

A todos mis primos

A toda mí familia, por su apoyo y confianza.

A mi novia y mis amigos

Gracias a ustedes, este día ha llegado, a todos les dedico este trabajo.

Dedicatoria De Michael

A nuestro señor Dios por darme la posibilidad de despertar cada día para regalar mi sonrisa.

Especialmente a Zenaida por ser mi bastón, mi guía y la corona de mi cabeza, te amo abuela.

Con todo el amor del mundo a mi abuelita Teresa, por velar desde el cielo cada paso que doy en esta vida y por todo el amor que me diste, te extraño.

A ti mamá, por tu amor, por confiar ciegamente en mi; por todos los besos que me has regalado; siempre serás la razón de mi existencia.

A ti viejo, por tus consejos, tu mano, tu prestigio y por regalarme tu carácter y tu fuerza el día en que vine a este mundo.

Dirigido a mí tío Severino, todo el esfuerzo que he realizado para llegar hasta aquí, por regalarme tantos momentos de felicidad y tanto amor, no te imaginas cuanto te quiero.

A mis hermanitas y mis primitos porque son dueños y herederos del fruto de tanto esfuerzo.

A Cesarito, Hugo, Alexey "El Chicharo", Kiosmeri, Deibel, Robertico, Rodney, Abel "El Feo", Enrique "Kiki", Yainer, y Dalver, Jonathan por ser mis hermanos y a todos mis amigos.

A Rey por darme tu voluntad, tu confianza, tu amistad, por tu batallar de cada a mi lado; que el Señor te regale todo lo que siempre has anhelado.

A mi tía Rosa y a mi familia por su apoyo y confianza.

A cada persona que durante mi vida ha puesto un grano de arena para contribuir a mi futuro.

Hacia ustedes va dirigido este trabajo y todo el fruto que pueda generar, gracias.

RESUMEN

El presente trabajo desarrollado en La Universidad de las Ciencias Informáticas (UCI) recoge los resultados de la investigación realizada para el desarrollo posterior de un sistema informático, donde queda elaborado el análisis y diseño de las principales actividades que componen el Módulo Entrada de Datos, las cuales ayudan a eliminar los problemas existentes referentes al control, toma de acuerdos y mantenimiento de la información sobre los proyectos desarrollados en el plano de las MIXTAS del Convenio Cuba – Venezuela, que actualmente, no cuenta con un soporte informático. Quedaron plasmados los artefactos más importantes de los flujos de trabajo Modelación del Negocio, Requerimientos y Análisis y Diseño. Se hizo un estudio de tecnologías, y técnicas de desarrollo del software en algunas de las metodologías, así como la aplicación de patrones de arquitectura y diseño garantizando la escalabilidad y entendimiento del sistema por parte de los desarrolladores. Esperando cumplir además con la política de migración en el desarrollo de aplicaciones sobre plataforma libre adoptada para los países inmiscuidos en el tratado del ALBA con el fin de disminuir los costos en licencias.

PALABRAS CLAVE: **INGENIERÍA DEL SOFTWARE, INGENIERÍA DE REQUISITOS (IR), CLIENTES Y/O USUARIOS, SOFTWARE DE GESTIÓN (SWG).**

INDICE

INTRODUCCIÓN..... 1
Antecedentes y problemática existente.....1
Situación problemática.....1
Definición del problema.....2
Objeto de estudio.....2
Campo de acción.....2
Hipótesis.....2
Variables.....2
Objetivos.....3
Tareas de la investigación.....3
Métodos y técnicas de investigación a utilizar.....3
Métodos Teóricos.....3
Métodos Empíricos.....4
Método Sistémico.....4
Revisión Bibliográfica.....4
Resultados esperados.....4
Estructura de la tesis.....5
CAPITULO 1. FUNDAMENTACIÓN TEÓRICA..... 6
1.1 Introducción.....6
1.2 La gestión de la información.....6
1.3 Conceptos de proyecto.....7
1.4 La gestión de proyectos.....7
1.4.1 Dimensión técnica.....8
1.4.2 Dimensión humana.....8
1.4.3 Variable gestión.....8
1.5 Ventajas de un sistema de gestión de proyectos en-línea.....9
1.6 Herramientas de gestión, control y seguimiento de proyectos en-línea.....10
1.6.1 B-KIN Project.....10
1.6.2 KMKey Project.....11
1.7 Selección de herramientas para desarrollar el sistema.....11
1.7.1 Herramientas CASE para diseño y construcción de sistemas.....12
1.7.1.1 Rational Rose Enterprise Edition 2003.....12
1.7.1.2 Visual Paradigm Para UML.....12
1.8 Plataformas para desarrollo de software.....13
1.8.1 Java 2, Enterprise Edition.....13
1.8.2 Microsoft.NET.....14
1.9 Tendencias de los lenguajes de programación de sistemas.....17
1.9.1 C++.....17
1.9.2 Java.....18
1.9.3 C#.....18
1.10 Frameworks de acceso a datos.....19
1.10.1 Hibernate.....19
1.10.2 TierDeveloper.....19
1.11 Sistema de gestión de bases de datos (SGBD).....19
1.11.1 Postgres.....20
1.11.1.1 Breve historia de Postgres.....21
1.11.1.2 Algunas de sus principales características.....21
1.11.1.3 Funciones.....22
1.11.2 Microsoft SQL Server.....23
1.11.3 Oracle.....24

1.12 Proceso de desarrollo de software 24
1.12.1 Metodología de desarrollo de software 26
1.12.2 Selección de la metodología de desarrollo de software 26
1.12.3 Lenguaje de Unificado y Modelado (UML) 27
1.12.4 Modelo 27
1.12.5 Rational Unified Process (RUP) 28
1.12.6 Extreme Programing (XP) 29
1.12.7 Microsoft Solution Framework (MSF) 31
1.13 El análisis 32
1.13.1 Principios del análisis 32
1.14 Gestión de riesgos 33
1.14.1 Riesgo 33
1.14.2 Fases 34
1.15 Ingeniería de requerimientos 34
1.15.1 Actividades 35
1.15.2 Técnicas principales 36
1.15.2.1 Entrevistas 36
1.15.2.2 Talleres 36
1.15.2.3 Forma de contrato 37
1.15.2.4 Objetivos mesurables 37
1.15.2.5 Prototipos 37
1.15.2.6 Casos de uso 37
1.15.3 Especificación de requisitos del software 38
1.15.4 Identificación de las personas involucradas 38
1.15.5 Problemas 38
1.15.5.1 Relacionados con las personas involucradas 38
1.15.5.2 Relacionados con los analistas 39
1.15.5.3 Relacionados con los desarrolladores 39
1.16 Diseño del software 40
1.16.1 Principios del diseño 40
1.16.2 Patrones de diseño 42
1.16.2.1 Objetivos de los patrones 42
1.16.2.2 Categorías de patrones 43
1.16.2.3 Estructuras o plantillas de patrones 43
1.16.2.4 Relación de principales patrones GoF 44
1.17 Métricas 46
1.17.1 Métricas del diseño arquitectónico 46
1.17.2 Métricas orientadas a clase 47
1.17.3 Métricas de diseño a nivel de componentes 47
1.18 Conclusiones 48
CAPITULO 2. CARACTERÍSTISTICAS DEL SITEMA 49
2.1 Introducción 49
2.2 Objetivo básico. Módulo Entrada de Datos 49
2.2.1 Objetivos estratégicos generales del Módulo 49
2.2.2 Objetivos estratégicos específicos del Módulo 49
2.3 Selección del modelado 50
2.3.1 Definición de las entidades y los conceptos principales 50
2.3.1.1 Estructura organizativa 50
2.3.1.2 Diagrama: Modelo del Dominio. 54
2.4 Reglas de negocio 55
2.4.1 Reglas del Módulo 55
2.5 Especificación de requerimientos de software 56

2.5.1 Requerimientos funcionales 56
2.5.2 Requerimientos no funcionales 58
2.5.2.1 Orientados al usuario: 58
2.5.2.2 Orientados al desarrollador: 59
2.5.2.3 Requisitos para los clientes: 59
2.5.2.4 Requisitos para servidores: 59
2.6 Modelo del sistema 60
2.6.1 Descripción de Actores del Sistema 61
2.6.2 Diagrama de Casos de Uso del Sistema 62
2.6.3 Gestionar Ficha Resumen 63
2.6.4 Gestionar Estado de Proyecto. 68
2.6.5 Gestionar Ejecución Financiera. 71
2.6.6 Gestionar Ejecución Física 74
2.7 Conclusiones 78
CAPITULO 3. ANÁLISIS Y DISEÑO 79
3.1 Introducción 79
3.2 Modelo de análisis 79
3.2.1 Diagrama de clases de análisis 79
3.2.1.1 DCA del CUS Gestionar Ficha Resumen 80
3.2.1.2 DCA del CUS Gestionar Estado de Proyecto 80
3.2.1.3 DCA del CUS Gestionar Ejecución Financiera 81
3.2.1.4 DCA del CUS Gestionar Ejecución Física 81
3.2.2 Diagrama de interacción 82
3.2.2.1 DI del CUS Gestionar Ficha Resumen (Aprobar) ....**¡Error! Marcador no definido.**
3.2.2.2 DI del CUS Gestionar Ficha Resumen (Modificar) ....**¡Error! Marcador no definido.**
3.2.2.3 DI del CUS Gestionar Ficha Resumen (Rechazar) ....**¡Error! Marcador no definido.**
3.2.2.4 DI del CUS Gestionar Estado de Proyecto (Aprobar) ....**¡Error! Marcador no definido.**
3.2.2.5 DI del CUS Gestionar Estado de Proyecto (Modificar) ....**¡Error! Marcador no definido.**
3.2.2.6 DI del CUS Gestionar Estado de Proyecto (Rechazar) ....**¡Error! Marcador no definido.**
3.2.2.7 DI del CUS Gestionar Ejecución Física (Aprobar) ....**¡Error! Marcador no definido.**
3.2.2.8 DI del CUS Gestionar Ejecución Física (Modificar) ....**¡Error! Marcador no definido.**
3.2.2.9 DI del CUS Gestionar Ejecución Física (Rechazar) ....**¡Error! Marcador no definido.**
3.2.2.10 DI del CUS Gestionar Ejecución Financiera (Aprobar) ....**¡Error! Marcador no definido.**
3.2.2.11 DI del CUS Gestionar Ejecución Financiera (Modificar) ....**¡Error! Marcador no definido.**
3.2.2.12 DI del CUS Gestionar Ejecución Financiera (Rechazar) ....**¡Error! Marcador no definido.**
3.3 Modelo de diseño 82
3.3.1 Diagrama de clases de diseño 82
3.3.2 Diagramas de clases de diseño utilizando extensiones UML para Web 83
3.3.4 DCD del CUS Gestionar Ficha Resumen 84
3.3.5 DCD del CUS Gestionar Estado de Proyecto 85
3.3.6 DCD del CUS Gestionar Ejecución Financiera 86
3.3.7 DCD del CUS Gestionar Ejecución Física 87
3.4 Patrones 88
3.5 Descripción de la arquitectura propuesta 89
3.5.1 Patrones aplicados 91
3.6 Diseño de la Base de Datos 92
3.7 Conclusiones 93
CAPITULO 4. ANÁLISIS DE LA FACTIBILIDAD. 94
4.1 Introducción 94
4.2 Factibilidad del análisis 94
4.3 Estimación basada en Casos de Uso 95

4.3.1 UUCP (Cálculo de puntos de Casos de Uso sin ajustar) 95
4.3.1.1 UAW (Factor de peso de los actores sin ajustar) 95
4.3.1.2 UUCW (Factor de peso de los Casos de Uso sin ajustar) 96
4.3.2 Cálculo de UCP (Puntos de Casos de Uso ajustados) 97
4.3.2.1 Factor de TCF (Complejidad técnica) 97
4.3.2.2 Cálculo del EF (Factor de ambiente) 99
4.3.3 (E) Cálculo del esfuerzo en horas – hombres 100
4.3.3.1 Conversión de los puntos de Casos de Uso ajustados a esfuerzo de desarrollo 100
4.4 Factibilidad del diseño 102
4.4.1 Métricas de diseño arquitectónico 102
Resultado 104
4.4.2 Resultado de las métricas orientadas a clases 104
Métricas propuestas por Lorenz y Kidd. Aplicación al modelo. 104
Resultado 105
4.4.3 La serie de métricas CK. Aplicación al modelo. 106
4.4.3.1 Árbol de profundidad de herencia (APH) 106
4.5 Beneficios tangibles e intangibles 106
4.6 Análisis de costos y beneficios 107
4.7 Conclusiones 107
CONCLUSIONES GENERALES 109
RECOMENDACIONES 110
BIBLIOGRAFÍA 111
GLOSARIO DE TÉRMINOS Y SIGLAS .... ¡ERROR! MARCADOR NO DEFINIDO.

INTRODUCCIÓN

Antecedentes y problemática existente

En el marco de la Alternativa Bolivariana para las América (ALBA), la Republica de Cuba y la Republica Bolivariana de Venezuela con la intención de fortalecer los tradicionales lazos de amistad, conscientes de su interés común por promover y fomentar el progreso social, económico y la integración de América Latina, se comprometieron a elaborar proyectos de cooperación; en los cuales se están realizando numerosas inversiones millonarias, que necesitan ser controladas. Para la ejecución de estos proyectos se considera la participación de organismos y entidades de los sectores públicos y privados de ambos países; los mismos se encuentran divididos en 4 niveles jerárquicos: Directivos de Gobierno, Secretarias Técnicas, Ministerios, y Entes Ejecutores. Los proyectos tienen un alcance nacional o de integración regional respondiendo a las prioridades contenidas en sus respectivos planes de desarrollo.

Los Gobiernos de Cuba y Venezuela trabajan sobre principios de evitar la corrupción y malversación de los bienes, por tanto como medida para fragmentar esta barrera y la necesidad que existe para las instancias superiores de conocer el estado en tiempo real del los proyecto y su evolución; trae consigo que ambos estados se hayan fijado la meta de crear una herramienta informática capaz de estabilizar y mantener un control sobre todos los flujos de gestión y negocios. Dicha misión está encomendada a la estructura productiva de La Universidad de las Ciencias Informáticas.

Se necesita la implementación de un sistema informático para la gestión de la información, control estadístico, seguimiento en los procesos físicos y financieros, ficha resumen y estado del los proyectos unido a una gestión confiable de la puesta de acuerdo entre los Entes Ejecutores.

Situación problemática

Los cronogramas físico-financieros y el control de los proyectos durante su desarrollo no están estandarizados y controlados por un proceso único. Cuando un Ente Ejecutor pretende realizar cambios en la información de un proyecto es necesaria la aprobación por una y otra parte (Cuba-Venezuela). Los Entes Ejecutores no se encuentran en un mismo territorio, tornándose insostenible la puesta de acuerdo de modo personal, auxiliándose de vías telefónicas y apoyándose en la mensajería electrónica. Estos

mecanismos no constituyen un método efectivo para la gestión de información requerida ante un cambio repentino sobre el cronograma de ejecución física de los proyectos; puesto que las tarifas telefónicas son muy caras y dificultan el entendimiento al no tener una visión sobre lo que se está pactando, y el correo electrónico limita el envío y recibo de grande volúmenes de información. La unión de todos estos factores trae como consecuencia un exceso en el presupuesto total asignado a un proyecto, una prolongación del tiempo con respecto a la fecha de terminación y un limitado seguimiento a los Entes Ejecutores y sus proyectos durante la ejecución, por parte de los Ministerios y a su vez por las Secretarias Técnicas. Dada esta situación problemática se hace necesario desarrollar el análisis y diseño (para dar paso a la implementación) de un sistema informático para la gestión de la información, control estadístico, seguimiento en los procesos físicos y financieros, ficha resumen y estado del los proyectos unido a una gestión confiable de la puesta de acuerdo entre los Entes Ejecutores.

Definición del problema

¿Cómo realizar el análisis y el diseño del Módulo Entrada de Datos para la gestión eficiente de los proyectos del Convenio Cuba – Venezuela, lo suficientemente flexible para que pueda ser implementado sin ambigüedades?

Objeto de estudio

Proceso de desarrollo de software para sistemas de gestión de proyectos.

Campo de acción

El análisis y el diseño del módulo de Entrada de Datos para la gestión y control de los proyectos del Convenio Cuba-Venezuela.

Hipótesis

Si se desarrolla el análisis y el diseño del módulo Entrada de Datos para el sistema de automatización del Convenio Cuba-Venezuela, entonces se facilitará una eficiente implementación de dicho módulo.

Variables

Desarrollar el análisis y el diseño del módulo Entrada de Datos. **(Independiente)**

Facilitar una eficiente implementación del módulo. **(Dependiente)**

Objetivos

El objetivo general es desarrollar el análisis y el diseño del módulo Entrada de Datos para El Sistema de Automatización del Convenio Cuba – Venezuela.

Partiendo de este objetivo general se derivan los siguientes objetivos específicos:

- Establecer el ámbito del proyecto y sus límites.
- Generar los artefactos necesarios.
- Encontrar los casos de uso críticos del módulo y escenarios básicos que definen su funcionalidad.
- Analizar los casos de usos del sistema, así como las reglas del negocio a tener en cuenta para la construcción del diseño.
- Aplicación de la propuesta de diseño adoptada y de los patrones.
- Mostrar una arquitectura candidata.
- Evaluar la calidad del diseño obtenido a través de la aplicación de métricas.

Tareas de la investigación

1. Estudio de los principios económicos que rigen los procesos de negocios en los proyectos del Convenio Cuba-Venezuela.
2. Estudio del estado del arte de los principales software en línea que existen para gestión de proyectos.
3. Estudio de los principios del análisis para la validación de requisitos del software.
4. Estudio del estado del arte del las principales tendencia del diseño software.
5. Estudio de las métricas correspondientes para la validación del análisis y el diseño.

Métodos y técnicas de investigación a utilizar

Métodos Teóricos

Histórico - Lógico: Para analizar la trayectoria y evolución de la metodología de desarrollo de software y demás herramientas que se utilizan durante el trabajo.

Método de la modelación: Para la creación de modelos como abstracciones del la actualidad del módulo de Entrada de Datos con el objetivo de explicar la realidad.

Métodos Empíricos

Entrevista: Para recopilar las características más importantes de los procesos del negocio al cual responde el sistema.

Método Sistémico

Para relacionar hechos aparentemente aislados del módulo Entrada de Datos y unificar los diversos elementos que conforman parte del mismo. Determinando por un lado la estructura del módulo y por otro su dinámica.

Revisión Bibliográfica

Su objetivo fundamental es estudiar un conjunto de fuentes de información referidas al tema, así como libros, artículos, revistas, publicaciones, boletines y una especie de materiales escritos y digitalizados, que son de gran utilidad para documentar la base teórica del trabajo a desarrollar, debidamente referenciadas para futuras consultas sobre el tema.

Resultados esperados

- Especificación de requisitos.
- Especificación de los casos de uso.
- Definición del prototipo de interfaz.
- Modelo de análisis.
    - ✓ Diagrama de clases del análisis.
    - ✓ Realización de casos de usos del análisis.

- Modelo de Diseño
    - ✓ Diagrama de clases del diseño.
    - ✓ Realización de casos de uso de diseño.
- Análisis y resultado de las métricas aplicadas.

Estructura de la tesis

El presente trabajo, estructurado en 4 capítulos, resume la siguiente información:

Capítulo 1. Fundamentación Teórica: Se analizan los conceptos y evolución de la gestión de proyectos. Se profundiza en la descripción del objeto de estudio, sistemas existentes vinculados al mismo, tendencias, ingeniería de requerimientos, metodologías y tecnologías actuales seleccionadas a emplear en el desarrollo de la propuesta y por qué su utilización.

Capítulo 2. Características del Sistema: Se plantea el modelo de dominio teniendo en cuenta los conceptos y clases principales que conforman el contexto del sistema, se describen las características del mismo, reglas de negocio y los requisitos con los que debe cumplir, obteniendo como artefacto fundamental el diagrama de casos de uso del sistema.

Capítulo 3. Análisis y Diseño: Modelar los artefactos necesarios que contribuyen a la implementación a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), orientado al desarrollo de software y al modelado de procesos de negocio.

Capítulo 4. Análisis de La Factibilidad: Se muestran los resultados obtenidos a partir de la aplicación de métodos y métricas correspondientes, una vez finalizado el análisis y diseño. Con el objetivo de medir la factibilidad del sistema, analizando los diferentes criterios que influyen en el cálculo del esfuerzo, tiempo de desarrollo, costo del proyecto y calidad del diseño.

CAPITULO 1

CAPITULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se analizan algunos conceptos y evolución de la gestión de la información como herramienta fundamental para el desarrollo y seguimiento de los proyectos. Se da una descripción de los principales conceptos asociados al dominio del problema. Se hace alusión también al surgimiento de aplicaciones que en la actualidad se dedican a la gestión y control de proyectos. Así como un acercamiento a algunas de las principales metodologías de desarrollo, tendencias, roles desempeñados y tecnologías sobre las que se apoyará la propuesta de solución actual para este entorno de negocio, y su evolución.

1.2 La gestión de la información

En los nuevos modelos de negocio, la gestión de la información adquiere importancia estratégica. Las tendencias observadas en la práctica son: la evolución hacia la denominada gestión de contenidos, que comprendería la gestión de documentos y datos tanto internos como externos; la aceptación definitiva de algunos documentos electrónicos en las organizaciones como forma válida de documento; la necesidad creciente de gestionar electrónicamente información no estructurada en bases de datos; el reconocimiento de la informática como una herramienta y no como base de la gestión de la información; la cada vez menos importante gestión de los soportes a favor de la accesibilidad de los contenidos y; por último, la previsión de la gestión de la información electrónica a medio-largo plazo.

Por otra parte la voluntad de conservar de forma permanente los documentos digitales ha estado presente desde los primeros tiempos de la revolución informática. Sin embargo, han sido pocas las experiencias de crear archivos digitales, debido principalmente a la dificultad para encontrar soluciones técnicas a los problemas de obsolescencia, al alto coste económico y a la ausencia de instituciones que asuman esta responsabilidad, y propone la solución Software como la herramienta fundamental que permite definir una

política de conservación de la información de documentos en forma digitales como parte fundamental de toda empresa para tener un alto nivel de competitividad y posibilidades de desarrollo. (Huidobro, 2001)

1.3 Conceptos de proyecto

Un proyecto puede ser descrito de varias formas, a continuación se presentan algunas definiciones de los principales organismos y autores en materia de administración de proyectos:

"Es un conjunto de esfuerzos temporales, dirigidos a generar un producto o servicio único". (Project Management Institute, 2000)

"Un proyecto es un esfuerzo emprendido para producir los resultados esperados por la parte que lo solicita". (Eisner, 2000)

"Es un conjunto único de actividades interrelacionadas con tiempos de inicio y fin definidos, diseñado para alcanzar un objetivo común". (Nacional Competency Standards for Project Management, 1995)

"Es un proceso único, que consta de un conjunto de actividades coordinadas y controladas con fechas de inicio y fin, emprendidas para alcanzar un objetivo, conforme a requerimientos específicos, incluyendo restricciones de tiempo, costo y recursos". (Gonzalez, 2006)

Estas definiciones hacen referencia a dos características esenciales en cualquier proyecto para que se considere como tal; que genere un producto o servicio "único" y que sean "temporales".

1.4 La gestión de proyectos

Ha existido desde tiempos muy antiguos, históricamente relacionada con proyectos de ingeniería de construcción de obras civiles (como los proyectos de ingeniería hidráulica en Mesopotamia, donde entraban en juego la logística o la creación de equipos de trabajo, con sus categorías profesionales definidas, o la cultura ingenieril desarrollada por el Imperio Romano, donde aparece el control de costes y tiempos y la aplicación de soluciones normalizadas, como por ejemplo en la construcción de una calzada), y en "campañas militares", donde también entran en juego muchos elementos de gestión (identificación de objetivos, gestión de recursos humanos, logística, identificación de riesgos, financiación, etc.). Pero es a

partir de La Segunda Guerra Mundial cuando el avance de estas técnicas desde el punto de vista profesional ha transformado la administración por proyectos en una disciplina de investigación.

Completar con éxito el Proyecto significa cumplir con los objetivos dentro de las especificaciones técnicas, de costo y de plazo de terminación. A un conjunto de proyectos orientados a un objetivo superior se denomina PROGRAMA, y un conjunto de programas constituye un PLAN, como corresponde generalmente a los grandes planes nacionales.

Todo proyecto tiene tres facetas o aspectos diferentes que son necesarios armonizar para la consecución del resultado deseado:

1.4.1 Dimensión técnica

Es necesario aplicar los conocimientos específicos de cada área de trabajo, cumpliendo con una forma de trabajar y unos requisitos (el "know how") que cada profesión impone. Es de sentido común que es necesario disponer de los conocimientos adecuados para resolver el problema en cuestión o realizar la obra encomendada. Pero la importancia de esta faceta técnica no debe eclipsar el resto de aspectos que intervienen en la consecución de un proyecto, y que otorgan a esta actividad de una trascendencia y complejidad mayores

1.4.2 Dimensión humana

Un proyecto es un complejo entramado de relaciones personales, donde se dan cita un gran número de intereses a veces contrapuestos. A las inevitables diferencias que surgen por ejemplo entre el jefe de proyecto y cliente o proveedores, hay que reseñar las disputas internas a la organización que surgen a la hora de repartir los recursos de que se dispone, pues son varios los proyectos que se pueden estar llevando a cabo paralelamente en dicha organización.

1.4.3 Variable gestión

Con este término, se hace referencia a algo que a veces se menosprecia porque no es tan espectacular o visible como otros elementos pero que es el catalizador que permite que el resto de los elementos se comporten adecuadamente. De gestionar bien o mal depende en gran medida el éxito o no de la operación. (Project Management Institute, 2000)

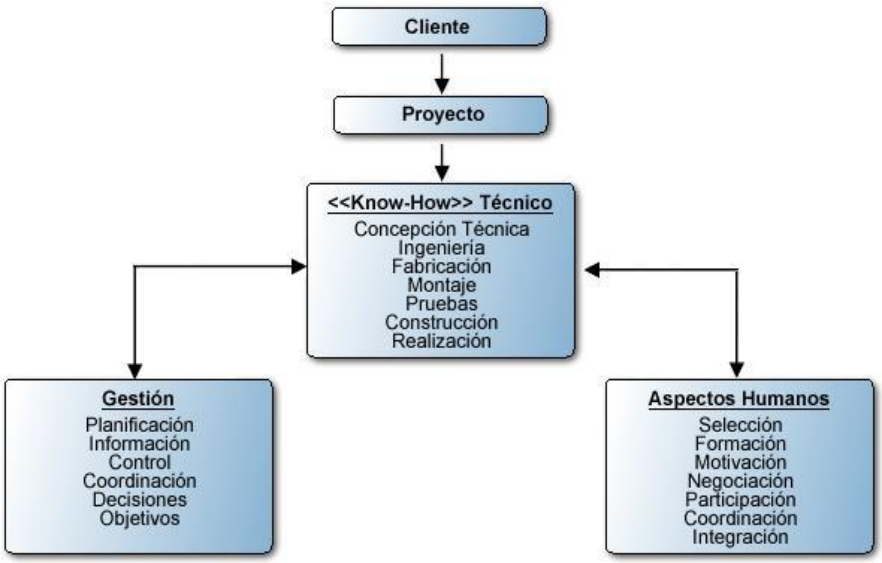


Figura 1. Diagrama de Gestión de Proyecto.

1.5 Ventajas de un sistema de gestión de proyectos en-línea

La decisión de realizar el software de gestión en-línea está justificada con las ventajas que se enuncian a continuación:

- **Es en-línea.** Perfecto para entornos distribuidos, en los que varias personas necesitan acceder a la información en tiempo real. Con el servicio que está disponible donde quieras y cuando quieras, sin instalaciones ni grandes desembolsos.
- **Es flexible**. Fácil adaptación a la entidad, independientemente de la localización ubicación y delegaciones, permitiendo que usuarios que lo utilizan desde diferentes lugares puedan trabajar juntos.
- **Es multi-plataforma.** Posibilidad de cambiar de PC, sistema operativo, etc. cuando se desee porque los datos están en línea. Tan sólo se necesita una conexión a Internet.
- **Es seguro.** se responsabiliza del alojamiento de los proyectos, garantizando la seguridad y confidencialidad de la información.
- **Es multi-proyecto.** Agrupa los proyectos por entes o grupos para poder analizarlos fácilmente.

- **Es colaborativo.** Asegura la participación mediante secciones de documentación para subir y compartir documentos dentro de cada proyecto.

1.6 Herramientas de gestión, control y seguimiento de proyectos en-línea

Diversos sistemas como el propuesto en este trabajo muestran continuamente su cara en Internet, aunque con disímiles características y funcionalidades, actualmente entre los más significativos que utilizan las grandes compañías y empresas dedicadas a La Atención Tecnológica y a La Estrategia Competitiva por seguir sus proyectos se encuentran:

Figura 2. B-KIN Logo.

1.6.1 B-KIN Project

Controla proyectos y tareas. Gestiona su avance, plazos, costes, esfuerzos, recursos (grupos de personas, perfiles y personas), y consulta los informes que necesites sobre la gestión de proyectos. Ofrece la información que necesites sobre cada elemento. Comparte la información del software de gestión de proyectos en forma de documentos, informes, listados... aportando valor a los proyectos y tareas con la facilidad de un rápido acceso desde cualquier punto de la red de toda la información relevante. (B-KIN, 2007)

Figura 3. kmkey logo.

1.6.2 KMKey Project

Es un software de gestión de proyectos donde cualquier empresa de servicios puede disponer de toda la información necesaria para desarrollar su negocio, desde la oferta hasta la entrega del proyecto y su posterior mantenimiento. KMKey Project es un software especialmente indicado para llevar el control de proyectos de cualquier tipo: desarrollo de proyectos de ingeniería, gestión de despachos de arquitectura, planificación seguimiento y control de obras, proyectos en tecnologías de la información, gestión de consultorías, ingeniería medioambiental... son algunas de las funcionalidades que actualmente son trabajadas con KMKey Project. (KMKEY, 2007)

Como se ha podido observar, en el mundo existen varios sistemas para la gestión de proyectos que brindan los servicios de control y seguimiento de los mismos, la mayoría son software propietarios y poseen un elevado precio de venta, por tanto no pueden ser adquiridos, en nuestro caso unido a este inconveniente está el de no poseer los requisitos, ni las características necesarias para cumplir con los procesos que se desarrollan entre los países de Cuba y Venezuela para llevar a cabo la gestión de los proyectos en el marco de las MIXTAS durante el convenio del ALBA.

1.7 Selección de herramientas para desarrollar el sistema

Las herramientas constituyen un componente básico en el desarrollo de cualquier proyecto. Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. La selección del conjunto de herramientas constituye un punto clave en todo proceso de desarrollo y debe llevarse a cabo cuidadosamente, debido, a que una inadecuada elección de las mismas podría dificultar el cumplimiento de los objetivos del negocio o dominio en cuestión que se hayan establecido en la toma de acuerdos.

1.7.1 Herramientas CASE para diseño y construcción de sistemas

1.7.1.1 Rational Rose Enterprise Edition 2003

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

Rational Rose Enterprise Edition es una herramienta que está dentro del grupo de herramientas más técnicas debido a que se encarga de llevar a cabo tanto la automatización de los sistemas para la posterior generación de código (esto es, realización de los distintos diagramas y generación del código posterior), como para labores de ingeniería inversa (es decir, realización de los diagramas una vez conocido el código). (Rational-Rose, 2003)

1.7.1.2 Visual Paradigm Para UML

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Visual-Paradigm).

Algunas de sus características principales son:

1. Diagramas de Procesos de Negocio, Decisión, Actor de negocio, Documento.
2. Ingeniería inversa - código a modelo, código a diagrama.
3. Ingeniería inversa Java, C++, esquemas XML, XML, .NET exe/dll, CORBA IDL.
4. Generación de código - modelo a código, diagrama a código.

5. Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
6. Diagramas de flujo de datos.
7. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.

1.8 Plataformas para desarrollo de software

1.8.1 Java 2, Enterprise Edition

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación (parte de la Plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de "n" niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; no obstante sin un estándar de ISO o ECMA. Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel. (SUN MICROSISTEMS).
Ejemplos de herramientas de desarrollo Java de código abierto de terceras partes son:

- NetBeans IDE, un IDE basado en Java.
- La plataforma Eclipse, un IDE basado en Java.

- Expand, un plugin de Eclipse, para desarrollo rápido.
- Jedit, de código abierto, un IDE basado en Java.
- Apache Software Foundation Apache Ant, una herramienta de construcción automática.
- Apache Software Foundation Apache Maven, una herramienta de construcción automática y gestión de dependencias.
- JUnit, un framework para Pruebas de unidad automatizadas.
- Apache Software Foundation Apache Tomcat, es un contenedor web de Servlet/JSP.
- Jetty, un servidor web y un contenedor web Servlet/JSP.
- Struts, un framework para desarrollar aplicaciones web EE conforme al modelo MVC.
- OpenXava, un framework de código abierto para desarrollo fácil de aplicaciones de negocio J2EE.
- JDeveloper, un IDE basado en Java y desarrollado por Oracle.
- JBuilder, desarrollado por Borland.JBuilder.

1.8.2 Microsoft.NET

Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado. .Net podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones o como la misma plataforma las denomina, soluciones permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo. La norma que define el conjunto de funciones que debe implementar la librería de clases base (BCL por sus siglas en inglés) (incluido en ECMA-335, ISO/IEC 23271) Tal vez el más importante de los componentes de la plataforma, esta norma define un conjunto funcional mínimo que debe implementarse para que el marco de trabajo sea soportado por un sistema operativo. Aunque Microsoft implementó esta norma para su sistema operativo Windows, la publicación de la norma abre la posibilidad de que sea implementada para cualquier otro sistema operativo existente o futuro, permitiendo que las aplicaciones corran sobre la plataforma

independientemente del sistema operativo para el cual haya sido implementada. El Proyecto Mono emprendido por Ximian pretende realizar la implementación de la norma para varios sistemas operativos adicionales bajo el marco del software libre o código abierto.

Common Language Runtime (CLR)

Este es el lenguaje insignia de .NET Framework (marco de trabajo .NET) y pretende reunir las ventajas de lenguajes como C, C++ y Visual Basic en uno solo. El CLR es el verdadero núcleo del framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo (W2k y W2003).

Figura 4. CLR

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio, el MSIL (Microsoft Intermediate Lenguaje), similar al BYTECODE de Java. Para generarlo, el compilador se basa en la especificación CLS (Common Language Specification) que determina las reglas necesarias para crear el código MSIL compatible con el CLR. Para ejecutarse se necesita un segundo paso, un compilador JIT (Just-In-Time) es el que genera el código máquina real que se ejecuta en la plataforma del cliente. De esta forma se consigue con .NET independencia de la plataforma de hardware. La compilación JIT la realiza el CLR a medida que el programa invoca métodos. El código ejecutable obtenido se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.

Características del .Net

Es el encargado de proveer lo que se llama código administrado, es decir, un entorno que provee servicios automáticos al código que se ejecuta. Los servicios son variados:

- Cargador de clases: permite cargar en memoria las clases.
- Compilador MSIL a nativo: transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
- Administrador de código: coordina toda la operación de los distintos subsistemas del Common Lenguage Runtime.
- Recolector de basura: elimina de memoria objetos no utilizados.
- Motor de seguridad: administra la seguridad del código que se ejecuta.
- Motor de depuración: permite hacer un seguimiento de la ejecución del código aún cuando se utilicen lenguajes distintos.
- Verificador de tipos: controla que las variables de la aplicación usen el área de memoria que tienen asignado.
- Administrador de excepciones: maneja los errores que se producen durante la ejecución del código.
- Soporte de multiproceso (threads): permite ejecutar código en forma paralela.
- Empaquetador de COM: coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.
- Soporte de La Biblioteca de Clases Base: interfaz con las clases base del .NET Framework. Esto quiere decir que existen tipos de estructuras como es la de java y la .NET.

Debido a las ventajas que la disponibilidad de una plataforma de este tipo puede darle a las empresas de tecnología y al público en general, muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .NET, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris), el desarrollo de lenguajes de programación adicionales para la plataforma (ANSI C de La Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, entre otros) o la creación de bloques adicionales para la plataforma (como controles, componentes y bibliotecas de clases adicionales); siendo algunas de ellas software libre, distribuibles bajo la licencia GPL. Con esta plataforma Microsoft incursiona

de lleno en el campo de los Servicios Web y establece el XML como norma en el transporte de información en sus productos y lo promociona como tal en los sistemas desarrollados utilizando sus herramientas. (Microsoft)

Como resultado final del estudio realizado entre ambas plataformas, se decidió optar por la plataforma JEE adoptando todos sus beneficios; librerías, interfaces, frameworks y otros más que brindan una infraestructura para el desarrollo de arquitecturas empresariales, abarcando cada uno de los elementos como las transacciones, mensajería, conexiones a base de datos, interfaces de usuario, manejo de recursos y seguridad, además brinda la posibilidad de empezar con poco o ningún coste el desarrollo del sistema puesto que la implementación Java EE de Sun Microsystems puede ser descargada gratuitamente. Cuenta con un gran número de herramientas de código abierto disponible para extender la plataforma o para simplificar el desarrollo. Es una plataforma con experiencia y amplio soporte en el mundo permitiendo la construcción de aplicaciones que puedan ejecutarse en cualquier tipo de plataforma o sistema operativo. Como el objetivo principal del trabajo se centra en el marco de los proyectos del ALBA, se apoyará en plataformas que soporten código abierto.

1.9 Tendencias de los lenguajes de programación de sistemas

Las generaciones de los lenguajes de programación, han alcanzado dos tendencias fundamentales:

- El desplazamiento del centro de atención de la programación de pequeños sistemas hechos por una persona a los medianos y grandes sistemas, hechos por equipos de desarrollo de software. (Grady, 1996)
- La evolución de los lenguajes, un desplazamiento desde los lenguajes que dicen al computador qué hacer, o lenguajes imperativos, hacia lenguajes que describen las abstracciones clave en el dominio del problema (lenguajes declarativos). (Grady, 1996)

Entre los lenguajes más populares en la actualidad se encuentran:

1.9.1 C++

Es un lenguaje de programación, diseñado a principios de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C, sus principales características son el soporte para programación orientada a objetos, puede manipular directamente la memoria de la computadora, no

tienen muchas verificaciones automáticas, no da soporte para interfaces de implementación, tiene entornos de desarrollos (IDE) más comunes que dan soporte a este lenguaje. Visual C++ .NET de Microsoft, GCC para software libre, Borland C++ Builder de Borland.

1.9.2 Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es interpretado (usando normalmente un compilador JIT), por una máquina virtual Java. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos mucho más simple y elimina herramientas de bajo nivel como punteros, da soporte a interfaces de implementación, y posee un modelo de objetos mejor definido que C++. Es un Lenguaje de programación independiente de la plataforma, lo cual significa, que no importa el Sistema Operativo ni el Hardware que se utilice para correr estas aplicaciones.

1.9.3 C#

Es un lenguaje de programación dentro de la plataforma Microsoft .Net es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes Object Pascal más conocido por su IDE (Delphi). El 7 de noviembre de 2005 acabó la beta y salió la versión 2.0 del lenguaje que incluye mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables, interfaces de implementación. Ya existe la versión 3.0 de C# en fase de beta destacando los tipos implícitos y el LINQ (Language Integrated Query).

En la actualidad existen los siguientes compiladores para el lenguaje C#:

- Microsoft .NET Framework SDK incluye un compilador de C#, pero no un IDE.
- Microsoft Visual C#, IDE por excelencia de este lenguaje, versión 2002, 2003 y 2005.
- #Develop, es un IDE libre para C# bajo licencia LGPL, muy similar a Microsoft Visual C#.
- Mono, es una implementación GPL de todo el entorno .NET desarrollado por Novell. Como parte de esta implementación se incluye un compilador de C#.

- Delphi 2006, de Borland Software Corporation.
- DotGNU Portable.NET, de La Free Software Foundation.

1.10 Frameworks de acceso a datos

1.10.1 Hibernate

Realiza el mapeo entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. Es la solución ORM (Object-Relational Mapping) una de las más populares en el mundo Java. Constituye un motor de persistencia que implementa múltiples funcionalidades. El lenguaje de consultas es HQL (Hibernate Query Language), diseñado como una mínima extensión orientada a objetos de SQL, proporciona un puente elegante entre los mundos objetual y relacional. Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web. Es una clara implementación del patrón DAO, pues crea una capa separada que se ocupa del acceso a datos con total independencia del gestor y la base de datos. (JBOSS)

1.10.2 TierDeveloper

Es una herramienta de generación de código para mapeos de objetos en la plataforma .NET tanto para aplicaciones ASP.NET como para aplicaciones Windows Forms. TierDeveloper permite disminuir sustancialmente el tiempo de desarrollo en dependencia de de la complejidad de la aplicación que se vaya a desarrollar. Esta herramienta se integra a VS. Net 2003 y 2005.

Se seleccionó Hibernate por ser el más acorde con la plataforma java previamente escogida para el desarrollo del sistema y además estar implementado en ella.

1.11 Sistema de gestión de bases de datos (SGBD)

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de los sistemas de gestión de base de datos es el de manejar de forma clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje

de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa. (MIGUEL, 1997)

Dentro de la amplia gama de sistemas gestores de bases de datos existe MicrosoftSQL Server, PostgreSQL y Oracle de los cuales se brindan a continuación algunas de sus características principales por estar entre los gestores de mayor aceptación en la actualidad.

1.11.1 Postgres

Los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMSs) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este modelo será inadecuado para las aplicaciones futuras de procesado de datos. El modelo relacional sustituyó modelos previos en parte por su "simplicidad espartana". Sin embargo, como se ha mencionado, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema.

- Clases
- Herencia
- Tipos
- Funciones

Otras características aportan potencia y flexibilidad adicional:

- Restricciones (Constraints)
- Disparadores (triggers)
- Reglas (rules)
- Integridad transaccional

Estas características colocan a Postgres en la categoría de las Bases de Datos identificadas como objeto-relacionales. Nótese que éstas son diferentes de las referidas como orientadas a objetos, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales.

Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgres fue pionera.

1.11.1.1 Breve historia de Postgres

El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocido como PostgreSQL (y brevemente llamado Postgres95) está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python).

1.11.1.2 Algunas de sus principales características

Alta concurrencia

Mediante un sistema denominado Acceso concurrente multi-versión (MVCC), PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Otras características:

- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (foreign keys).
- Disparadores (triggers).
- vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

1.11.1.3 Funciones

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos brinda, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientación a objetos o la programación funcional.

Los disparadores son funciones enlazadas a operaciones sobre los datos.

Algunos de los lenguajes que se pueden usar son los siguientes:

- Un lenguaje propio llamado PL/PgSQL (similar al PL/SQL de oracle).
- C.
- C++.
- Gambas
- Java PL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.

PostgreSQL soporta funciones que retornan filas, donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta.

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como procedimientos almacenados. (manuales de ayuda)

1.11.2 Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Entre sus características figuran:

- Soporte de transacciones.
- Gran estabilidad.
- Gran seguridad.
- Escalabilidad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo accedan a la información.
- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle o Sybase. Es común desarrollar proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa una potente base de datos (Microsoft SQL Server) con un entorno de desarrollo cómodo y de alto rendimiento (VBA Access) a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows.

Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server contiene interfaces de acceso para la mayoría de las plataformas de desarrollo, incluyendo .NET.

Microsoft SQL Server, al contrario de su más cercana competencia, no es multiplataforma, ya que sólo está disponible en Sistemas Operativos de Microsoft.

1.11.3 Oracle

Oracle es un sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation. Se considera como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias). Otro aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

Como resultado del estudio realizado se selecciona PostgreSQL por ser un servidor de base de datos altamente sofisticado, con alto rendimiento, estable y capacitado para lidiar con grandes volúmenes de datos. Además de ser un producto Open Source, sin costos de licencia, posibilita la alternativa extremadamente atractiva para las empresas que buscan un ahorro significativo de costos en activos TI.

1.12 Proceso de desarrollo de software

Un sistema informático está compuesto por hardware y software. En cuanto al hardware, su producción se realiza sistemáticamente y la base de conocimiento para el desarrollo de dicha actividad está claramente definida. La fiabilidad del hardware es, en principio, equiparable a la de cualquier otra máquina construida

por el hombre. Sin embargo, respecto del software, su construcción y resultados han sido históricamente cuestionados debido a los problemas asociados, entre ellos se pueden destacar los siguientes.

- Los sistemas no responden a las expectativas de los usuarios.
- Los programas “fallan” con cierta frecuencia.
- Los costes del software son difíciles de prever y normalmente superan las estimaciones.
- La modificación del software es una tarea difícil y costosa.
- El software se suele presentar fuera del plazo establecido y con menos prestaciones de las consideradas inicialmente.
- Normalmente, es difícil cambiar de entorno hardware usando el mismo software.
- El aprovechamiento óptimo de los recursos (personas, tiempo, dinero, herramientas, etc.) no suele cumplirse.

Según el Centro Experimental de Ingeniería de Software (CEIS), el estudio de mercado realizado en 1996, concluyó que sólo un 16% de los proyectos de software son exitosos (terminan dentro de plazos y costos y cumplen los requerimientos acordados). Otro 53% sobrepasa costos y plazos y cumple parcialmente los requerimientos. El resto ni siquiera llega al término. Algunas deficiencias comunes en el desarrollo de software son:

- Escasa o tardía validación con el cliente.
- Inadecuada gestión de los requisitos.
- No existe medición del proceso ni registro de datos históricos.
- Estimaciones imprevistas de plazos y costos.
- Excesiva e irracional presión en los plazos.
- Escaso o deficiente control en el progreso del proceso de desarrollo.
- No se hace gestión de riesgos formalmente.
- No se realiza un proceso formal de pruebas.
- No se realizan revisiones técnicas formales e inspecciones de código.

Para lograr la productividad del software se necesita regir el proceso de desarrollo sobre metodologías que permitan obtener resultados de acuerdo a lo establecido con los clientes, por lo que a la hora de desarrollar un producto software uno de los aspectos más importante a tener en cuenta es la metodología a adoptar. (CEIS)

1.12.1 Metodología de desarrollo de software

Las metodologías de desarrollo de software abarcan todo el ciclo de vida del software, y se definen como "un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software, adoptando la misma se obtiene un producto software más predecible y permite ciertas características deseables como:

- Existencias de reglas bien definidas.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Utilización sobre un abanico amplio de proyectos.
- Fácil formación.
- Herramientas CASE (Computer Aided Software Engineering).
- Actividades que mejoren el proceso de desarrollo.
- Soporte al mantenimiento.
- Soporte de la reutilización de software.

De acuerdo a su definición de metodología, los procedimientos detallan consejos para elaborar una actividad; las técnicas serían la forma de ejecutar un procedimiento para obtener un resultado determinado; las herramientas software son las que hacen posible automatizar el proceso de desarrollo del software y la documentación es la que identifica el software que se está desarrollando.

1.12.2 Selección de la metodología de desarrollo de software

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo. (Pressman, 2002)

Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí si toma sentido el basarnos en una metodología de desarrollo, y se empieza a buscar cual sería la más apropiada para nuestro caso.

Lo cierto es que muchas veces no encontramos la más adecuada y terminamos por hacer o diseñar nuestra propia metodología, algo que por supuesto no está mal, siempre y cuando cumpla con el objetivo (Ivar Jacobson, 2000). La mayoría de las veces se realiza el diseño del software de manera rígida, con los requerimientos que el cliente solicita, de tal manera que cuando el cliente en la etapa final (etapa de prueba), solicita un cambio se hace muy difícil realizarlo, pues si se hace, altera muchas cosas que no habían previsto, y es justo éste, uno de los factores que ocasiona un atraso en el proyecto y por tanto la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido. Obviamente para evitar estos incidentes se debe haber llegado a un acuerdo formal con el cliente, al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique al desarrollo del mismo.

Muchas veces los usuarios finales, se dan cuenta de las cosas que dejaron de mencionar, recién en la etapa final del proyecto, pese a que se les mostró un prototipo del software en la etapa inicial del proyecto. Los proyectos en problemas son los que salen del presupuesto, tienen importantes retrasos, o simplemente no cumplen con las expectativas del cliente.

Para dar una idea de qué metodología se utilizará y cual se adapta más al medio, se mencionan tres de ellas a continuación:

RUP, XP y MSF.

1.12.3 Lenguaje de Unificado y Modelado (UML).

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Probablemente, una de las innovaciones conceptuales en el mundo tecnológico del desarrollo de software que más expectativas y entusiasmos haya generado en muchos años (Rumbaugh, 2000). Es un estándar en la industria del software, creado por Grady Booch, James Rumbaugh e Ivar Jacobson.

1.12.4 Modelo

Un modelo en una aplicación informática es una abstracción del software, que se especifica desde cierto punto de vista y con determinado nivel de abstracción, detalles del diseño de un sistema. (Ivar Jacobson, 2000)

1.12.5 Rational Unified Process (RUP)

RUP Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para el desarrollo de proyectos, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- **Inicio**: El Objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: En esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- **Transición**: El objetivo es llegar a obtener el reléase del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de desarrollo

- **Ingeniería de Negocios:** Entendiendo las necesidades del negocio.
- **Requerimientos:** Trasladando las necesidades del negocio a un sistema automatizado.
- **Análisis y Diseño:** Trasladando los requerimientos dentro de la arquitectura de software.
- **Implementación:** Creando software que se ajuste a la arquitectura y que tenga compartimiento deseado.
- **Pruebas:** Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado está presente.

Disciplina de soporte

- **Configuración y administración del cambio:** Guardando todas las versiones del proyecto.
- **Administrando el proyecto:** Administrando horarios y recursos.
- **Ambiente:** Administrando el ambiente de desarrollo.
- **Distribución:** Hacer todo lo necesario para la salida del proyecto.

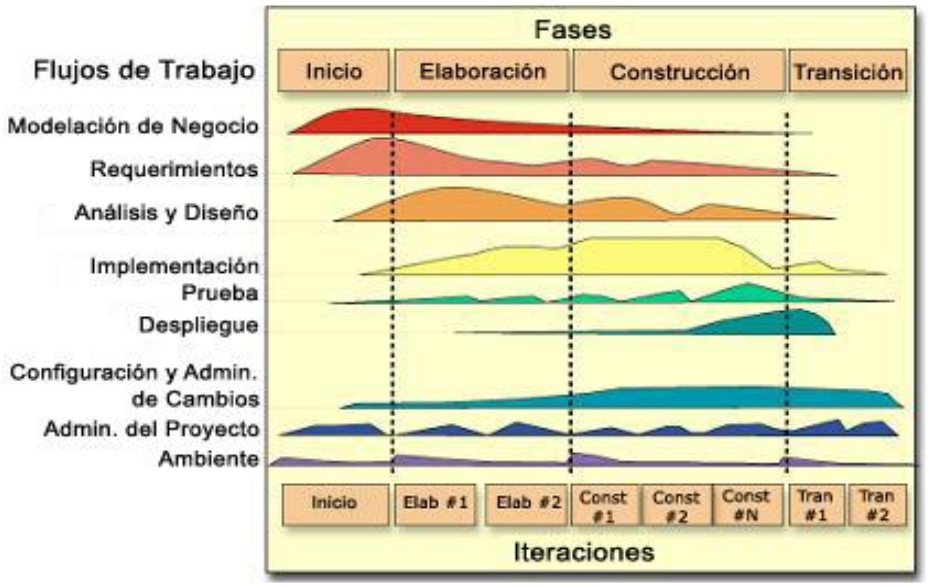


Figura5. Fases e Iteraciones de la Metodología RUP

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- **Actividades**: Son los procesos que se llegan a determinar en cada iteración.
- **Trabajadores**: Vienen hacer las personas o entes involucrados en cada proceso.
- **Artefactos**: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (Ivar Jacobson, 2000)

1.12.6 Extreme Programing (XP)

La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

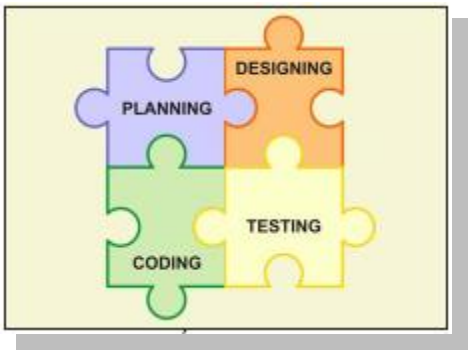


Figura 6. Metodología Extreme Programing

Características de XP

La metodología se basa en**:**

Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelante en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

Re-fabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo 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. (Extreme Programming)

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua
- El manejo del cambio se convierte en parte sustantiva del proceso
- El costo del cambio no depende de la fase o etapa
- No introduce funcionalidades antes que sean necesarias
- El cliente o el usuario se convierte en miembro del equipo

1.12.7 Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

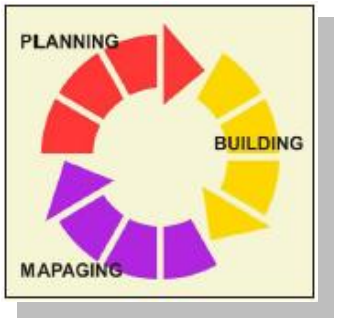


Figura 7. Metodología MSF

MSF tiene las siguientes características:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Con el apoyo del análisis hecho anteriormente sobre algunas metodologías y procesos de desarrollo de software se puede decir que cada una tiene características propias:

- La Metodología RUP es más adaptable para proyectos de largo plazo.
- La Metodología XP en cambio, se recomienda para proyectos de corto plazo.
- La Metodología MSF se adapta a proyectos de cualquier dimensión y de cualquier tecnología.

Como conclusión del estudio realizado se escogió la metodología RUP y el lenguaje UML como notación siguiendo el alcance esperado que obtenga este proyecto y la documentación que se necesite de cada uno de los artefactos, los cuales deberán estar correctamente definidos y documentado para facilitar el posterior desarrollo, mantenimiento y aumentar el alcance del sistema con nuevos requisitos en determinados intervalos de tiempo.

1.13 El análisis

El análisis de requisitos es la primera fase técnica del proceso de ingeniería del software. En este punto se refina la declaración general del ámbito del software en una especificación concreta que se convierte en el fundamento de todas las actividades siguientes de la ingeniería del software.

El análisis debe enfocarse en los dominios de la información, funcional y de comportamiento del problema. Para entender mejor lo que se requiere, se crean modelos, los problemas sufren una partición y se desarrollan representaciones que muestran la esencia de los requisitos y posteriormente los detalles de la implementación.

En muchos casos, no es posible especificar completamente un problema en una etapa tan temprana. La creación de prototipos ofrece un enfoque alternativo que produce un modelo ejecutable del software en el que se pueden refinar los requisitos. Se necesitan herramientas especiales para poder realizar adecuadamente la creación de prototipos.

Como resultado del análisis, se desarrolla la especificación de requisitos del software. La revisión es esencial para asegurarse que el cliente y el desarrollador tienen el mismo concepto del sistema. Desgraciadamente, incluso con los mejores métodos, la cuestión es que el problema sigue cambiando.

1.13.1 Principios del análisis

Durante las dos pasadas décadas, se han desarrollado un gran número de métodos de modelado. Los investigadores han identificado los problemas del análisis y sus causas y han desarrollado varias notaciones de modelado y sus correspondientes conjuntos de heurísticas para solucionarlos. Cada método de análisis tiene su punto de vista. Sin embargo, todos los métodos de análisis se relacionan por un conjunto de principios operativos:

1. Debe representarse y entenderse el dominio de información de un problema.

2. Deben definirse las funciones que debe realizar el software.
3. Debe representarse el comportamiento del software (como consecuencia de acontecimientos externos).
4. Deben dividirse los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas (o jerárquicamente).
5. El proceso de análisis debería ir desde la información esencial hasta el detalle de la implementación. (Pressman, 2002)

¿Por qué es importante? Para validar los requisitos del software se necesita examinarlos desde diferentes puntos de vista. El análisis representa los requisitos en tres dimensiones, por esa razón, se incrementa la probabilidad de encontrar errores, descubrir inconsistencia y detectar omisiones.

¿Cuáles son los pasos? Los requisitos de datos, funciones y comportamientos son modelados utilizando diferentes diagramas. El modelado de datos define objetos de datos, atributos y relaciones. El modelado de funciones indica como los datos son transformados dentro del sistema. El modelado del comportamiento representa el impacto de los sucesos. Se crean unos modelos preliminares que son analizados y refinados para valorar su claridad, completitud y consistencia. Una especificación incorporada en el modelo es creada y luego validada, tanto por el ingeniero del software, como por los clientes/usuarios.

¿Cuál es el producto obtenido? Las descripciones de los objetos de datos, los diagramas entidad–relación, los diagramas de flujo de datos, los diagramas de transición de estados, las especificaciones del proceso y las especificaciones de control son creadas como resultados de las actividades del análisis. (Pressman, 2002)

1.14 Gestión de riesgos

1.14.1 Riesgo

Riesgo es el daño potencial que puede surgir por un proceso presente o suceso futuro, (y esto se puede dar en cualquier ámbito laboral) Diariamente en ocasiones se le utiliza como sinónimo de probabilidad, pero en el asesoramiento profesional de riesgo, el riesgo combina la probabilidad de que ocurra un evento negativo con cuanto daño dicho evento causaría. Es decir, en palabras claras, el riesgo es la posibilidad de que un peligro pueda llegar a materializarse.

Entre las principales definiciones de gestión de riesgo se pueden resaltar las del Project Management Institute (Duncan, 1996)

- La gestión de riesgos es el proceso por el que los factores de riesgo se identifican sistemáticamente y se evalúan sus propiedades.
- La gestión de riesgos es una metodología sistemática y formal que se concentra en identificar y controlar áreas de eventos que tienen la capacidad de provocar un cambio no deseado.
- La gestión de riesgos, en el contexto de un proyecto, es el arte y ciencia de identificar, analizar y responder a los factores de riesgo a lo largo de la vida del proyecto y en el mejor cumplimiento de sus objetivos.

De acuerdo con estas definiciones, un riesgo tecnológico se conceptúa como la posibilidad de que existan consecuencias indeseables o inconvenientes de un acontecimiento relacionado con el acceso o uso de la tecnología y cuya aparición no se puede determinar a priori.

1.14.2 Fases

La gestión de riesgos posee varias fases:

- Fase Identificación
    - Establecer y mantener alcances y estrategias de gestión de riesgo.
    - Identificar, documentar y clasificar riesgos
- Fase de Evaluación
    - Definir métricas de riesgo
    - Analizar y priorizar riesgos
- Definición del Plan de Acción e Implementación
    - Desarrollar estrategias de mitigación
- Seguimiento de Riesgos
    - Monitorear y controlar riesgos

1.15 Ingeniería de requerimientos

En la ingeniería de sistemas y la ingeniería de software la Ingeniería de requerimientos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requerimientos de los inversores, que

pueden entrar en conflicto entre ellos. Puede ser conocida también como "Análisis de requerimientos" o especificación de requerimientos".

El propósito de la ingeniería de requerimientos es hacer que los mismos alcancen un estado óptimo antes de seguir adelante con el proyecto. Los buenos Requerimientos deben ser medibles, comprobables, sin ambigüedades o contradicciones.

1.15.1 Actividades

Desde un punto de vista conceptual, las actividades se dividen en 5 clases.

- **Obtener requerimientos:** A través de entrevistas o comunicación con clientes o usuarios, para saber cuáles son sus deseos.
- **Analizar requerimientos:** Detectar y corregir las falencias comunicativas, transformando los requerimientos obtenidos de entrevistas y requerimientos, en condiciones apropiadas para ser tratados por el diseño; los requerimientos deben estar debidamente documentados.
- **Verificar los requerimientos:** Una vez que la especificación de requisitos ha sido desarrollada, los requisitos son verificados. La verificación de requisitos es un proceso para asegurar que la especificación de requisito del producto es una representación exacta de las necesidades del cliente. Este proceso también asegura que los requisitos sean trazados y verificados a través de varias fases del ciclo de vida; particularmente en el diseño, implementación y pruebas. Los requisitos deben ser trazados desde fuentes externas, tales como los clientes, para derivar requisitos del nivel del sistema, para especificar requisitos del producto hardware/software. Además, todos estos requerimientos deben ser trazados al diseño, implementación y pruebas para asegurarse que los requerimientos han sido satisfechos.
- **Validar los requerimientos:** Comprobar que los requerimientos implementados se corresponden con lo que inicialmente se pretendía.
- **Gestión de cambios**: Gestión de cambios es un proceso formal para identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto. Como el proyecto va evolucionando, los requerimientos pueden cambiar o expandirse para ajustar algunas modificaciones en el alcance o diseño del proyecto. Un proceso de gestión de cambios proporciona un rastreo completo y preciso de todos los cambios que son pertinentes al proyecto.

1.15.2 Técnicas principales

La ingeniería de requisitos puede ser un proceso largo y arduo para el que se requiere de habilidades psicológicas. Los nuevos sistemas cambian el entorno y las relaciones entre la gente, así que es importante identificar a todas las personas implicadas, considerar sus necesidades y asegurar que entienden las implicaciones de los nuevos sistemas. Los analistas pueden emplear varias técnicas para obtener los requisitos del cliente. Históricamente, esto ha incluido técnicas tales como las entrevistas, o talleres con grupos para crear listas de requisitos. Técnicas más modernas incluyen los prototipos, y utilizan casos de uso. Cuando sea necesario, el analista empleará una combinación de estos métodos para establecer los requisitos exactos de las personas implicadas, para producir un sistema que resuelva las necesidades del negocio.

1.15.2.1 Entrevistas

Las entrevistas son un método común. Por lo general no se entrevista a toda la gente que se relacionará con el sistema, sino a una selección de personas que represente a todos los sectores críticos de la organización, con el énfasis puesto en los sectores más afectados o que harán un uso más frecuente del nuevo sistema. Los requerimientos que surgen de las entrevistas a menudo se contradicen unos a otros o se formulan desde la ignorancia de los detalles del funcionamiento del sistema, sus potencialidades, interdependencias o limitaciones; por lo que se debe trabajar con los mismos para corregir sus fallas.
Las entrevistas pueden ser personales o grupales.

1.15.2.2 Talleres

Los requisitos tienen a menudo implicaciones cruzadas desconocidas para las personas implicadas individuales y que a menudo no se descubren en las entrevistas o quedan incompletamente definidas durante la misma. Estas implicaciones cruzadas pueden descubrirse realizando en un ambiente controlado, talleres facilitados por un analista del negocio, en donde las personas implicadas participan en discusiones para descubrir requisitos, analizan sus detalles y las implicaciones cruzadas. A menudo es útil la selección de un secretario dedicado a la documentación de la discusión, liberando al analista del negocio para centrarse en el proceso de la definición de los requisitos y para dirigir la discusión.

1.15.2.3 Forma de contrato

En lugar de una entrevista, se pueden llenar formularios o contratos indicando los requerimientos. En sistemas muy complejos éstos pueden tener centenares de páginas.

1.15.2.4 Objetivos mesurables

Los requerimientos formulados por los usuarios se toman como objetivos generales, a largo plazo, y en cambio se los debe analizar una y otra vez desde el punto de vista del sistema hasta determinar los objetivos críticos del funcionamiento interno que luego darán forma a los comportamientos apreciables por el usuario. Luego, se establecen formas de medir el progreso en la construcción, para evaluar en cualquier momento qué tan avanzado se encuentra el proyecto.

1.15.2.5 Prototipos

Un prototipo es una pequeña muestra, de funcionalidad limitada, de cómo sería el producto final una vez terminado. Ayudan a conocer la opinión de los usuarios y rectificar algunos aspectos antes de llegar al producto terminado.

1.15.2.6 Casos de uso

Un caso de uso es una técnica para documentar posibles requerimientos, graficando la relación del sistema con los usuarios u otros sistemas. Dado que el propio sistema aparece como una caja negra, y sólo se representa su interacción con entidades externas, permite omitir dichos aspectos y determinar los que realmente corresponden a las entidades externas. El objetivo de esta práctica es mejorar la comunicación entre los usuarios y los desarrolladores, mediante la prueba temprana de prototipos para minimizar cambios hacia el final del proyecto y reducir los costes finales. Esta técnica se enfrenta a los siguientes peligros potenciales.

- A los directivos, una vez que ven un prototipo, les cuesta comprender que queda mucho trabajo por hacer para completar el diseño final.
- Los diseñadores tienden a reutilizan el código de los prototipos por temor a “perder el tiempo” al recomenzar otra vez.
- Los prototipos ayudan principalmente a las decisiones del diseño y del interfaz de usuario. Sin embargo, no proporcionan explícitamente cuáles son los requisitos.

- Los diseñadores y los usuarios finales pueden centrarse demasiado en diseño del interfaz de usuario y demasiado poco en producir un sistema que sirva el proceso del negocio.

Los prototipos pueden ser, por ejemplo, diagramas, aplicaciones operativas con funcionalidades sintetizadas. Los diagramas en los casos donde se espera que el software final tenga diseño gráfico, se realizan en una variedad de documentos de diseño gráficos y a menudo elimina todo el color del diseño del software (es decir utilizar una gama de grises). Esto ayuda a prevenir la confusión sobre la apariencia final de la aplicación.

1.15.3 Especificación de requisitos del software

Una especificación de requisitos del software es una descripción completa del comportamiento del sistema a desarrollar. Incluye un conjunto de casos de uso que describen todas las interacciones que se prevén que los usuarios tendrán con el software. También contiene requisitos no funcionales (o suplementarios). Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del diseño).
Las estrategias recomendadas para la especificación de los requisitos del software están descritas por IEEE 830-1998. Este estándar describe las estructuras posibles, contenido deseable, y calidades de una especificación de requisitos del software.

1.15.4 Identificación de las personas involucradas

Debido a que los cambios que introduce un sistema nuevo tienden a afectar a más de un tipo de usuario, los analistas de requisitos han de tomar en consideración a todos los implicados para que se obtengan y depuren sus requerimientos de la forma más fidedigna posible. Entre las personas implicadas hay que considerar:

- Organizaciones que integran la organización del analista que está diseñando el sistema
- Organizaciones o sistemas de respaldo
- Dirección
- Usuarios

1.15.5 Problemas

1.15.5.1 Relacionados con las personas involucradas

Vías que pueden dificultar la determinación de los requisitos son:

- Los usuarios no tiene claro lo que desean
- Los usuarios no se involucran en la elaboración de requisitos escritos
- Los usuarios insisten en nuevos requisitos después de que el coste y la programación se hayan fijado.
- La comunicación con los usuarios es lenta
- Los usuarios no participan en revisiones o son incapaces de hacerlo.
- Los usuarios no comprenden los problemas técnicos
- Los usuarios no entienden el proceso del desarrollo

Esto puede conducir a la situación donde las exigencias del consumidor cambian, incluso cuando el desarrollo del producto ya está en marcha.

1.15.5.2 Relacionados con los analistas

La correcta redacción de las Especificaciones de Requisitos Software es imprescindible para el correcto desarrollo del proyecto. Por ello, en su redacción hay que evitar:

- Uso de terminología ambigua en la redacción de los documentos de requisitos
- Sobre especificación de los requisitos
- Escritura poco legible, voz pasiva, abuso de negaciones
- Uso de verbos en condicional, expresiones subjetivas
- Ausencia de términos y verbos del dominio de la aplicación

1.15.5.3 Relacionados con los desarrolladores

Los problemas posibles causados por los desarrolladores durante análisis de requisitos son:

- El personal técnico y los usuarios finales pueden tener diversos vocabularios y pueden llegar a creer incorrectamente que están de acuerdo, no dándose cuenta del desacuerdo hasta que se provee el producto final.
- Los desarrolladores pueden intentar encajar el sistema en un modelo existente, en vez de desarrollar un sistema adaptado a las necesidades del cliente.

- El análisis de requisitos se puede realizar a menudo por los ingenieros o programadores, en vez de personal con las dominio habilidades de relación con la gente y el conocimiento del para entender las necesidades de un cliente correctamente.

1.16 Diseño del software

1.16.1 Principios del diseño

El diseño de software es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en los requisitos del cliente, y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un diseño bueno. En el contexto de la ingeniería del software, el diseño se centra en cuatro aéreas importantes de interés: datos, arquitectura, interfaces y componentes. En estas cuatro áreas se aplican los principios que se abordan a continuación.

- En el proceso de diseño no deberá utilizarse “orejeras”. Un buen diseñador deberá tener en cuenta enfoques alternativos, juzgando todos los que se basan en los requisitos del problema, los recursos disponibles para realizar el trabajo y los conceptos de diseño.
- El diseño deberá poderse rastrear hasta el modelo de análisis. Dado que un solo elemento del modelo de diseño suele hacer un seguimiento de los múltiples requisitos, es necesario tener un medio de rastrear como se han satisfecho los requisitos por el modelo de diseño.
- El diseño no deberá inventar nada que ya esté inventado. Los sistemas se construyen utilizando un conjunto de patrones de diseño, muchos de los cuales probablemente ya se han encontrado antes. Estos patrones deberán elegirse siempre como una alternativa para reinventar. Hay poco tiempo y los recursos son limitados. El tiempo de diseño se deberá invertir en la representación verdadera de ideas nuevas y en la integración de esos patrones que ya existen.
- El diseño deberá minimizar la distancia intelectual entre el software y el problema como si de la misma vida real se tratara. Es decir, la estructura del diseño del software (siempre que sea posible) imita la estructura del dominio del problema.
- El diseño deberá presentar uniformidad e integración. Un diseño es uniforme si parece que fue una persona la que lo desarrolló por completo. Las reglas de estilo y de formato deberán definirse para

un equipo de diseño antes de comenzar el trabajo sobre el diseño. Un diseño se integra si se tiene cuidado a la hora de definir interfaces entre los componentes del diseño.

- El diseño deberá estructurarse para admitir cambios. Los conceptos de diseño estudiados hacen posible un diseño que logra este principio.
- El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operación aberrantes. Un software bien diseñado no deberá nunca explotar, deberá diseñarse para adaptarse a circunstancias inusuales y si debe terminar de funcionar, que lo haga de forma suave.
- El diseño no es escribir código y escribir código no es diseñar. Incluso cuando se crean diseños procedimentales para componentes de programas, el nivel de abstracción del modelo de diseño es mayor que el código fuente. Las únicas decisiones de diseño realizadas a nivel de codificación se enfrentan con pequeños datos de implementación que posibilitan codificar el diseño procedimental.
- El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminarlo. Para ayudar al diseñador en la evaluación de la calidad se dispone de conceptos de diseño y de medidas de diseño.
- El diseño deberá revisarse para minimizar los errores conceptuales. A veces existe la tendencia de centrarse en minucias cuando se revisa el diseño, olvidándose del bosque por culpa de los árboles. Un equipo de diseñadores deberá asegurarse de haber afrontado los elementos conceptuales principales antes de preocuparse por la sintaxis del modelo del diseño. (Pressman, 2002)

¿Quién lo hace? El ingeniero del software es quien diseña los sistemas basados en computadoras, pero los conocimientos que se requieren en cada nivel de diseño funcionan de diferentes maneras. En el nivel de datos y de arquitectura, el diseño se centra en los patrones de la misma manera a como se aplican en la aplicación que se va a construir. En el nivel de la interfaz, es la ergonómica humana la que dicta nuestro enfoque de diseño. Y en el nivel de componentes, un “enfoque de programación” conduce a diseños de datos y procedimentales eficaces.

¿Por qué es importante? Si se construye una casa, ¿se hace sin un plano? Se correrían riesgos, se cometerían errores, habría un plano de casa sin sentido, con ventanas y puertas en sitios equivocados…

un desastre. El software de computadora es considerablemente más complejo que una casa, de aquí que se necesita un plano “el diseño”.

¿Cuáles son los pasos? El diseño comienza con el modelo de los requisitos. Se trabaja por transformar este modelo y obtener cuatro niveles de detalles de diseño: la estructura de datos, la arquitectura del sistema, la representación de la interfaz y los detalles a nivel de componentes. Durante cada una de las actividades del diseño, se aplican los conceptos y principios básicos que llevan a obtener una alta calidad.

¿Cuál es el producto obtenido? Por último se produce una especificación del diseño, La especificación se compone de los modelo del diseño que describen los datos, arquitectura, interfaces y componentes. Cada una de esta parte es la que forma el producto obtenido del proceso de diseño.

¿Cómo puedo estar seguro de que lo he hecho correctamente? En cada etapa de revisión los productos del diseño del software en cuanto a claridad, corrección, finalización y consistencia, y se comparan con los requisitos y unos con otros. (Pressman, 2002)

1.16.2 Patrones de diseño

Los patrones de diseño (design patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

1.16.2.1 Objetivos de los patrones

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones siempre, solo en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error.

1.16.2.2 Categorías de patrones

Según la escala o nivel de abstracción:

- Patrones de arquitectura: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas software.
- Patrones de diseño: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.
- Idiomas: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto. Además, también es importante reseñar el concepto de Antipatrón de Diseño, que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software. La idea de los antipatrones es dar a conocer los problemas que acarrean ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida por haber cometido los mismos errores.

1.16.2.3 Estructuras o plantillas de patrones

Para describir un patrón se utilizan plantillas más o menos estandarizadas, de forma que se expresen uniformemente y puedan constituir efectivamente un medio de comunicación uniforme entre diseñadores. Varios autores eminentes en esta área han propuesto plantillas ligeramente distintas. Si bien la mayoría definen los mismos conceptos básicos.

La plantilla más común es la utilizada precisamente por el GoF y consta de los siguientes apartados:

- Nombre del patrón: nombre estándar del patrón por el cual será reconocido en la comunidad (normalmente se expresan en inglés).
- Clasificación del patrón: creacional, estructural o de comportamiento.
- Intención: ¿Qué problema resuelve el patrón?

- También conocido como: Otros nombres de uso común para el patrón.
- Motivación: Escenario de ejemplo para la aplicación del patrón.
- Aplicabilidad: Criterios de aplicabilidad del patrón.
- Estructura: Diagramas de clases oportunos para describir las clases que intervienen en el patrón.
- Participantes: Enumeración y descripción de las entidades abstractas (y sus roles) que participan en el patrón.
- Colaboraciones: Explicación de las interrelaciones que se dan entre los participantes.
- Consecuencias: Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón.
- Implementación: Técnicas o comentarios oportunos de cara a la implementación del patrón.
- Código de ejemplo: Código fuente ejemplo de implementación del patrón.
- Usos conocidos: Ejemplos de sistemas reales que usan el patrón.
- Patrones relacionados: Referencias cruzadas con otros patrones.

1.16.2.4 Relación de principales patrones GoF

Patrones creacionales

- Abstract Factory (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- Builder (Constructor virtual): Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.
- Factory Method (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.
- Prototype (Prototipo): Crea nuevos objetos clonándolos de una instancia ya existente.
- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones estructurales

- Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- Bridge (Puente): Desacopla una abstracción de su implementación.
- Composite (Objeto compuesto): Permite tratar objetos compuestos como si de un objeto simple se tratase.
- Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.
- Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- Flyweight (Peso ligero): Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- Proxy: Mantiene un representante de un objeto.

Patrones de comportamiento

- Chain of Responsibility (Cadena de responsabilidad): Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
- Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- Interpreter (Intérprete): Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
- Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.
- Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- Memento (Recuerdo): Permite volver a estados anteriores del sistema.
- Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.
- State (Estado): Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.

- Template Method (Método plantilla): Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.
- Visitor (Visitante): Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.

1.17 Métricas

La medición es fundamental para cualquier disciplina de ingeniería, y la ingeniería del software no es una excepción. La medición permite tener una visión más profunda proporcionando un mecanismo para la evaluación objetiva. *Lord Kelvin* en una ocasión dijo: Cuando pueda medir lo que está diciendo y expresarlo con números, ya conoce algo sobre ello; cuando no pueda medir, cuando no pueda expresar lo que dice con números, su conocimiento es precario y deficiente: puede ser el comienzo del conocimiento, pero en sus pensamientos, apenas está avanzando hacia el escenario de la ciencia. (Pressman, 2002)

Las métricas del software se refieren a un amplio elenco de mediciones para el software de computadora. La medición se puede aplicar al proceso del software con el intento de mejorarlo sobre una base continua. Se puede utilizar en el proyecto del software para ayudar en la estimación, el control de calidad, la evaluación de productividad y el control de proyectos.

Finalmente, el ingeniero de software puede utilizar la medición para ayudar a evaluar la calidad de los resultados de trabajos técnicos y para ayudar en la toma de decisiones táctica a medida que el proyecto evoluciona.

A continuación se examina una de las métricas de diseño más comunes para el software. Aunque ninguna es perfecta, todas pueden proporcionar al diseñador una mejor visión interna y ayudar a que el diseño evolucione a un nivel superior de calidad.

1.17.1 Métricas del diseño arquitectónico

Las métricas de diseño de alto nivel se concentran en las características de la arquitectura del programa, con especial énfasis en la estructura arquitectónica y en la eficiencia de los módulos. Estas métricas son de caja negra en el sentido que no requieren ningún conocimiento del trabajo interno de un módulo en particular del sistema.

Algunos expertos definen tres medidas de la complejidad del diseño del software: **complejidad estructural, complejidad de datos y complejidad del sistema.**

La complejidad estructural, $S(i)$ de un módulo i se define de la siguiente manera: $S(i) = f_{out}^{2}(i)$ donde $f_{out}(i)$ es la expansión del módulo i.

La complejidad de datos, $D(i)$ proporciona una indicación de la complejidad de la interfaz interna de un módulo i y se define como: $D(i) = V(i)/\left\lfloor f_{out}(i)+1 \right\rfloor$ donde $V(i)$ es el número de variable de entrada y de salida que entran y salen del módulo i.

La complejidad del sistema, $C(i)$ se define como las sumas de las complejidades estructural y de datos, y se define como: $C(i) = S(i) + D(i)$

A medida que crecen los valores de complejidad, la complejidad arquitectónica o global del sistema también aumenta. Esto lleva a una mayor probabilidad de que aumente el esfuerzo necesario para la integración y las pruebas. (Pressman, 2002)

1.17.2 Métricas orientadas a clase

Lorenz y Kidd en su libro sobre métricas OO, separa las métricas basadas en clases en cuatro amplias categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño para las clases OO se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema OO como un todo. Las métricas basadas en la herencia se centran en la forma en que las operaciones se reutilizan en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y los aspectos orientados al código; las métricas orientadas a valores externos, examinan el acoplamiento y la reutilización.

1. Tamaño de clase (MPC).
2. Número de operaciones redefinidas por una subclase (NOR).
3. Número de operaciones añadidas por una subclase (NOA).

1.17.3 Métricas de diseño a nivel de componentes

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen medidas de las “3Cs” -la cohesión, acoplamiento y complejidad del

módulo-. Estas tres medidas pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes. Estas métricas son de caja blanca en el sentido de que requieren conocimiento del trabajo interno del módulo en cuestión. Las métricas de diseño de los componentes se pueden aplicar una vez que se ha desarrollado un diseño procedimental. También se pueden retrasar hasta tener disponible el código fuente.

1.18 Conclusiones

En este capítulo se han descrito procesos y dado una definición sobre conceptos claves para el objeto de estudio, se ha hecho alusión al campo de acción del entorno de negocio. Se hizo un análisis de los productos actuales desarrollados como antecedentes y base para la aplicación. Ha quedado plasmado un análisis de algunas de las principales plataformas y herramientas de desarrollo, que posibilitan una mayor productividad. Se definieron las tecnologías y herramientas a utilizar en el diseño de la solución y se estipularon las métricas correspondientes.

Se arribaron a las siguientes conclusiones:

- Emplear RUP como metodología de desarrollo, específicamente lo inherente al flujo de trabajo de Análisis y Diseño.
- Utilizar la plataforma JEE, específicamente Eclipse 3.3 y como lenguaje de programación Java, los cuales facilitan el trabajo de los desarrolladores considerablemente, elemento necesario dado el corto periodo de entrega del software.
- Utilizar el sistema gestor de base de datos PostgreSQL 8.1 debido a que es una herramienta de software libre que brinda un rendimiento máximo, su escalabilidad y capacidad de almacenamiento, respuesta rápida y seguridad.
- Emplear la herramienta CASE Rational Rose para crear los diagramas del diseño del sistema, utilizando la notación estándar en la arquitectura de software (UML).
- Utilizar Hibernate como el Framework para generar las capas de acceso a datos.
- Utilizar métrica definida por Pressman para el chequeo y validación del Modelo de Diseño obtenido a nivel de arquitectura.

Estas conclusiones sirven de guía para el posterior desarrollo de la propuesta de solución al problema planteado.

CAPITULO 2

CAPITULO 2. CARACTERÍSTISTICAS DEL SITEMA

2.1 Introducción

En el presente capítulo se describe la solución propuesta mediante el modelo de dominio teniendo en cuenta la definición de las entidades y los conceptos principales, así como la representación gráfica de dicho modelo utilizando los componentes de la metodología RUP. Se plantean las reglas de negocio, los requerimientos funcionales y no funcionales de la aplicación a desarrollar y se describe la solución propuesta en términos de casos de uso del sistema.

2.2 Objetivo básico. Módulo Entrada de Datos

El objetivo básico del Módulo es asegurar una eficaz entrada de la información al sistema unido a la generación de reportes.

2.2.1 Objetivos estratégicos generales del Módulo

Tiene como objetivos garantizar la fiabilidad, seguridad, y estandarización de la información que se introduce en el sistema y que los reportes correspondan con las necesidades de los diferentes usuarios.

2.2.2 Objetivos estratégicos específicos del Módulo

1. Crear estándares para la introducción de la información y sus procesos.
2. Validar la autenticidad de la información que se introduce en el sistema.
3. Garantizar que la puesta de acuerdo entre las partes involucradas en la ejecución de un proyecto se realice de forma automática, sencilla y fiable.
4. Permitir la actualización de la información.
5. Asegurar un acceso rápido a la información que necesita el usuario mediante opciones de búsqueda.

2.3 Selección del modelado

Existen como mínimo dos aproximaciones para expresar el contexto de un sistema en una forma utilizable para desarrolladores de software: modelado del dominio y modelado del negocio.

En la investigación se decidió desarrollar el modelo del dominio porque se hace difícil encontrar la definición de procesos del negocio, quienes son las personas que lo inician, quienes son los beneficiados con cada uno de estos procesos, pero además quienes son las personas que desarrollan las actividades en cada uno de estos procesos. Se necesita describir mediante una serie de conceptos y sus relaciones; agrupados en un modelo del dominio para un fácil entendimiento de la aplicación.

2.3.1 Definición de las entidades y los conceptos principales

La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema. Para comprender mejor este moldeamiento se muestra a continuación una descripción de la estructura organizativa por niveles que rige el convenio de colaboración Cuba – Venezuela.

2.3.1.1 Estructura organizativa

Como se muestra en la Fig.8 existen de cada una de las partes, o sea, de la parte cubana y la venezolana toda una estructura que permite la coordinación para que fluyan los proyectos aprobados en La Mixta, el primer nivel está constituido por los Directivos del Gobierno (DG). En un nivel inferior están las Secretarias Técnicas (ST). Subordinados a este nivel de ST se encuentran los ministerios correspondientes a cada país. Así mismo, los ministerios incluyen a un conjunto de entidades que se desempeñan como ejecutoras de los proyectos los cuales se nombran como Entes Ejecutores (EE).

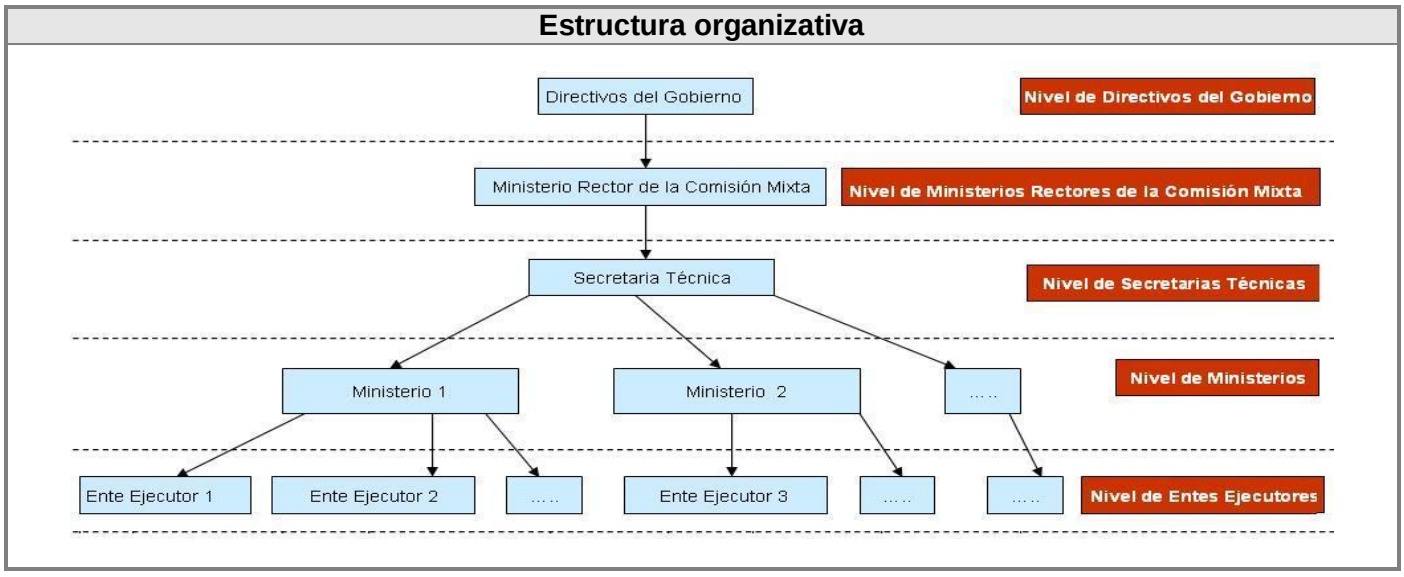


Figura 8. Estructura organizativa.

Con la aplicación del Modelo del dominio se detectaron las siguientes entidades y conceptos (objetos):

- **Usuario:** Este objeto surge como una generalización del resto de los objetos o entidades involucradas (Ente Ejecutor, Ministerio, Secretaría Técnica, a esto se llamaría roles) dentro de los sucesos que ocurren. Es el encargado de gestionar todas las funcionalidades del sistema y solicitarle al mismo la generación de reportes.

- **Secretaria Técnica:** Son los encargadas de coordinar, en ambas partes, todas las actividades necesarias para el buen desarrollo de cada proyecto. (MINVEC y MPPENPET). Además llevan un control estadístico de la evolución de cada uno de los proyectos. Solo debe señalarse que el trabajo de los especialistas en finanzas y misiones es revisado por el resto de los especialistas que se dedican a la atención de los ministerios. Las Secretarias Técnicas no cambian nada en las fichas generalmente, ellas solo sugieren modificaciones en caso de que en los niveles inferiores se haya aprobado una ficha que tenga algún tipo de error. Se debe tener en cuenta si un nivel superior cambia la ficha, este se lo notifica al nivel inferior implicado.

- **Ministerio:** Controlan y aprueban las acciones relacionadas con los proyectos. Este nivel incorpora tanto al Ministro como al Responsable Ejecutivo, este último tiene los derechos de decidir en nombre del Ministro aunque sí puede realizarle consultas para tomar una determinada decisión que

puede resultar compleja. Registran la información, aprueban o sugieren modificaciones a las fichas en caso de que en los niveles inferiores se haya cometido algún tipo de error.

- **Ente Ejecutor:** Son los encargados de coordinar y ejecutar cada una de las acciones relacionadas con los proyectos correspondientes a la entidad. Tienen derecho absoluto (editar, aprobar, rechazar, eliminar) sobre los documentos que rigen los procesos en la parte que coordinan. Registran la información, aprueban o sugieren modificaciones a las fichas en caso de que en los niveles inferiores se haya cometido algún tipo de error.

- **Ficha Resumen:** Guarda información sobre el estado actual del proyecto, su procedencia y contrapartes involucradas en la ejecución. Los datos que recoge la misma son: Nombre del proyecto, Ministerio venezolano, Ente venezolano, Ministerio cubano, Ente cubano, Duración en meses, Marco de aprobación del proyecto, Modalidad, Estado, Monto total del proyecto (Se divide en: A transferir a Cuba, A invertir en Venezuela), porciento de ejecución física, porciento de ejecución financiera, y Fondos.

- **Estado de Proyecto:** Resume la secuencia de estados por los cuales pasa un proyecto dentro de su tiempo de vida; está dividido en: No Iniciado, Iniciado, Contratado y Terminado. Es válido destacar que sobre estos estados se emiten reglas que están estipuladas en el próximo epígrafe.

- **Ejecución Física:** Realización física del proyecto, lo constituye las actividades que se deben realizar en el proyecto. Esta fase puede comenzar en cualquier momento, ya sea antes o después de la firma de La Comisión Mixta, del contrato o ambos.

- **Ejecución Financiera:** Realización financiera del proyecto, constituido por los pagos y las transacciones financieras referentes al proyecto. Los Ministros aprueban o rechazan las solicitudes de financiamiento de sus entidades y le pasan la solicitud al ministro del MINVEC, que a su vez pasa el pedido a La Secretaria Técnica para que sea evaluado el pedido después de haber emitido alguna sugerencia si lo considera, los especialistas en La Secretaria Técnica revisan la petición y se tramita la solicitud con el banco.

- **Reporte General:** Recoge información sobre los ministerios correspondientes a cada país dicha información esta desglosada en los siguientes términos: Ministerio, Ente ejecutor, Proyectos, Monto contratado, Monto ejecutado, porciento de Ejecución Financiera. Este reporte se divide niveles más detallados de información referente a un país, un ministerio, o un ente ejecutor.

- **Reporte de Ejecución Financiera:** Recoge información sobre el Monto a Transferir a Cuba y el Monto de Inversión en Venezuela; contiene los siguientes campos: Ministerio, Ente Ejecutor, Proyectos, Monto a Transferir (En el caso de Cuba), Monto Transferido (En el caso de Cuba), Monto a Invertir (En el caso de Venezuela), Monto Invertido (En el caso de Venezuela), y porciento de Ejecución Financiera. Este reporte se divide niveles más detallados de información referente a un país, un ministerio, o un ente ejecutor.

- **Reporte Ejecución Física:** Recoge información específica de un país, dividida en los siguientes campos: Ministerio, Ente Ejecutor, Proyectos, porciento de ejecución física de Cuba y porciento de ejecución física de Venezuela. Este reporte se divide niveles más detallados de información referente a un país, un ministerio, o un ente ejecutor.

- **Sistema CCV:** Podrá ser utilizado por todos los miembros de los ministerios involucrados en ambos países en los proyectos del convenio. Garantizándose altas prestaciones de calidad e integridad de la información que en ellos se maneja. Informatizaría las actividades de gestión de estos proyectos, además de posibilitar contar con un grupo de reportes que tributaran información para la toma de decisiones de las partes involucradas. La explotación y uso del sistema, disminuirá los errores y permitirá una mayor optimización del uso de los recursos y una mejora en las estrategias de trabajo de gestión y control de los proyectos contratados. Podría ser extendido a otros países que se involucren en proyectos de convenio del ALBA.

2.3.1.2 Diagrama: Modelo del Dominio.

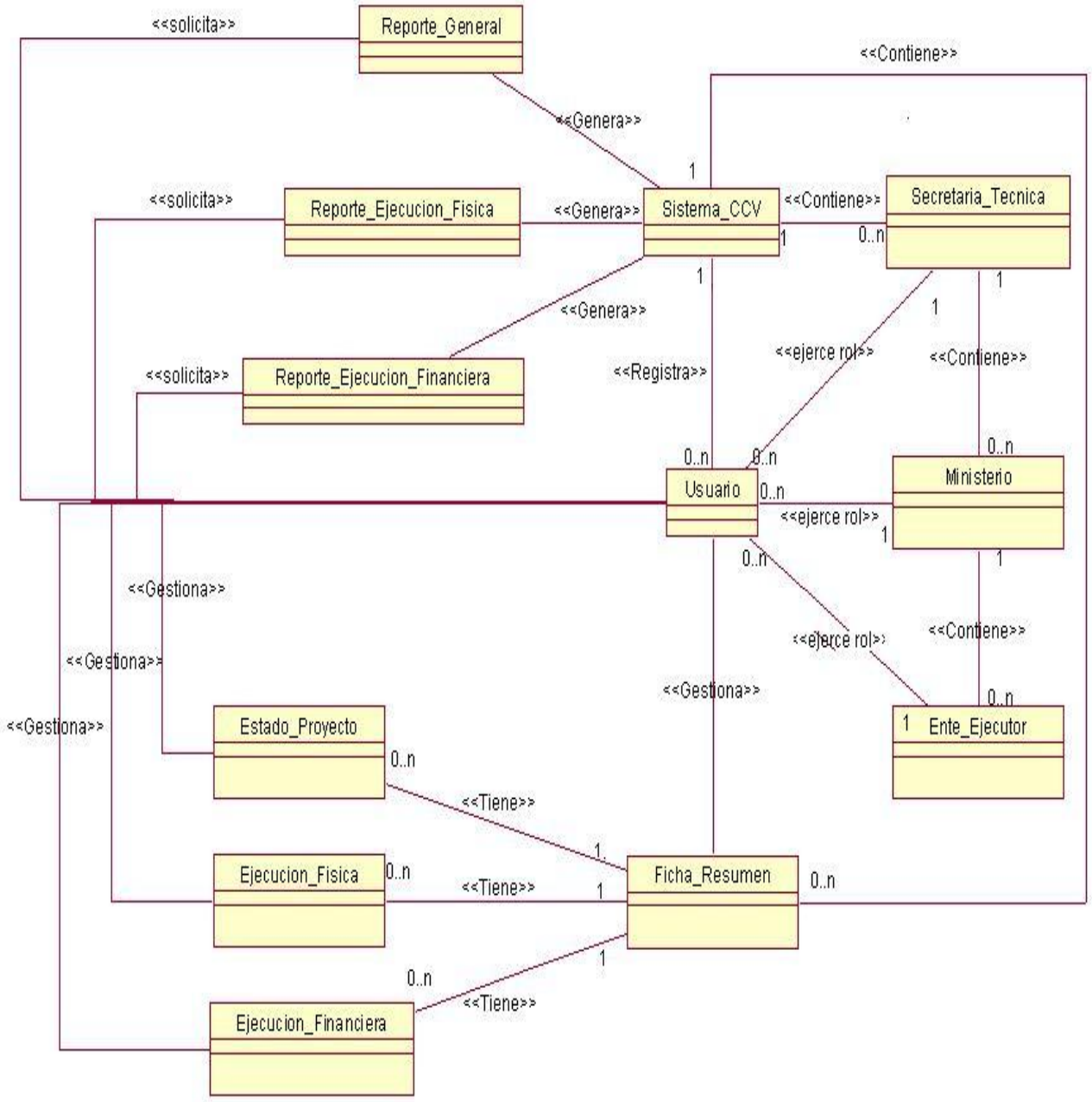


Figura 9. Diagrama Modelo del Dominio.

2.4 Reglas de negocio

Conociendo que “las reglas de negocio” son una colección de políticas y restricciones de negocio de una “organización” y que esto a su vez es un sistema de actividades conscientemente coordinadas formado por dos o más personas, construido bajo un conjunto de cargos cuyas reglas y normas de comportamiento deben sujetarse a todos sus miembros. Se detectaron las reglas específicas para la gestión de proyectos y los permisos según una jerarquía de niveles; sobre lo cual está regido el flujo de los procesos que desarrollan los países de Cuba y Venezuela durante el tiempo de vida de un proyecto. Las cuales serían la base de un producto final que cumpla con las expectativas del cliente.

Para elaborar dichas reglas se realizo un estudio del “Manifiesto de Reglas de Negocio” cuyos artículos citan a continuación: (Business Rules Group, 2005)

Artículo 1. Los requisitos como elementos principales, nunca como secundarios.

Artículo 2. Independientes de los procesos y no contenidas en ellos.

Artículo 3. Proporcionar conocimiento meditado, no un sub-producto.

Artículo 4. Declarativas, no de procedimiento.

Artículo 5. Expresiones bien formadas, no expresiones creadas con fines específicas para un caso.

Artículo 6. Arquitectura basada en las reglas, no una implementación indirecta.

Artículo 7. Procesos guiados por reglas, no programación basada en excepciones.

Artículo 8. Al servicio del negocio, no al de la tecnología.

Artículo 9. “De, por y para” el personal de negocio. No “de, por y para” el personal de IT (Tecnología de la Informática).

Artículo 10. Gestionando la lógica de negocio, no las plataformas de Hardware/Software.

2.4.1 Reglas del Módulo

Por tanto Las Reglas de Negocio quedan estipuladas de la siguiente forma:

1. El ente ejecutor es el único capaz de crear, modificar o desechar un proyecto.
2. Un proyecto puede ser creado por al menos un Ente Ejecutor.
3. Un proyecto se considera válido para ser elevado solo cuando las contrapartes cubana y venezolana estén de acuerdo.
4. Si un proyecto es denegado durante el transcurso de ascenso por los niveles superiores es enviado a directamente sus respectivos entes ejecutores.

5. Un proyecto al ser rechazado nunca es eliminado.
6. Cuando el proyecto está en estado No Iniciado, no se pueden modificar sus datos de Ejecución Física y Ejecución Financiera.
7. Un proyecto puedo ser iniciado sin ser contratado y viceversa.
8. Cuando está en estado Iniciado pero no Contratado no se puede editar su Ejecución Financiera.
9. Cuando un proyecto está en estado de Iniciado se puede modificar su Ejecución Física.
10. A un proyecto se le puede modificar su Ejecución Financiera solo cuando está en estado Iniciado y Contratado.
11. Un proyecto no puede ser terminado sin antes ser al menos iniciado.
12. Un ente, siempre que realice una modificación de estado de un proyecto, está obligado a esperar por la respuesta de la contraparte sin realizar otra modificación.
13. A un proyecto terminado no se le pueden realizar modificaciones.
14. Los entes solo pueden realizar cambios a los datos correspondientes a su país.
15. La fecha de culminación de un proyecto tiene que ser posterior a la fecha de contratación.
16. La secretaria técnica debe conocer todos los trámites que se efectúan durante el tiempo de vida de un proyecto.

2.5 Especificación de requerimientos de software

La información contenida en el presente epígrafe refleja la captura de requisitos para el Módulo de Entrada de Datos. Acopiando los requisitos que debe cumplir la aplicación a desarrollar, identificados a partir de las necesidades reales de los usuarios y las demandas del cliente, con el propósito de guiar el posterior diseño del sistema.

Los requisitos planteados persiguen como alcance llegar a un entendimiento entre el cliente y el equipo de desarrollo mostrando las condiciones que debe presentar el producto desde el punto de vista funcional.

2.5.1 Requerimientos funcionales

Un requerimiento funcional son capacidades o condiciones que el sistema debe cumplir definiendo el comportamiento interno del software así como los comportamientos del sistema. Luego de conocer el modelo de dominio y las clases o conceptos principales que conforman el mismo, se pueden definir los siguientes requisitos funcionales:

RF02.001 Autenticar Usuario.

RF02.002 Gestionar Usuario.

RF02.002.01 Crear Usuario.

RF02.002.02 Buscar Usuario.

RF02.002.03 Modificar Usuario.

RF02.002.04 Listar Usuarios.

RF02.003 Gestionar Ente.

RF02.003.01 Crear Ente.

RF02.003.02 Buscar Ente.

RF02.003.03 Modificar Ente.

RF02.003.04 Listar Entes.

RF02.004 Gestionar Ministerios.

RF02.004.01 Crear Ministerio.

RF02.004.02 Buscar Ministerio.

RF02.004.03 Modificar Ministerio.

RF02.004.04 Listar Ministerios.

RF02.005 Cargar Ficha Resumen de los Proyectos Automáticamente desde un Fichero.

RF02.005.01 Modificar Ficha Resumen de los Proyectos.

RF02.005.02 Aprobar Modificación de las Ficha Resumen de los Proyectos.

RF02.005.03 Rechazar Modificación de las Ficha Resumen de los Proyectos.

RF02.006 Definir Ejecución Física de los Proyectos.

RF02.006.01 Aprobar Ejecución Física de los Proyectos.

RF02.006.02 Rechazar Ejecución Física de los Proyectos.

RF02.007 Definir Ejecución Financiera de los Proyectos.

RF02.007.01 Aprobar Ejecución Financiera de los Proyectos.

RF02.007.02 Rechazar Ejecución Financiera de los Proyectos.

RF02.008 Definir Estado de los Proyectos.

RF02.008.01 Aprobar Modificación de Estado de los Proyectos.

RF02.008.02 Rechazar Modificación de Estado de los Proyectos.

RF02.009 Buscar Proyecto.

RF02.009.01 Mostrar Ficha Resumen de los Proyectos.

RF02.009.02 Imprimir Ficha Resumen de los Proyectos.

RF02.0010 Generar Reporte General.

RF02.0010.01 Imprimir Reporte General.

RF02.0011 Generar Reporte de Ejecución Física.

RF02.0011.01 Imprimir Reporte de Ejecución Física.

RF02.0012 Generar Reporte de Ejecución Financiera por Países.

RF02.0012.01 Imprimir Reporte de Ejecución Financiera por Países.

2.5.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Existen múltiples categorías para clasificar a los requerimientos no funcionales, siendo las siguientes, representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

2.5.2.1 Orientados al usuario:

Fiabilidad:

1. Redondear las cifras significativas a 2 unidades después de coma.

Seguridad (security):

2. Autenticación obligatoria y segura.
3. Los datos que circulan por la red no viajan en texto plano.
4. Acceso a la información según el rol.
5. Manejo de sesiones del usuario, expira la sesión en 1 hora.
6. Debe permitir ocultar la información que aparase en la URL.
7. No permitir SQL injection.

Seguridad (safety):

1. Realizar salvas periódicamente de la información contenida en la base de datos.
2. El sistema debe recuperase ante fallos, ya sea por perdida de conexión, alimentación.

Usabilidad:

1. Permitir uso del teclado para realizar operaciones sobre el sistema.

2. Debe poseer una interfaz agradable para el cliente de acuerdo a los estándares de diseño.
3. Mostar la información de forma lógica y correctamente estructurada.

Disponibilidad:

1. El sistema debe estar accesible desde Internet.

Rendimiento (Tiempo de respuesta, Capacidad, Throughput):

1. Las páginas de la aplicación deben cargar en un tiempo inferior a 3 segundo.
2. Debe garantizarse que con 300 usuarios conectados concurrentemente no disminuya el rendimiento y rapidez de la aplicación.

2.5.2.2 Orientados al desarrollador:

Disponibilidad:

1. Tener una correcta y completa configuración del entorno de trabajo.

Portabilidad:

1. Debe permitirse ser usado en cualquier plataforma.

Adaptabilidad:

1. El sistema debe garantizar la configuración y cambio de sus parámetros de forma fácil y rápida.

Testabilidad:

1. Debe permitir la testabilidad de forma automática de todos los componentes de software.

Comprensibilidad:

1. Debe garantizarse el uso de estándares de codificación.

2.5.2.3 Requisitos para los clientes:

Hardware:

1. CPU Intel Pentium 4.
2. Memoria RAM Mínimo 768 Mb, Memoria RAM Máximo 1 Gb.
3. Capacidad Disco Duro 160 Gb.

Software:

1. Microsoft Internet Explorer 6.0, Mozilla Firefox 2.0.

2.5.2.4 Requisitos para servidores:

Hardware:

1. CPU Intel Pentium 4.
2. Memoria RAM Mínimo 1 Gb, Memoria RAM Máximo 2Gb.
3. Capacidad Disco Duro 160 Gb.

Software:

1. Sistema Operativo: GNU/Linux Debian Etch 4.0.
2. Servidor Web Apache 2.0.
3. Servidor Web Tomcat 5.5.

2.6 Modelo del sistema

En este epígrafe se expone el resultado del trabajo de la captura de requisitos para el Módulo de Entrada de Datos, el mismo recoge la especificación de los casos de uso del sistema (CUS), que han sido identificados a partir de las necesidades reales de los usuarios y de las demandas del cliente. Se establece el acuerdo sobre qué funcionalidades debe implementar la aplicación, a fin de que este trabajo se oriente en satisfacer exclusivamente los requisitos reflejados anteriormente. Además de servir de guía a los desarrolladores a lo largo de todo el proceso de trabajo que ha de convertir estos requisitos en un producto final adaptado a las necesidades del los usuarios. El modelo de sistema permite tener un entendimiento más detallado de cómo va a estar estructurado el sistema a partir de los CU identificados.
Para ello se identifican los actores y se especifican los casos CUS.

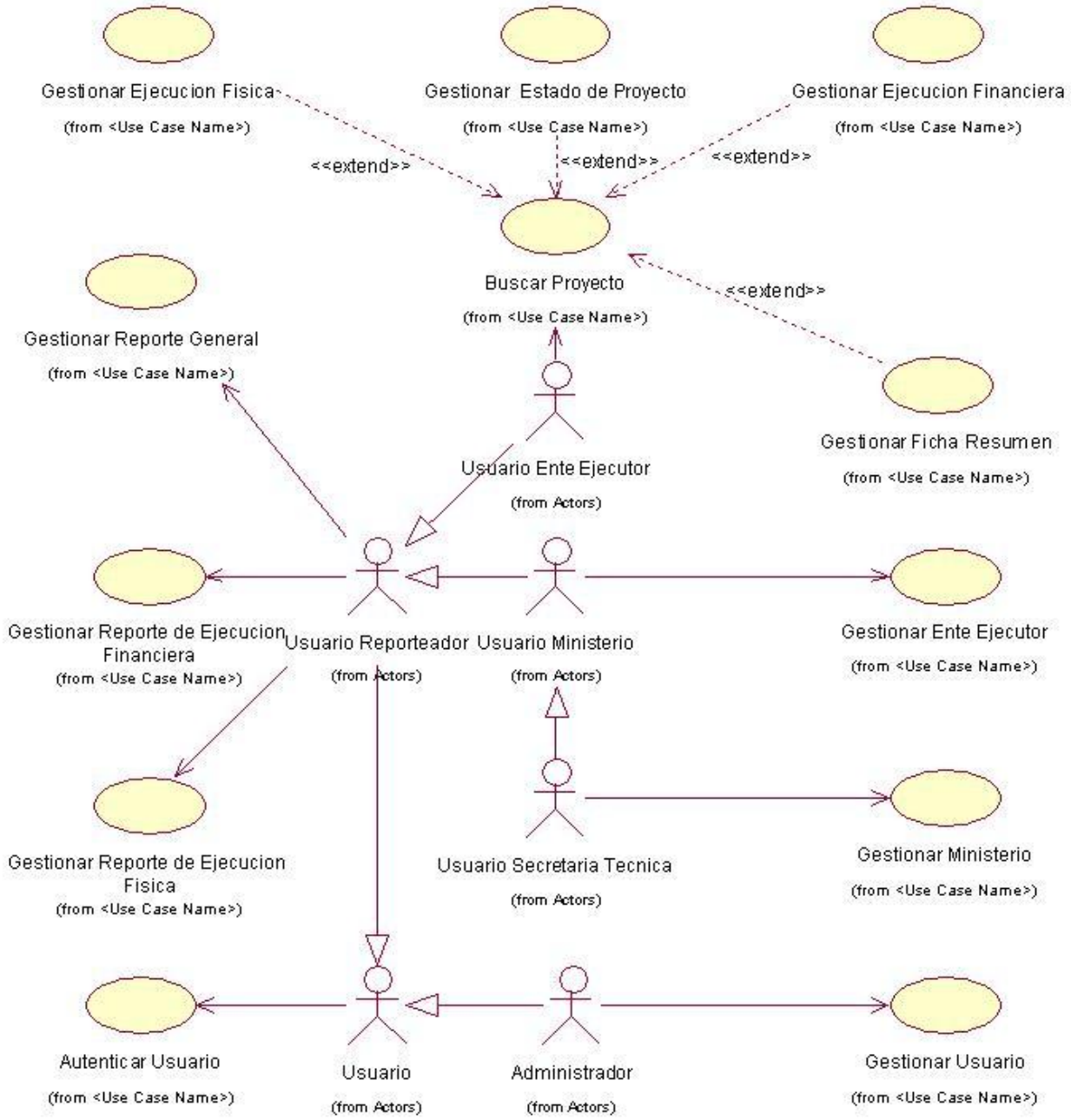
- Los actores de un sistema son agentes externos, es decir, aquellas personas o sistemas que interactúan con él.
- Los casos de uso son un conjunto de secuencia de acciones que un sistema ejecuta y produce un resultado observable para un actor.
- El modelo del sistema es utilizado para representar los de casos de uso y actores con sus relaciones entre sí.

2.6.1 Descripción de Actores del Sistema

Actor	Descripción
Usuario EE	Es el actor que accede al sistema, el que se autentifica y se le da por tanto un determinado nivel de acceso según su rol. Puede crear, modificar y eliminar la Ficha Técnica del Proyecto. También es encargado de actualizar el cronograma de Ejecución Física, Financiera y estado del Proyecto.
Usuario M	Este actor tiene acceso supervisar los proyecto, pero solos los específicos de los entes pertenecientes a su Ministerio. Estas tareas se realizan mediante los reportes.
Usuario ST	Supervisa todos los proyectos del Convenio Cuba-Venezuela. Estas tareas se realizan mediante los reportes.
Administrador	Es el encargado de crear los usuarios del sistema asignándoles los distintos privilegios por roles
Usuario Reporteador	Es el encargado de realizar todos los reportes del sistema.

A continuación se muestra el diagrama de casos de uso del sistema y la especificación de los más relevantes para una mejor comprensión de los mismos.

2.6.2 Diagrama de Casos de Uso del Sistema



2.6.3 Gestionar Ficha Resumen

Caso de Uso:	Gestionar Ficha Resumen
Actores:	Usuario Ente Ejecutor
Resumen:	El caso de uso se inicia cuando el usuario EE accede al sistema y lleva a cabo funcionalidades como modificación, aprobación o rechazo de la Ficha Resumen del proyecto.
Precondiciones:	Que el usuario EE se haya autenticado. Exista conexión con el servidor. Existan Fichas Resumen registradas en el sistema.
Referencias	RF02.005, RF02.005.01, RF02.005.02, RF02.005.03, RF02.009.01, RF02.009.02
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El actor selecciona en el menú de la página la opción (Proyectos).	1.1 Muestra los datos referentes a los proyectos e interfaz de búsqueda. a) Si desea modificar ficha resumen del proyecto, ver sección “Modificar Ficha Resumen”. b) Si desea aprobar la modificación de la Ficha Resumen, ver sección “Aprobar Modificación de Ficha Resumen”. c) Si desea rechazar la modificación de la Ficha Resumen, ver sección “Rechazar Modificación de Ficha Resumen”.
Sección “Modificar Ficha Resumen”	
Acción del Actor	Respuesta del Sistema.

2. El actor busca el proyecto a partir del filtro de búsqueda.	2.1 El sistema muestra datos de los proyecto según el filtro entrado por el actor. 2.1.1El sistema muestra el link del nombre en color verde cuando están en espera de aprobación o rechazo de una modificación por el usuario del sistema. 2.1.2 El sistema muestra el link del nombre en color rojo cuando están en espera de aprobación o rechazo de una modificación por el ente contraparte. 2.1.3 El sistema muestra el link de nombre en color azul (las dos partes están de acuerdo) cuando el usuario del sistema puede realizar modificaciones sobre el mismo.
3. El actor selecciona un proyecto con el nombre en azul o rojo	3.1 El sistema muestra los datos de la ficha (los datos son no editables)
4. El actor selecciona modificar la ficha	4.1 El sistema muestra una interfaz con los datos de la ficha y los componentes en el caso de los datos editables (Duración en meses, Modalidad, A transferir a Cuba y A invertir en Venezuela).
5. El actor edita los datos que desea modificar y acepta la operación	5.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
6. El actor selecciona aceptar.	6.1 El sistema actualiza la ficha en base de datos y regresa a la interfaz de búsqueda de proyecto terminado así el caso de uso.
Sección “Aprobar Modificación de Ficha Resumen”	
Acción del Actor	Respuesta del Sistema.
7. El actor busca el proyecto a partir del filtro de búsqueda.	7.1 El sistema muestra datos de los proyecto según el filtro entrado por el actor.
8. El actor selecciona un proyecto con el nombre en verde.	8.1 El sistema muestra los datos de la ficha (los datos son no editables).
9. El actor acepta los cambios propuestos por el EE contraparte	9.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
10. El actor selecciona aceptar.	10.1 El sistema actualiza la información en la base de datos y regresa a la interfaz de búsqueda de proyecto terminado así el caso de uso.
Sección “Rechazar Modificación de Ficha Resumen”	
Acción del Actor	Respuesta del Sistema.

11. El actor busca el proyecto a partir del filtro de búsqueda.	11.1 El sistema muestra datos de los proyecto según el filtro entrado por el actor.
12. El actor selecciona un proyecto con el nombre en verde.	12.1 El sistema muestra los datos de la ficha (los datos son no editables)
13. El actor rechaza los cambios propuestos por el EE contraparte	13.1 El sistema muestra un mensaje "Está seguro de realizar esta operación".
14. El actor selecciona Aceptar.	14.1 El sistema actualiza la información en la base de datos y regresa a la interfaz de búsqueda de proyecto terminado así el caso de uso.
Flujo alterno al paso 5,9,13	
Acción del Actor	Respuesta del Sistema.
4. ó 8. ó 11. El actor decide imprimir la Ficha Resumen.	4.1 ó 8.1 ó 11.1 El sistema imprime la Ficha Resumen a través de una impresora.
Flujo alterno al paso 4,8,11,5	
Acción del Actor	Respuesta del Sistema.
4., 8., 11. ó 5. El actor decide cancelar la operación.	4.1, 8.1, 11.1 ó 5.1 El sistema cancela la operación y regresa a la interfaz de búsqueda de proyecto.
Flujo alterno al paso 6,10,14	
Acción del Actor	Respuesta del Sistema.
6., 10., 14. El actor selecciona cancelar.	6.1, 10.1, 14.1 El sistema vuelve al estado anterior.
Pos condiciones	El sistema queda con las fichas actualizadas.
Especificación de datos de las interfaces del CU	

Interfaz "Listado de Proyecto"

Proyecto: Nombre de cualquier proyecto que el actor desee buscar. No se permite introducir ningún carácter extraño, solo letras, números, guiones.

Contraparte: Nombre del ente contraparte, este campo puede ser seleccionado por el actor del sistema y se carga desde la base de datos.

Estado: Es el estado en que se encuentra el proyecto, puede ser aprobado por Cuba y no aprobado por Venezuela, no aprobado por Cuba y aprobado por Venezuela, aprobado por ambas partes

Estos datos corresponden a la búsqueda que va a utilizar el actor para encontrar los proyectos que él como EE tiene con los entes ejecutores contraparte. Se mostrarán los proyectos según el filtro de búsqueda.

Nombre del proyecto: Nombre del proyecto que fue encontrado a partir de la búsqueda. Se muestra de color rojo cuando el ente ejecutor está esperando por la aprobación de una modificación del ente

ejecutor contraparte, en color azul cuando ambos entes están de acuerdo sobre la información, color verde cuando el ente contraparte está esperando por la aprobación de una modificación.

Contraparte: Nombre del ente contraparte que está ejecutando ese proyecto en específico.

Marco de aprobación: Nombre identificativo de la mixta donde fue presentado y aprobado el proyecto.

A transferir a Cuba: Monto del proyecto que se va a transferir a Cuba

A invertir en Venezuela: Monto que se va a invertir en Venezuela.

Total: Monto total del proyecto que representa la suma de A transferir a Cuba más A invertir en Venezuela.

Interfaz “Ficha Resumen del Proyecto”

Nombre del proyecto: Descrito en la interfaz anterior.

Ministerio venezolano: Nombre del ministerio al que pertenece el ente que está ejecutando el proyecto por la parte venezolana.

Ente venezolano: Nombre del ente responsable de la ejecución del proyecto por la parte venezolana.

Ministerio Cubano: Nombre del ministerio al que pertenece el ente que está ejecutando el proyecto por la parte cubana.

Ente cubano: Nombre del ente responsable de la ejecución del proyecto por la parte cubana.

Duración en meses: tiempo que dura el proyecto para terminarse, esto se mide en meses.

Marco de aprobación: Nombre identificativo de la mixta donde fue presentado y aprobado el proyecto.

Modalidad: Es el tipo de modalidad a la que pertenece el proyecto.

Estado: Descrito en la interfaz

Monto total: Lo mismo que total descrito en la interfaz anterior.

A transferir a Cuba: Monto del proyecto que se va a transferir a Cuba

A invertir en Venezuela: Monto que se va a invertir en Venezuela.

Porciento ejecución física: por ciento de la ejecución física por Venezuela y por Cuba, esto se obtiene

del estado en que estén las ejecuciones físicas de ambos países para el proyecto, entrados en el caso de uso Gestionar Ejecución Física.

Porciento ejecución financiera: por ciento de la ejecución financiera por Venezuela y por Cuba, esto se obtiene del estado en que estén las ejecuciones financieras de ambos países para el proyecto, entrados en el caso de uso Gestionar Ejecución Financiera.

Fondos: Tipo de fondo por el cual el proyecto es o será financiado.

Interfaz “Modificar Proyecto”

Nombre del proyecto: Descrito en la interfaz anterior. (Modificable). No Se permite introducir ningún carácter extraño, solo letras, números, guiones.

Ministerio venezolano: Nombre del ministerio al que pertenece el ente que está ejecutando el proyecto por la parte venezolana.

Ente venezolano: Nombre del ente responsable de la ejecución del proyecto por la parte venezolana.

Ministerio Cubano: Nombre del ministerio al que pertenece el ente que está ejecutando el proyecto por la parte cubana.

Ente cubano: Nombre del ente responsable de la ejecución del proyecto por la parte cubana.

Duración en meses: tiempo que dura el proyecto para terminarse, esto se mide en meses. (Modificable). Solo se permite introducir números enteros

Marco de aprobación: Nombre identificativo de la mixta donde fue presentado y aprobado el proyecto.

Modalidad: Es el tipo de modalidad a la que pertenece el proyecto. (Modificable)

A transferir a Cuba: Monto del proyecto que se va a transferir a Cuba. (Modificable). Se permite introducir solo el formato de dinero ejemplo: xxxxx,xx (solo dos lugares después de la coma).

A invertir en Venezuela: Monto que se va a invertir en Venezuela. (Modificable). Se permite introducir solo el formato de dinero ejemplo: xxxxx,xx (solo dos lugares después de la coma).

Fondos: Tipo de fondo por el cual el proyecto es o será financiado.

Interfaz “Confirmar Operación”

Esta interfaz se utiliza en todo el sistema, y se mostrará cada vez que usted vaya a realizar una operación sobre el mismo, el actor confirmaría o no la operación a través de la misma.

2.6.4 Gestionar Estado de Proyecto.

Caso de Uso:	Gestionar Estado de Proyecto.
Actores:	Usuario Ente Ejecutor.
Resumen:	El caso de uso se inicia cuando el usuario EE accede al sistema y realiza la búsqueda del proyecto para actualizar, aceptar o rechazar el estado del proyecto o cuando desde la interfaz de la ficha resumen del proyecto el actor decide ver la ejecución de un proyecto que esté aprobado por ambas partes (azul).
Precondiciones:	Que el usuario EE se haya autenticado. Exista conexión con el servidor. Existan fichas resúmenes en el sistema.
Referencias	RF02.008, RF02.008.01, RF02.008.02, RF02.009.01
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El actor selecciona en el menú de la página la opción o en la interfaz de la ficha resumen “Reporte de Ejecución”.	1.1 El sistema muestra la interfaz para buscar el o los proyecto según filtro de búsqueda.
2. El actor edita los datos del filtro y ejecuta la búsqueda	2.1 El sistema ubica los proyectos. 2.2 El sistema muestra los proyectos 2.2.1 Muestra el link de estado en color verde cuando el ente contraparte está en espera de aprobación o rechazo de una modificación ejecutada. 2.2.2 Muestra el link de estado en color rojo cuando el ente ejecutor está en espera de aprobación o rechazo de una modificación por el ente contraparte. 2.2.3 Muestra el link de estado en color azul cuando no existe ninguna modificación por ninguna de las dos partes (están de acuerdo las dos partes) a) Si desea actualizar el estado del Proyecto ver sección, “Modificar Estado del Proyecto”. b) Si desea aprobar la modificación del estado del proyecto, ver sección “Aprobar Modificación del Estado del Proyecto” c) Si desea rechazar modificación del estado del

	proyecto, ver sección “Rechazar Modificación del Estado del Proyecto”.
Sección “Modificar Estado del Proyecto”	
Acción del Actor	Respuesta del Sistema.
3. El actor decide Modificar el estado de proyecto.	3.1 El sistema muestra el estado del proyecto seleccionado por el actor.
4. El actor edita el estado deseado para el proyecto y acepta la modificación	4.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
5. El actor selecciona que si	5.1 El sistema modifica el estado y actualiza los datos en la base de datos terminando así el caso de uso
“Aprobar Modificación del Estado del Proyecto”	
Acción del Actor	Respuesta del Sistema
6. El actor decide aprobar el estado del proyecto modificado por el ente contraparte	6.1 El sistema muestra el estado del proyecto seleccionado por el actor
7. El actor selecciona que si acepta la modificación y acepta la operación.	7.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
8. El actor selecciona aceptar.	8.1 El sistema actualiza los cambios en la BD terminado así el caso de uso.
“Rechazar Modificación del Estado del Proyecto”.	
Acción del Actor	Respuesta del Sistema
9. El actor decide rechazar el estado del proyecto modificado por el ente contraparte	9.1 El sistema muestra el estado del proyecto seleccionado por el actor
10. El actor selecciona que no acepta la modificación y acepta la operación.	10.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
11. El actor selecciona aceptar.	11.1 El sistema actualiza los cambios en la BD terminado así el caso de uso.
Flujo Alterno al paso 3,6,9	
Acción del Actor	Respuesta del Sistema
3., 6. ó 9. El actor decide ver ficha del proyecto	3.1 ó 6.1 ó 9.1 El sistema muestra la ficha del proyecto terminado así el caso de uso.
Flujo Alterno al paso 4,7,10	
Acción del Actor	Respuesta del Sistema
4., 7. ó 10. El actor decide cancelar la operación	4., 7. ó 10. El sistema cancela la operación y regresa a la interfaz de reportar ejecución, terminado así el caso de uso
Flujo Alterno al paso 5,8,11	
Acción del Actor	Respuesta del Sistema
5., 8. ó 11. El actor selecciona que no	5., 8. ó 11. El sistema regresa al la interfaz de Estado.

Pos condición	El sistema queda con el estado de los proyectos modificados y aprobados.

Especificación de datos de las interfaces del CU

Interfaz “Reportar Ejecución”

Proyecto: Nombre de cualquier proyecto que el actor desee buscar. No se permite introducir ningún carácter extraño, solo letras, números, guiones.

Contraparte: Nombre del ente contraparte, este campo puede ser seleccionado por el actor del sistema y se carga desde la base de datos.

Estado: Es el estado en que se encuentra el proyecto, Iniciado, Contratado, Iniciado y Contratado ó Terminado. Esto es un link para ir a definir estado del proyecto.

Estos datos corresponden a la búsqueda que va a utilizar el actor para encontrar los proyecto que el cómo EE tiene con los entes ejecutores contraparte. Se mostrarán los proyectos según el filtro de búsqueda.

Nombre del proyecto: Nombre del proyecto que fue encontrado a partir de la búsqueda. Aquí se mostrarán solo los proyectos que cumpla con el filtro de búsqueda y además estén aprobados por ambas partes.

Contraparte: Nombre del ente contraparte que está ejecutando ese proyecto en específico.

Estado: Estado en que está el proyecto en este momento, estos estados pueden ser Iniciado, Contratado, Iniciado y Contratado ó Terminado. El estado que esté en rojo es porque el ente ejecutor está esperando por la aprobación de una modificación. El estado que está en azul es porque los dos entes están de acuerdo en la información correspondiente al estado. El estado que está en verde es porque el ente tiene que aprobarle una modificación al ente contraparte.

Interfaz “Definir Estado del Proyecto”

Nombre del proyecto: Nombre del proyecto que fue seleccionado por el actor.

Estado: Es el estado en que esta el proyecto este estado puede ser: Iniciado, Iniciado y Contratado, Contratado ó Terminado. El proyecto puede estar iniciado y contratado ó contratado ó Terminado, para poder terminar un proyecto debe estar iniciado, no así para contratado, puede estar en el estado contratado sin estar iniciado.

Fecha: Para cada modificación del estado del proyecto se registra la fecha que tiene que ser editado por el usuario, esta fecha tiene formato mes y año.

Si o No: Cuando un ente define un nuevo estado para el proyecto este entra en aprobación y por tanto se

habilita al ente contraparte el radio-button para seleccionar si acepta la nueva definición del estado o no, después que los dos entes se pusieron de acuerdo se deshabilita y no puede ser mas nunca editado por ninguno de los entes implicados en el proyecto.

Interfaz “Confirmar Operación”

Esta interfaz se utiliza en todo el sistema, y se mostrará cada vez que usted vaya a realizar una operación sobre el mismo, el actor confirmaría o no la operación a través de la misma.

Interfaz “Ficha Resumen del Proyecto”

Esta especificada en el caso de Uso Gestionar Ficha Resumen.

2.6.5 Gestionar Ejecución Financiera.

Caso de Uso:	Gestión de Ejecución Financiera.
Actores:	Usuario Ente Ejecutor.
Resumen:	El caso de uso se inicia cuando el usuario accede al sistema y dentro del reporte de ejecución desea actualizar, aprobar o rechazar Cronograma de Ejecución Financiera. El actor solo tiene derecho a modificar los datos referentes a su nacionalidad.
Precondiciones:	Que el usuario se haya autenticado. Exista conexión con el servidor. Existan fichas resúmenes en el sistema.
Referencias	RF02.007, RF02.007.01, RF02.007.02, RF02.009.01
Prioridad	Importante
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El actor selecciona en el menú de la página la opción (Reporte de Ejecución).	1.1 El sistema muestra la interfaz para buscar el o los proyecto según filtro de búsqueda.

2. El actor edita los datos del filtro y ejecuta la búsqueda	2.1 El sistema ubica los proyectos. 2.2 El sistema muestra los proyectos 2.2.1 Muestra el link de ejecución financiera en color verde cuando el ente contraparte está en espera de aprobación o rechazo de una modificación ejecutada. 2.2.2 Muestra el link de ejecución financiera en color rojo cuando el ente ejecutor está en espera de aprobación o rechazo de una modificación por el ente contraparte. 2.2.3 Muestra el link de ejecución financiera en color azul o negro cuando no existe ninguna modificación por ninguna de las dos partes (están de acuerdo las dos partes) a) Si desea modificar la ejecución financiera del Proyecto ver sección, "Modificar Ejecución Financiera del Proyecto". b) Si desea aprobar la modificación la ejecución financiera del proyecto, ver sección "Aprobar Modificación de Ejecución Financiera del Proyecto" c) Si desea rechazar modificación de la ejecución financiera del proyecto, ver sección "Rechazar Modificación de Ejecución Financiera del Proyecto".
Sección "Modificar Ejecución Financiera del Proyecto".	
Acción del Actor	Respuesta del Sistema
3. El Actor decide modificar la ejecución financiera del proyecto.	3.1 El sistema muestra la ejecución financiera tanto por la parte cubana como por la parte venezolana del proyecto seleccionado por el actor.
4. El actor edita la ejecución financiera de su parte correspondiente y adiciona el nuevo registro de ejecución. Indicando el monto ejecutado hasta el momento, esto lo hace considerando el dólar como moneda.	4.1 El sistema adiciona el nuevo registro y calcula el porciento que esto representa del total a transferir o invertir para la parte que está haciendo la entrada.
5. El actor acepta la modificación.	5.1 El sistema muestra un mensaje "Está seguro de realizar esta operación".
6. El actor selecciona aceptar.	6.1 El sistema modifica la ejecución y actualiza la información en base de datos terminado así el caso de

	uso.
Sección “Aprobar Modificación de Ejecución Financiera del Proyecto”	
Acción del Actor	Respuesta del Sistema
7. El actor selecciona la ejecución financiera de la parte venezolana en verde	7.1 El sistema muestra los datos referentes al cronograma de ejecución financiera del proyecto seleccionado.
8. El actor selecciona que si acepta la modificación realizada por la parte venezolana y acepta la operación.	8.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
9. El actor selecciona que aceptar.	9.1 El sistema actualiza la información en base de datos terminado así el caso de uso
Sección “Rechazar Modificación de Ejecución Financiera del Proyecto”.	
Acción del Actor	Respuesta del Sistema
10. El actor selecciona un proyecto en verde.	10.1 El sistema muestra la ejecución financiera tanto por la parte cubana como por la parte venezolana del proyecto seleccionado por el actor.
11. El actor selecciona que no acepta la modificación realizada por la parte venezolana y acepta la operación.	11.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
12. El actor selecciona que aceptar.	12.1 El sistema actualiza la información en base de datos terminado así el caso de uso
Flujo Alterno al paso 5	
Acción del Actor	Respuesta del Sistema
5. El actor decide eliminar el registro adicionado.	5.1 El sistema elimina el registro adicionado.
6.1 El actor acepta la operación	6.1 El sistema muestra un mensaje “Está seguro de realizar esta operación”.
7. El actor selecciona aceptar.	7.1 El sistema actualiza la información en base de datos terminado así el caso de uso
Flujo Alterno al paso 6, 7, 9, 12	
Acción del Actor	Respuesta del Sistema
6. ó 7. ó 9.1 ó 12. El actor decide cancelar.	6.1 ó 7.1 ó 9.1 ó 12. El sistema regresa al estado anterior.
Flujo Alterno al paso 4, 8, 11	
Acción del Actor	Respuesta del Sistema
4. ó 8. ó 11 El actor decide ver la ficha del proyecto.	El sistema muestra la ficha del proyecto terminado así el caso de uso.
Pos condición	El proyecto queda con el cronograma de ejecución financiera actualizado.
Especificación de datos de las interfaces del CU	

Interfaz “Reportar Ejecución”

Caso de que el ente autenticado en el sistema sea cubano

Ente cubano. Ejecución financiera: En este campo se mostrará en porciento la ejecución financiera del proyecto por la parte cubana. Cuando el dato esta en azul es que las dos partes están de acuerdo en la información de la ejecución financiera para cuba. Cuando está rojo, es que el ente cubano está en espera.

Ente venezolano. Ejecución financiera: En este campo se mostrará en porciento la ejecución financiera del proyecto por la parte venezolana. Cuando el dato está en negro es que las dos partes están de acuerdo en la información de la ejecución financiera para Venezuela. Cuando está verde, es que el ente venezolano está en espera de que el ente cubano apruebe alguna modificación.

Caso de que el ente autenticado en el sistema sea venezolano

Ente venezolano. Ejecución financiera: En este campo se mostrará en porciento la ejecución financiera del proyecto por la parte venezolana. Cuando el dato esta en azul es que las dos partes están de acuerdo en la información de la ejecución financiera para Venezuela. Cuando está rojo, es que el ente venezolano está en espera.

Ente venezolano. Ejecución financiera: En este campo se mostrará en porciento la ejecución financiera del proyecto por la parte cubana. Cuando el dato está en negro es que las dos partes están de acuerdo en la información de la ejecución financiera para Cuba. Cuando está verde, es que el ente cubano está en espera de que el ente venezolano apruebe alguna modificación.

Interfaz “Ejecución Financiera del Proyecto”

Nombre del proyecto: Nombre del proyecto seleccionado por el actor.

Porciento: Es el porciento que va a indicar el actor del sistema para el proyecto. Mientras el ente contraparte no apruebe este porciento puede ser eliminado por el ente cubano, así pasaría para el caso de Venezuela.

Cuando uno de los dos entes propone un nuevo por ciento, el ente contraparte puede aceptarlo o rechazarlo través del sí o el no, después de una de estas dos operaciones debe deshabilitarse esta opción.

2.6.6 Gestionar Ejecución Física.

Caso de Uso:	Gestionar Ejecución Física.
Actores:	Usuario Ente Ejecutor.
Resumen:	El caso de uso se inicia cuando el usuario accede al sistema y dentro de reporte de ejecución desea actualizar, aprobar o rechazar Cronograma de

	Ejecución física. El actor solo tiene derecho a modificar los datos referentes a su nacionalidad.
Precondiciones:	Que el usuario se haya autenticado. Exista conexión con el servidor. Existan fichas resúmenes en el sistema.
Referencias	RF02.006, RF02.006.01, RF02.006.02, RF02.009.01
Prioridad	Importante
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El actor selecciona en el menú de la página la opción (Reporte de Ejecución).	1.1 El sistema muestra la interfaz para buscar el o los proyecto según filtro de búsqueda.
2. El actor edita los datos del filtro y ejecuta la búsqueda	2.1 El sistema ubica los proyectos. 2.2 El sistema muestra los proyectos 2.2.1 Muestra el link de ejecución física en color verde cuando el ente contraparte está en espera de aprobación o rechazo de una modificación ejecutada. 2.2.2 Muestra el link de ejecución física en color rojo cuando el ente ejecutor está en espera de aprobación o rechazo de una modificación por el ente contraparte. 2.2.3 Muestra el link de ejecución física en color azul o negro cuando no existe ninguna modificación por ninguna de las dos partes (están de acuerdo las dos partes) a) Si desea modificar la ejecución física del Proyecto ver sección, “Modificar Ejecución física del Proyecto”. b) Si desea aprobar la modificación la ejecución física del proyecto, ver sección “Aprobar Modificación de Ejecución Física del Proyecto” c) Si desea rechazar modificación de la ejecución física del proyecto, ver sección “Rechazar Modificación de Ejecución Física del Proyecto”.
Sección “Modificar Ejecución Física del Proyecto”.	
Acción del Actor	Respuesta del Sistema
3. El Actor decide modificar la ejecución física del proyecto.	3.1 El sistema muestra la ejecución física tanto por la parte cubana como por la parte venezolana del proyecto seleccionado por el actor.

4. El actor edita la ejecución física de su parte correspondiente y adiciona el nuevo registro de ejecución.	4.1 El sistema adiciona el nuevo registro
5. El actor acepta la modificación	5.1 El sistema muestra un mensaje "Está seguro de realizar esta operación".
6. El actor selecciona aceptar.	6.1 El sistema modifica la ejecución y actualiza la información en base de datos terminado así el caso de uso.
Sección "Aprobar Modificación de Ejecución Física del Proyecto"	
Acción del Actor	Respuesta del Sistema
7. El actor selecciona la ejecución física de la parte venezolana en verde	7.1 El sistema muestra los datos referentes al cronograma de ejecución física del proyecto seleccionado.
8. El actor selecciona que si acepta la modificación realizada por la parte venezolana y acepta la operación.	8.1 El sistema muestra un mensaje "Está seguro de realizar esta operación".
9. El actor selecciona aceptar.	9.1 El sistema actualiza la información en base de datos terminado así el caso de uso.
Sección "Rechazar Modificación de Ejecución Física del Proyecto".	
Acción del Actor	Respuesta del Sistema
10. El actor selecciona un proyecto en verde.	10.1 El sistema muestra la ejecución física tanto por la parte cubana como por la parte venezolana del proyecto seleccionado por el actor.
11. El actor selecciona que no acepta la modificación realizada por la parte venezolana y acepta la operación.	11.1 El sistema muestra un mensaje "Está seguro de realizar esta operación".
12. El actor selecciona aceptar.	12.1 El sistema actualiza la información en base de datos terminado así el caso de uso
Flujo Alterno al paso 5	
Acción del Actor	Respuesta del Sistema
5. El actor decide eliminar el registro adicionado.	5.1 El sistema elimina el registro adicionado.
6.1 El actor acepta la operación	6.1 El sistema muestra un mensaje "Está seguro de realizar esta operación".
7. El actor selecciona aceptar.	7.1 El sistema actualiza la información en base de datos terminado así el caso de uso
Flujo Alterno al paso 6, 7, 9, 12	
Acción del Actor	Respuesta del Sistema
6. ó 7. ó 9.1 ó 12. El actor decide cancelar	6.1 ó 7.1 ó 9.1 ó 12. El sistema regresa al estado anterior.
Flujo Alterno al paso 4, 8, 11	

Acción del Actor	Respuesta del Sistema
4. ó 8. ó 11 El actor decide ver la ficha del proyecto.	El sistema muestra la ficha del proyecto terminado así el caso de uso.
Pos condición	El proyecto queda con el cronograma de ejecución física actualizado.
Especificación de datos de las interfaces del CU	
Interfaz “Reportar Ejecución” Caso de que el ente autenticado en el sistema sea cubano Ente cubano. Ejecución física: En este campo se mostrará en porciento la ejecución física del proyecto por la parte cubana. Cuando el dato esta en azul es que las dos partes están de acuerdo en la información de la ejecución física para cuba. Cuando está rojo, es que el ente cubano está en espera. Ente venezolano. Ejecución física: En este campo se mostrará en porciento la ejecución física del proyecto por la parte venezolana. Cuando el dato está en negro es que las dos partes están de acuerdo en la información de la ejecución física para Venezuela. Cuando está verde, es que el ente venezolano está en espera de que el ente cubano apruebe alguna modificación. Ente venezolano. Ejecución física: En este campo se mostrará en porciento la ejecución física del proyecto por la parte venezolana. Cuando el dato esta en azul es que las dos partes están de acuerdo en la información de la ejecución física para Venezuela. Cuando está rojo, es que el ente venezolano está en espera. Ente cubano. Ejecución física: En este campo se mostrará en porciento la ejecución física del proyecto por la parte cubana. Cuando el dato está en negro es que las dos partes están de acuerdo en la información de la ejecución física para Venezuela. Cuando está verde, es que el ente cubano está en espera de que el ente venezolano apruebe alguna modificación. Interfaz “Ejecución Física del Proyecto” Nombre del proyecto: Nombre del proyecto seleccionado por el actor. Porciento: Es el porciento que va a indicar el actor del sistema para el proyecto. Mientras el ente contraparte no apruebe este porciento puede ser eliminado por el ente cubano, así pasaría para el caso de Venezuela. (Editable). Se puede introducir solo números, puede ser con comas. Cuando uno de los dos entes propone un nuevo por ciento, el ente contraparte puede aceptarlo o rechazarlo través del sí o el no, después de una de estas dos operaciones debe deshabilitarse esta opción.	

Para ver las especificaciones de los casos de uso restante consulte el **Anexo 1.**

2.7 Conclusiones

Al finalizar este capítulo se detalló en los objetivos estratégicos y se logró conformar un modelo de dominio donde se identificaron las principales clases y conceptos enmarcados en las comisiones MIXTAS Cuba-Venezuela, se logró un buen refinamiento de los requisitos que fueron capturados, se identificaron los actores que serán los encargados de interactuar con el sistema, se generó el diagrama de CUS y su especificación. Se definieron las Reglas de Negocio del módulo que serán restricciones explicitas de comportamiento y proporcionarán un soporte para la dirección de las actividades del Convenio Cuba-Venezuela. Como resultado de este análisis se deriva el diseño de un sistema automatizado, partiendo de los requerimientos funcionales y no funcionales, que permitirá al equipo de desarrollo una mejor comprensión del mismo facilitando así la implementación del Módulo.

CAPITULO 3

CAPITULO 3. ANÁLISIS Y DISEÑO

3.1 Introducción

En el presente capítulo se expone el análisis y diseño para la solución de la aplicación, modelándose los artefactos necesarios que contribuyen y sirven como plano a la implementación. Se hace un estudio detallado del modelo de caso de uso del sistema con el objetivo de esbozar las clases a implementar a través del Análisis, las que finalmente se representarán en un diagrama de clases Web del Diseño.

3.2 Modelo de análisis

En el modelo de análisis se refinan los requisitos, no se toma en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reutilizables de otras aplicaciones, entre otras características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye El Diagrama de Clases del Análisis.

3.2.1 Diagrama de clases de análisis

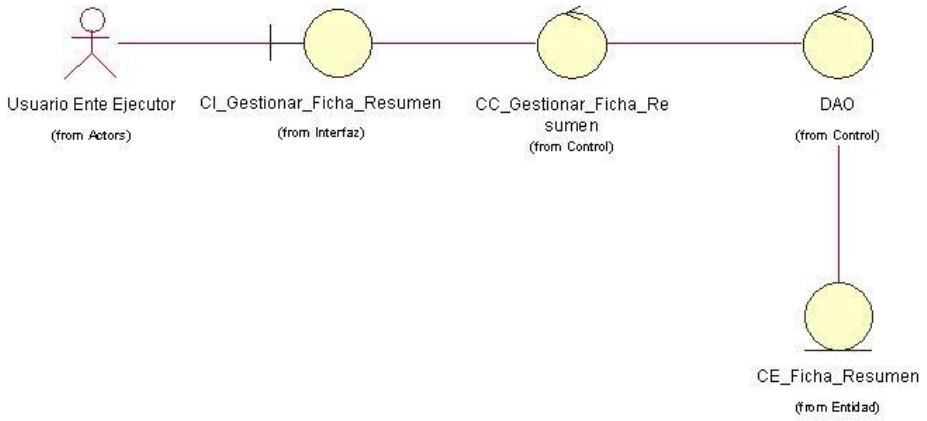
El Diagrama de Clases de Análisis es un artefacto en el que se representa los conceptos fundamentales en un dominio del problema. Los diagramas de clases de análisis, representan las definiciones y relaciones entre las clases. Las clases del análisis se clasifican en Interfaz, de Control o Entidad.

- Clase de Interfaz: Modelan la interacción entre el sistema y sus actores.
- Clase Entidad: Modelan información que posee una larga vida.
- Clase de Control: Representan coordinación, secuencia, transacciones, y control de otros objetos y a menudo encapsula a un caso de uso en concreto.

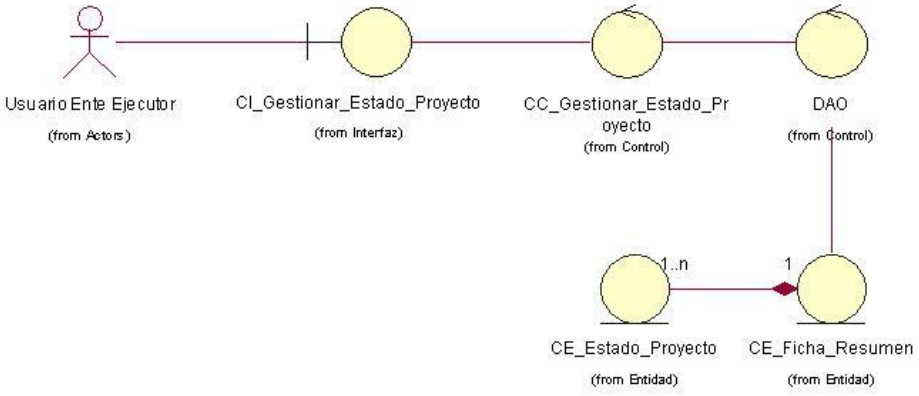
Son diagramas estáticos que muestran **"qué"** es lo que interactúa, pero no **"cómo"** interactúa o **"qué pasa"** cuando ocurre la interacción.

Para ver los restantes diagramas de clases de análisis consulte el **Anexo 2.1.**

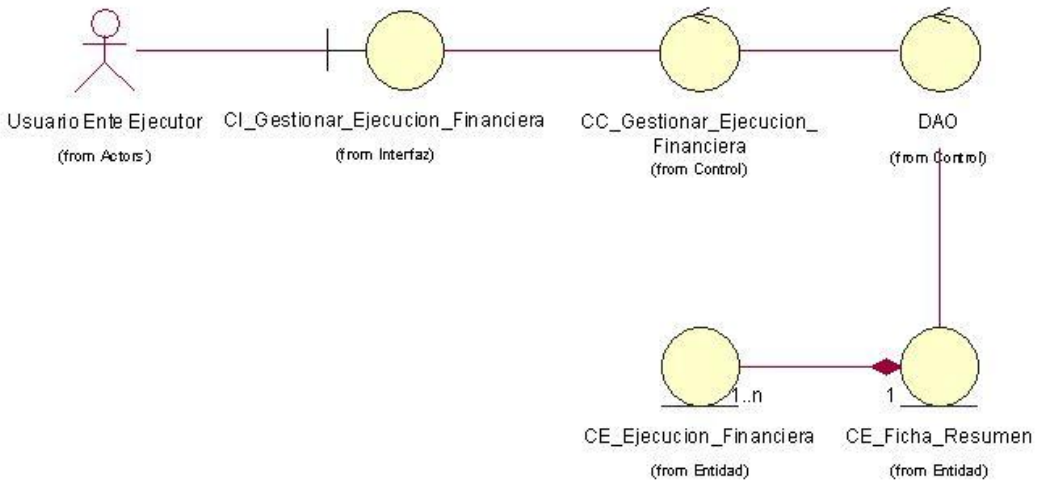
3.2.1.1 DCA del CUS Gestionar Ficha Resumen



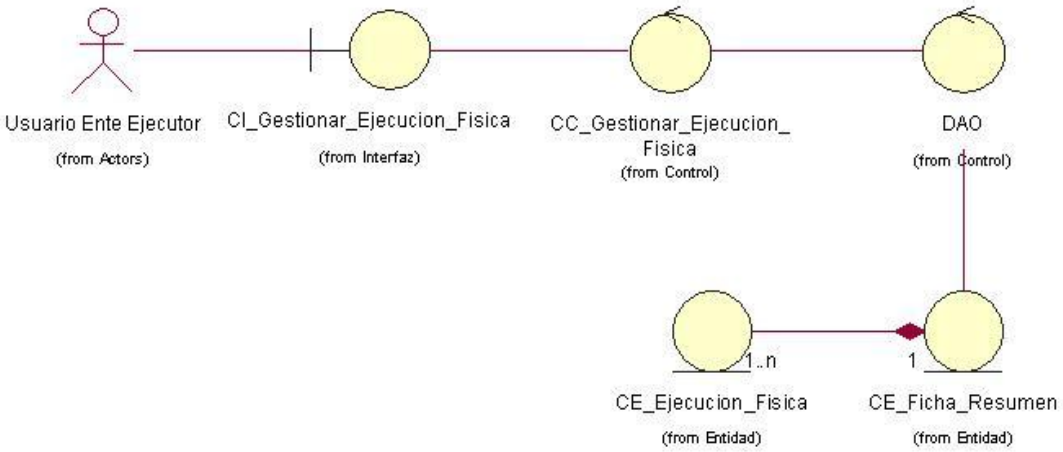
3.2.1.2 DCA del CUS Gestionar Estado de Proyecto



3.2.1.3 DCA del CUS Gestionar Ejecución Financiera



3.2.1.4 DCA del CUS Gestionar Ejecución Física



3.2.2 Diagrama de interacción

Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.
Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes.
Los diagramas de interacción pueden utilizarse para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o se pueden utilizar para modelar un flujo de control particular de un caso de uso. Representa la forma en como un Cliente (Actor) u Objetos (Clases) se comunican entre si en petición a un evento.
Para ver los diagramas de interacción ver **Anexo 2.2.**

3.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación.

3.3.1 Diagrama de clases de diseño

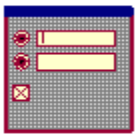
“Una clase de diseño es una abstracción de una clase o construcción en la implementación del sistema”. (Ivar Jacobson, 2000)
Un diagrama de clases de diseño es un diagrama que muestra un conjunto de interfaces, colaboraciones y sus relaciones. Los diagramas de clases de diseño se utilizan para modelar principalmente la vista de diseño estática de un sistema. Esto incluye modelar el vocabulario del sistema, las colaboraciones o esquemas.
Los diagramas de clases, son importantes, no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

3.3.2 Diagramas de clases de diseño utilizando extensiones UML para Web

UML posee una extensión para el modelado de aplicaciones Web, usada para el diseño de las clases, dicha extensión utiliza diferentes estereotipos que permiten definir un nuevo significado de la semántica para el elemento a modelar, los estereotipos más usados son:

<Server Page> Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página. Esta clase sólo puede tener relaciones con objetos en el servidor.

<Client Page> Una instancia de Página Cliente es una página Web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para páginas cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.



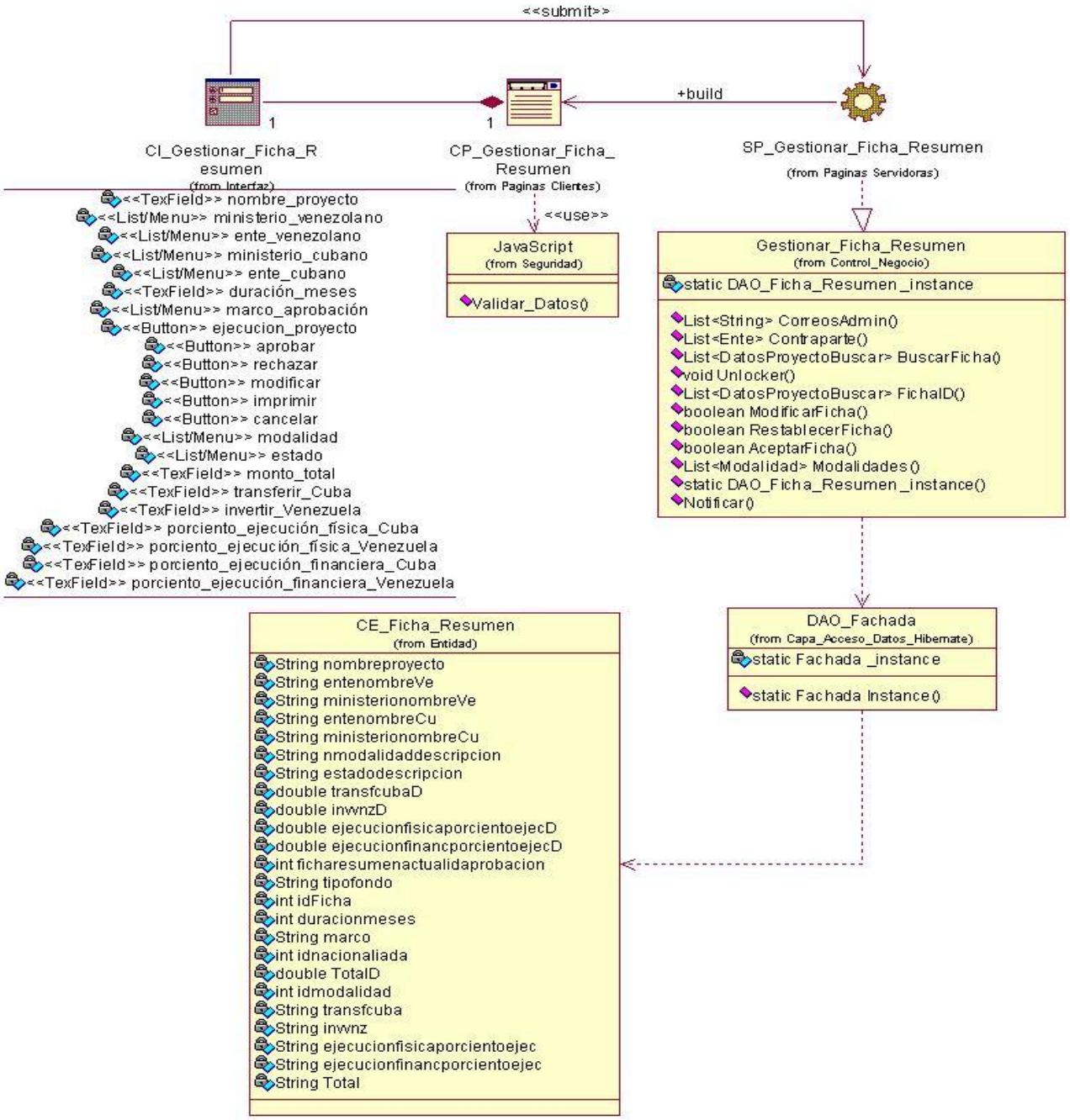
<Form> Colección de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y Hidden Fields).

<Build> Representa una asociación especial que relaciona las páginas cliente con las páginas servidor, de forma general se expresa como que las páginas que se encuentran en el servidor construyen las páginas en el cliente. Debe ser una relación direccional, donde una página servidor puede construir una o más páginas cliente.

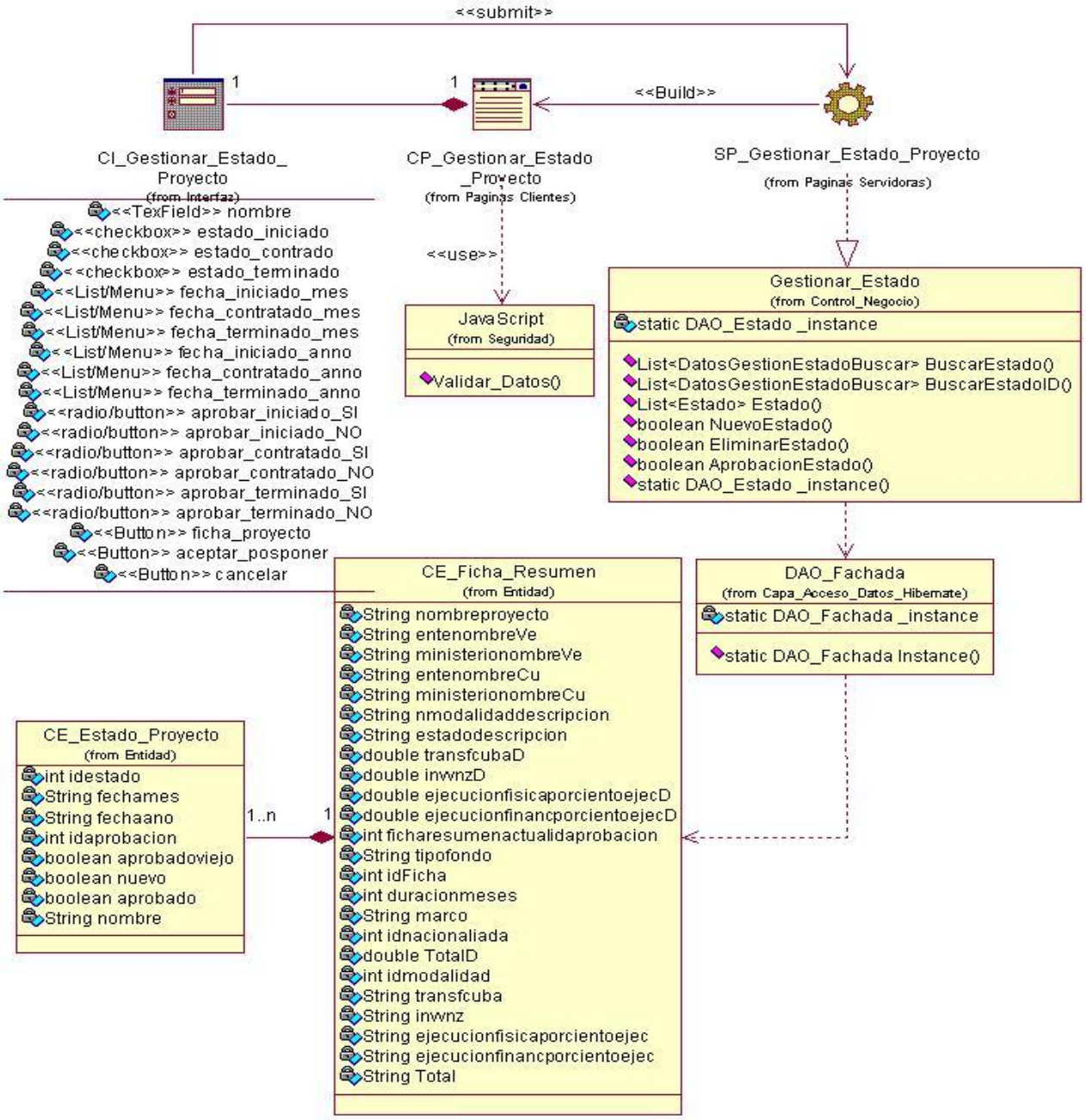
<Submit> Es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.

Para ver los diagramas de clases de diseño consulte el **Anexo 2.3.**

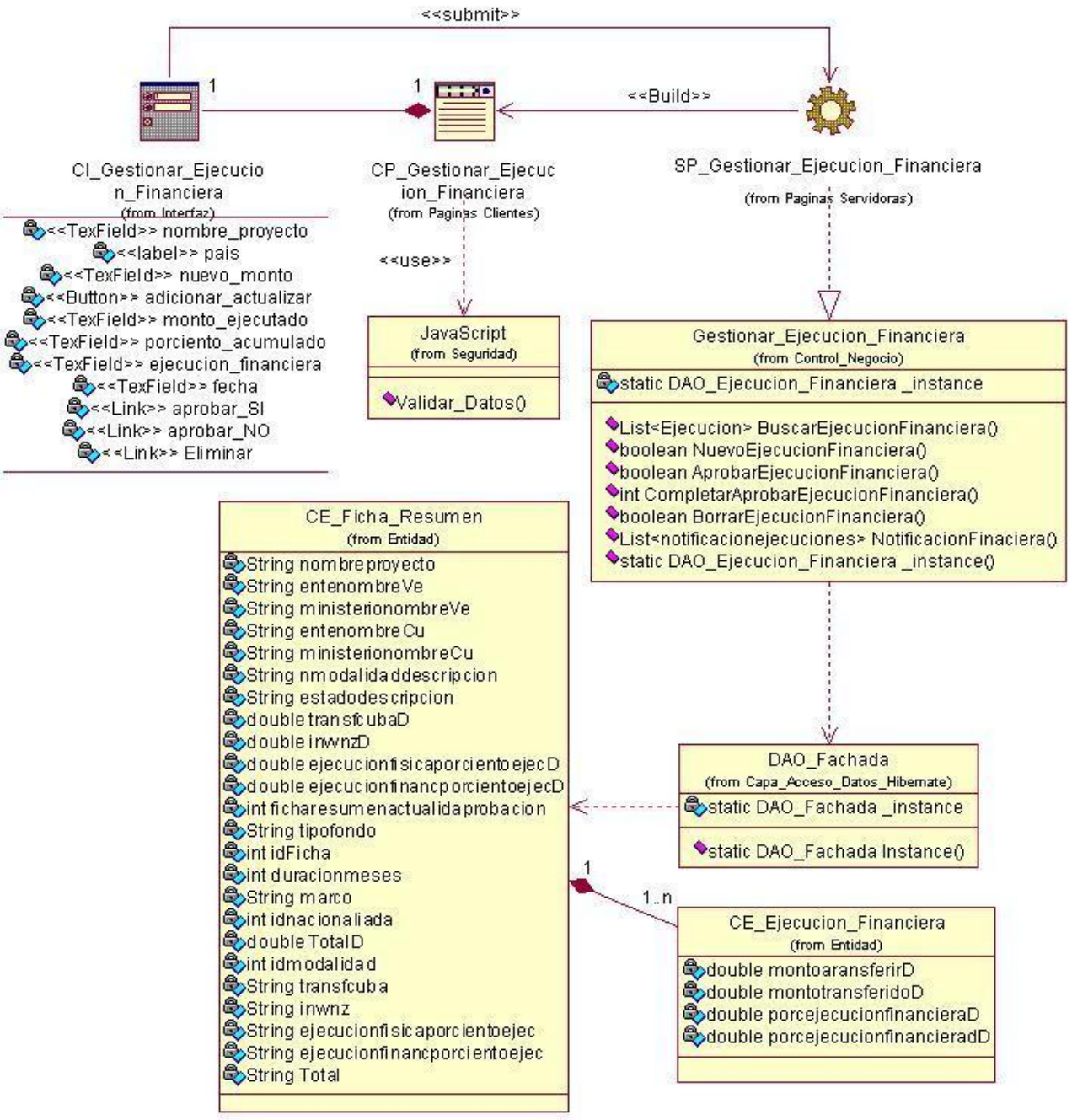
3.3.4 DCD del CUS Gestionar Ficha Resumen



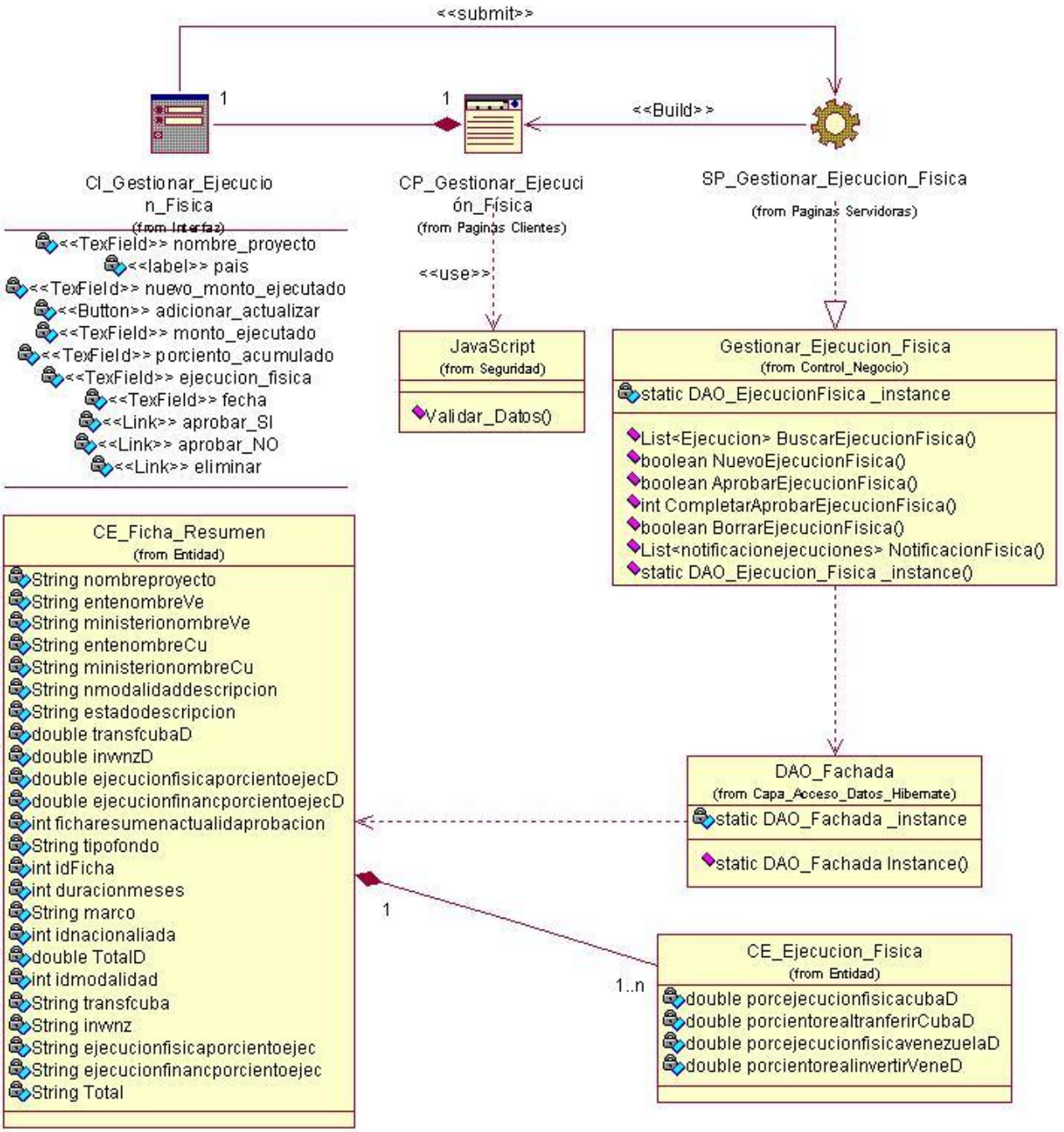
3.3.5 DCD del CUS Gestionar Estado de Proyecto



3.3.6 DCD del CUS Gestionar Ejecución Financiera



3.3.7 DCD del CUS Gestionar Ejecución Física



3.4 Patrones

Para hacer un diseño eficiente se tomaron en cuenta un conjunto de patrones, que al ser experiencias de diseñadores expertos en orientación a objetos permiten dar solución a problemas a través de la codificación del conocimiento y principios existentes, facilitando notablemente el trabajo posterior. Los patrones son:

- Soluciones concretas: proponen soluciones a problemas concretos, son teorías genéricas.
- Soluciones técnicas: indican soluciones basadas en programación orientada a objetos (OO).
- Se utilizan en situaciones frecuentes: se basan en la experiencia acumulada por los que resuelven problemas reiterativos.

Se utilizaron fundamentalmente tres tipos de patrones:

1. Patrones de organización: patrón de tres capas.
2. Patrones de distribución: cliente/servidor. Básicamente este patrón se definió al escoger la plataforma de desarrollo Web, explicado en el capítulo 1 del presente trabajo.
3. Patrones de diseño: GRASP y GoF.

El patrón tres capas se utiliza en la arquitectura de tres capas ya que ofrece entre sus principales ventajas las que se listan a continuación: (Larman, 2003)

- Aísla la lógica de la aplicación y la convierte en una capa intermedia bien definida y lógica del software.
- En la capa de presentación se realiza relativamente poco procesamiento de la aplicación.
- Simplifica la comprensión y organización del desarrollo del sistema, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle de las superiores.
- Facilita la reutilización.

1. La capa de presentación o interfaz de usuario.

En este caso, está formada por los formularios **.html** y sus controles o atributos, las **client page** y las **server page**. Está definida como la capa con la que interactúa el usuario.

2. La capa de negocio.

Esta capa está formada por las entidades que se definen en el sistema, que representan objetos que van a ser manejados o consumidos por toda la aplicación y por las clases **.java** que se encuentran dentro del negocio de la aplicación.

3. La capa de acceso a datos.

Contiene clases que interactúan con la base de datos, estas clases altamente especializadas permiten, utilizando los procedimientos almacenados generados, realizar todas las operaciones con la base de datos de forma transparente para la capa de negocio. Están definidas por las clases **DAO** del sistema.

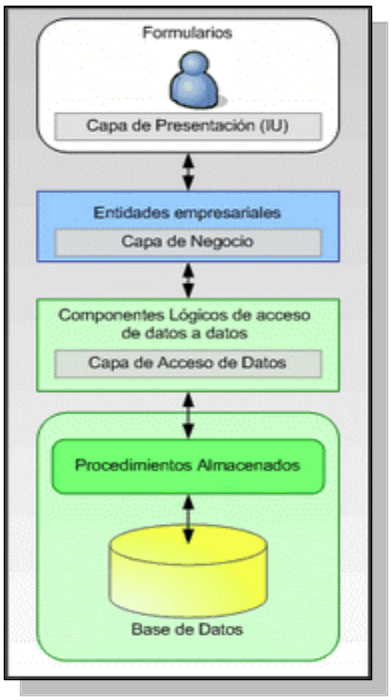


Figura 10. Tres Capas

3.5 Descripción de la arquitectura propuesta

La propuesta se apoya en los siguientes frameworks de desarrollo para implementar las principales capas.

1. Spring MVC para la capa de presentación.
2. Spring para gestionar la lógica de negocio y otras funciones a nivel de aplicación como la seguridad, las trazas y como enlace entre la capa de acceso a datos y la capa de presentación.
3. Hibernate, como mecanismo persistente.

Además se utiliza como entorno de ejecución el contenedor web Tomcat.

El diseño debe ser específico al problema que se tiene entre manos, pero suficientemente general para adaptarse a problemas y requerimientos futuros. Los diseñadores experimentados en OO dicen que un diseño reusable y flexible es difícil, si no imposible de obtener bien, la primera vez. Antes de que un diseño sea terminado, usualmente tratan de reutilizarlo varias veces, modificándolo cada vez. (Gamma, 1995)

Partiendo de esta idea y apoyados en los patrones descritos en el epígrafe anterior se obtiene un diseño estructurado en cinco capas reflejado en la Fig.11.

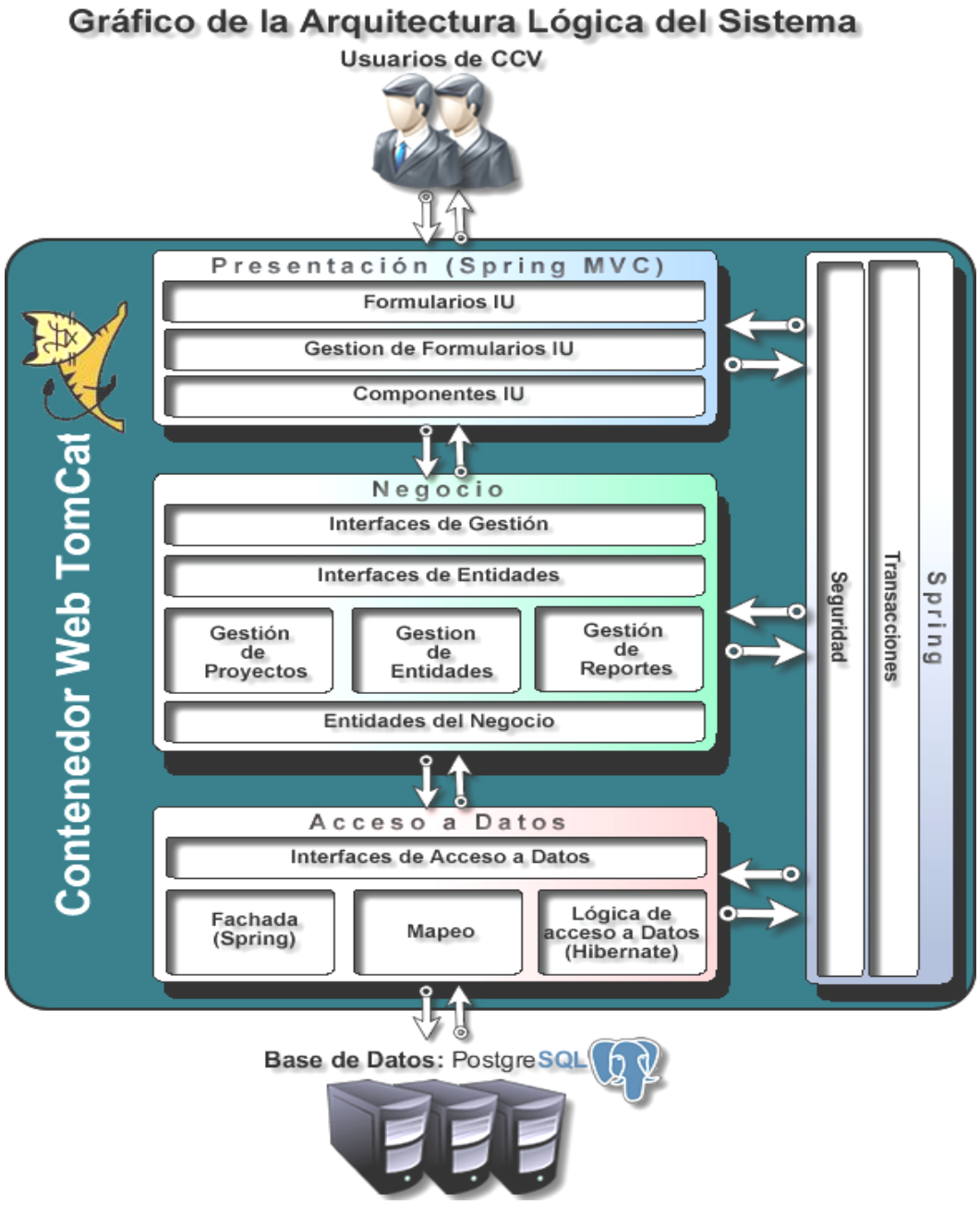


Figura11. Gráfico de Arquitectura Lógica del Sistema

3.5.1 Patrones aplicados

Los patrones GRASP (**G**eneral **R**esponsability **A**signment **S**oftware **P**atterns, Patrones de los Principios Generales para Asignar Responsabilidades) permiten asignar correctamente las responsabilidades a cada una de las clases que intervienen en el modelo. Se utilizaron cuatro de los cinco patrones GRASP fundamentales:

- Experto: se asignaron responsabilidades a las clases con la información necesaria para cumplirla.
- Creador: se asignaron responsabilidades a las clases de crear instancias de otras conociendo que las primeras son las que contienen la información para ello.
- Alta cohesión: se asignaron responsabilidades a las clases de manera que todos sus métodos tuvieran un comportamiento bien definido.
- Bajo acoplamiento: cada clase está acoplada a las clases estrictamente necesarias.

De los patrones GoF explicado anteriormente en el capítulo 1 se aplicaron al diseño de clases los siguientes:

- Singleton: asegura que una clase tendrá solo una instancia, y provee un mundo global de acceso a la misma. El constructor es privado y el método **instance()** es el que devuelve la única instancia de esta clase, o la crea si no existe.

Figura12. Estructura General del Patrón.

- Facade: Típicamente, un diseño adecuado llevará a una factorización en clases más allá de la estrictamente necesaria para un sistema determinado. En ese caso, es posible construir una clase facade, (fachada), que tenga la interfaz esperada y sea el que realmente se comunique con la estructura de clases real, cuyas interfaces pueden variar.

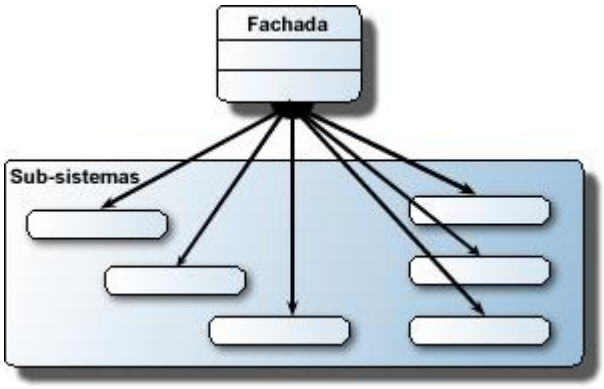


Figura 13. Estructura del Patrón de Diseño Facade.

La clase fachada se interpone entre el cliente (no mostrado) y el subsistema que proporcione la funcionalidad deseada. Facade simplifica la interfaz del subsistema.

- Observer: Define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes.

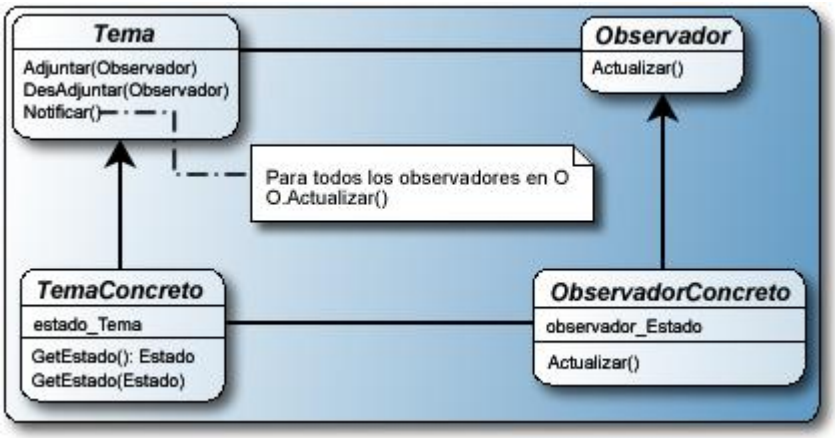


Figura 14. Estructura del Patrón de Diseño Observer.

- El objetivo de este patrón es desacoplar la clase de los objetos clientes del objeto, aumentando la modularidad del lenguaje, así como evitar bucles de actualización.

3.6 Diseño de la Base de Datos

Para desarrollar un buen modelo de datos RUP propone que las bases de datos (BD) necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido, y recuperar la

información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usaran. La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones de soporte del negocio o dominio en cuestión.

A continuación se listan los pasos que guían el diseño de la base de datos a partir de un modelo orientado a objetos.

Los pasos en el diseño de la base de datos son:

1. Definir las clases persistentes.
2. Refinar las clases.
3. Clasificar las clases y los atributos.
4. Realizar el diagrama de clases.
5. Obtener las restricciones estáticas y las fórmulas dinámicas.
6. Convertir las clases al medio de almacenamiento.

Para ver el diseño de la base de datos consulte el **Anexo 2.4.**

3.7 Conclusiones

En este capítulo con el apoyo de las características del sistema antes definidas, se realizó una modelación del mismo en términos del análisis y diseño de los casos de uso como respuesta a la solución propuesta, generándose los artefactos necesarios en el flujo de trabajo que propone RUP, identificándose las clases interfaces, de control y de entidad. Se obtuvo la realización de los casos de uso en el análisis y el diseño mostrando como resultado los diagramas de clases de análisis, un diagrama de interacción por cada escenario de caso de uso y los diagramas de clases del diseño. Se aplicaron los patrones necesarios para logra un diseño eficiente y se estableció la arquitectura lógica del sistema para una mejor comprensión de su estructura.

CAPITULO 4

CAPITULO 4. ANÁLISIS DE LA FACTIBILIDAD.

4.1 Introducción

Para un buen análisis y diseño de software, no solo debe tenerse en cuenta el problema a resolver, la información que se maneja y las potencialidades informáticas de implementación, también corresponde llevar presente una adecuada planificación y evaluación de los costos para la culminación de un producto en tiempo deseado y utilización de los recursos que realmente sean necesarios, razón por la que es de suma importancia demostrar su factibilidad, mediante métodos que permitan analizar el costo y los beneficios que este reportará. De esta misma forma, las métricas del diseño juegan un papel importante en la validación de la calidad del mismo, proporcionando una mejor visión interna y ayudando a mejorar el software de forma continua.

4.2 Factibilidad del análisis

Algunos de los métodos más usados son:

Método de Casos de Usos, Análisis de Puntos de Función y el COCOMO II.

El método de Casos de Usos es uno de los métodos más efectivos para capturar la funcionalidad de un sistema, este método permite documentar los requerimientos de un sistema en términos de Actores y Casos de Uso. (Peralta, 2004)

Para la estimación del tamaño de un sistema a partir de sus requerimientos, es el de Análisis de Puntos de Función. Ésta técnica permite cuantificar el tamaño de un sistema en unidades independientes del lenguaje de programación, las metodologías, plataformas y/o tecnologías utilizadas, denominadas Puntos de Función. (Peralta, 2004)

Por otro lado, el SEI (Software Engineering Institute) propone desde hace algunos años un método para la estimación del esfuerzo llamado COCOMO II. Este método está basado en ecuaciones matemáticas que permiten calcular el esfuerzo a partir de ciertas métricas de tamaño estimado, como el Análisis de Puntos de Función y las líneas de código fuente (en inglés SLOC, Source Line Of Code). (Peralta, 2004).

El objetivo fundamental de la planificación y el análisis de la factibilidad es establecer planes razonables para desarrollar la Ingeniería de Software y manejar los cambios de los proyectos de Software (Giraldo, 2007).

4.3 Estimación basada en Casos de Uso

En este capítulo se hace un estudio de factibilidad utilizando el método de Puntos de Casos de Usos que consiste en la realización de una secuencia de pasos que se desarrollan a continuación:

4.3.1 UUCP (Cálculo de puntos de Casos de Uso sin ajustar)

El primer paso para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin ajustar. Este valor, se calcula a partir de la siguiente ecuación:

UUCP = UAW + UUCW

Donde,

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin Ajustar

4.3.1.1 UAW (Factor de peso de los actores sin ajustar)

Para determinar el UAW se ha de conocer el total de actores que interactúan con el sistema y la complejidad de los mismos, esta última se determina teniendo en cuenta si dicho actor representa a una persona u otro sistema y la forma en que el mismo interactúa con el software.

Los criterios se muestran en la siguiente tabla:

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface).	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

Tabla 3. Factor de Peso de los actores según su complejidad.

Tomando en cuenta que el actor que inicializa todos los casos de uso del sistema es (una persona) y su clasificación según la tabla antes analizada, se llega a la conclusión de que se tiene un actor complejo y por tanto un factor de peso 3.

Clasificación de la complejidad de los actores de acuerdo a la naturaleza de los mismos.

Actor	Complejidad	Peso
Usuario	Complejo	3

Tabla 4. Complejidad del actor Usuario.

UAW = Σcant actores * peso

UAW = 4 x 3 = 12

4.3.1.2 UUCW (Factor de peso de los Casos de Uso sin ajustar)

Con el UUCW sucede lo mismo, pues está determinado por la cantidad de Casos de Uso en el sistema y la complejidad de los mismos. Luego la complejidad de los Casos de Uso depende del número de transacciones que contengan cada uno. Las transacciones son las secuencias de actividades atómicas.

Tipos de Caso de Uso	Descripción	Factor de Peso
Simple	El Caso de Uso que contiene de 1 a 3 transacciones.	5
Medio	El caso de Uso que contiene de 4 a 7 transacciones.	10
Complejo	El Caso de Uso que contiene más de 8 transacciones.	15

Tabla 5. Clasificación de la complejidad de los Casos de Uso de acuerdo al número de transacciones.

Caso de Uso	Transacciones	Factor de peso
Gestionar Usuario	8	15
Gestionar Ente Ejecutor	8	15
Gestionar Ministerio	8	15
Gestionar Ficha Resumen	14	15
Gestionar Estado de Proyecto	12	15
Gestión de Ejecución Financiera	13	15
Gestionar Ejecución Física	13	15
Gestionar Reporte General	3	5

Gestionar Reporte de Ejecución Física	3	5
Reporte de Ejecución Financiera	3	5

Tabla 6. Complejidad de los Casos de Uso del Sistema.

Atendiendo la tabla anterior se tienen 3 casos de uso de complejidad simple y factor de peso 5, y 7 de complejos por ende factor de peso 15, para un total de 10 casos de uso. De aquí que:

UUCW = Σ (cant. CU (pesoA) x pesoA)

UUCW = (3 x 5) + (7 x 15) = 120

Una vez conocidos los valores de los factores de peso de los actores y de los casos de uso sin ajustar **(UAW y UUCW)** se pueden calcular los Puntos de Casos de Uso sin ajustar **(UUCP):**

UUCP = UAW + UUCW

UUCP = 12 + 120

UUCP = 132

4.3.2 Cálculo de UCP (Puntos de Casos de Uso ajustados)

El próximo paso es obtener el valor de los Puntos de Casos de Uso ajustados. Esto es posible a través de la siguiente ecuación:

UCP = UUCP x TCF x EF

Donde:

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

4.3.2.1 Factor de TCF (Complejidad técnica)

Existen una serie de factores que determinan la complejidad técnica del sistema, la cuantificación de estos contribuyen a obtener el valor del TCF. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde:

0: No presente o sin influencia,

1: Influencia incidental o presencia incidental

2: Influencia moderada o presencia moderada

3: Influencia media o presencia media

4: Influencia significativa o presencia significativa

5: Fuerte influencia o fuerte presencia

Factor	Descripción	Peso	Valor	Σ(peso x valor)
T1	Sistema distribuido	2	0	0
T2	Objetivos de performance (funcionamiento) o tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	3	3
T4	Procesamiento interno complejo	1	2	2
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0.5	3	1.5
T7	Facilidad de uso	0.5	4	2
T8	Portabilidad	2	5	10
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	4	4
T11	Incluye objetivos especiales de seguridad	1	5	5
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	3	3

Tabla 7. Clasificación del Factor de Complejidad Técnica del Sistema.

TCF = 0.6 + 0.01 x Σ (Peso i x Valor i)

Según los datos de la tabla anterior se tiene que:

TCF = 0.6 + 0.01 x Σ (Peso i x Valor i)

TCF = 0.6 + 0.01 x Σ (0 + 4 + 3 + 2 + 4 + 1.5 + 2 + 10 + 3 + 4 + 5+ 0 + 3)

TCF = 0.6 + 0.01 x 41.5

TCF = 1.015

4.3.2.2 Cálculo del EF (Factor de ambiente)

Para determinar el factor ambiente hay que tener en cuenta las habilidades y experiencia del conjunto de personas que van a desarrollar el producto. Estos factores ejercen una gran influencia a la hora de realizar estimaciones de tiempo y se cuantifican con valores de 0 a 5, utilizando la ecuación

EF =1.4 - 0.03 x Σ (Peso i x Valor i)

En la siguiente tabla se muestra el valor y el peso de cada uno de éstos factores:

Factor	Descripción	Peso	Valor	Σ(peso x valor)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3	4.5
E2	Experiencia en la aplicación	0.5	1	1.5
E3	Experiencia en orientación a objetos	1	4	3
E4	Capacidad del analista líder	0.5	3	1.5
E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	3	6
E7	Personal a tiempo compartido	-1	2	-2
E8	Dificultad del lenguaje de programación	-1	3	-3

Tabla 8. Clasificación del Factor de Ambiente del Sistema.

- Para los factores E1 al E4, los valores asignados de 0 significan sin experiencia, 3 con experiencia media y 4 experiencia media-avanzada.
- Para el factor E5, 5 significa alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 5 significa que todo el personal trabaja a tiempo compartido (nadie trabaja a tiempo completo).
- Para el factor E8, 3 significa que el lenguaje de programación es medianamente fácil de usar.

Según los datos recogidos en la tabla anterior, se puede decir que:

EF =1.4 - 0.03 x Σ (Peso i x Valor i)

EF =1.4 - 0.03 x Σ (4.5 + 1.5 + 3 + 1.5 + 5 + 6 - 2 - 3)

EF =1.4 - 0.03 x 16.5

EF = 0.91

Teniendo ya todos los valores que conforman la parte derecha de la ecuación **(UUCP, TCF, EF)** para calcular los Casos de Uso Ajustados **(UCP)**, se pasa a resolver la misma.

UCP = UUCP x TCF x EF

UCP = 132 x 1.015 x 0.91

UCP = 121.2519

4.3.3 (E) Cálculo del esfuerzo en horas – hombres

Cuando se habla de esfuerzo en este contexto se refiere a la relación entre la cantidad de hombres y el tiempo. Para hallar este valor depende del valor de los Puntos de Casos de Uso Ajustados y el Factor de Conversión, como lo muestra la ecuación:

E = UCP x CF

Donde:

E: Esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso Ajustados

CF: Factor de Conversión

4.3.3.1 Conversión de los puntos de Casos de Uso ajustados a esfuerzo de desarrollo

Para la búsqueda del Factor de Conversión se siguen una serie de pasos que contribuirán a la toma de un criterio:

Paso 1 - Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.

Para este caso la cantidad de factores por debajo del valor medio es 1.

Paso 2 - Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

Para este caso la cantidad de factores por encima del valor medio es 0.

Luego para calcular **CF**:

Se tiene que:

CF = 20 horas-hombre (si Total $_{EF}$ ≤ 2)

CF = 28 horas-hombre (si Total $_{EF}$ = 3 ó Total $_{EF}$ = 4)

CF = abandonar o cambiar proyecto (si Total $_{EF}$ ≥ 5)

Total $_{EF}$ = Cant EF < 3 (entre E1, E6) + Cant EF > 3 (entre E7, E8)

Como

Total EF = 1 + 0

Total EF = 1

CF = 20 horas-hombre (porque Total $_{EF}$ **≤** 2**)**

Finalmente

E = UCP x CF

E = 121.2519 x 20

E = 2425,038 Horas/Hombre

Para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software. Para ello se puede tener en cuenta el siguiente criterio, que estadísticamente se considera aceptable. El criterio plantea la distribución del esfuerzo entre las diferentes actividades de un proyecto, según la siguiente aproximación:

Actividad	% esfuerzo	Valor esfuerzo
Análisis	10%	606.260 horas-hombre
Diseño	20%	1212.519 horas-hombre
Implementación	40%	2425.038 horas-hombre
Prueba	15%	909.389 horas-hombre
Sobrecarga	15%	909.389 horas-hombre
Total	100%	6062.595 horas-hombre

Tabla 9. Distribución del esfuerzo.

Si **ET = 6062.595 horas-hombre** y se estima que cada mes tiene 4 semanas y cada semana 6 días laborales, por tanto se trabajaría 24 días al mes y si se trabaja 6 horas diarias como promedio se trabajarían en un mes 144 horas laborables, eso daría un **ET = 42.1 mes-hombre**.

Si: Tiempo = **ET / CH**

Tiempo = 42,10 / 10

Tiempo = **4.21**

Esto quiere decir que con 10 hombres trabajando en el modulo del proyecto el mismo se desarrolla en aproximadamente **4 meses y una semana**.

4.4 Factibilidad del diseño

4.4.1 Métricas de diseño arquitectónico

En este epígrafe se realizan los cálculos para el Módulo Entrada de Datos, aplicando las métricas para el diseño arquitectónico. Haciendo referencia a las fórmulas definidas en el capítulo 1.

Algunos actores han propuesto umbrales para estas métricas como se explica a continuación:
Para caracterizar un módulo en no complejo estructuralmente, los valores de los umbrales, según la fórmula definida en el capítulo 1 deben ser de: 1; 4; 9; 16; 25, los cuales corresponde con valores de 1; 2; 3; 4; 5 para $f_{out}(i)$, donde $f_{out}(i)$ representa la cantidad de relaciones de este módulo con los de más módulos. Así mismo se definen valores para un módulo complejo de 36; 49; 64 y muy complejo mayor igual que 81.Para caracterizar un módulo de "baja" complejidad de datos $D(i)$, debe ser menor igual que 7, para la categoría de "complejo" mayor que 7 y menor igual que 12 y para "muy complejo" mayor que 12.

En el caso de la complejidad de sistema $C(i)$, que representa la suma de la complejidad estructural más la de datos, se define que los valores para un sistema no complejo deben estar en el rango de menor igual que 32. Complejo de mayor que 32 y menor igual que 50 y muy complejo los valores deben ser mayor que 50.

A continuación se muestra una tabla con los valores definidos anteriormente:

Complejidad del Sistema			
Complejidad	**S(i)**	**D(i)**	**C(i)**
No complejo	1, 4. 9, 16 ,25	<=7	<=32
Complejo	36, 49, 64	>7 y <=12	>32 y <=50
Muy complejo	81…	>12	>50

Tabla 10. Umbrales de complejidad.

Los indicadores de calidad que se pueden medir con estas métricas corresponden a la integración entre los módulos y la complejidad de las pruebas. En la siguiente tabla se muestran estos indicadores o parámetros de calidad:

Parámetros de Calidad	Sistemas complejos y muy complejos
Integración	Aumenta el esfuerzo necesario para la integración.
Complejidad de las pruebas	Aumenta la complejidad de las pruebas.

Tabla 11. Parámetros de calidad para sistemas altamente complejos.

A continuación se definen las relaciones con el Módulo entrada de Datos

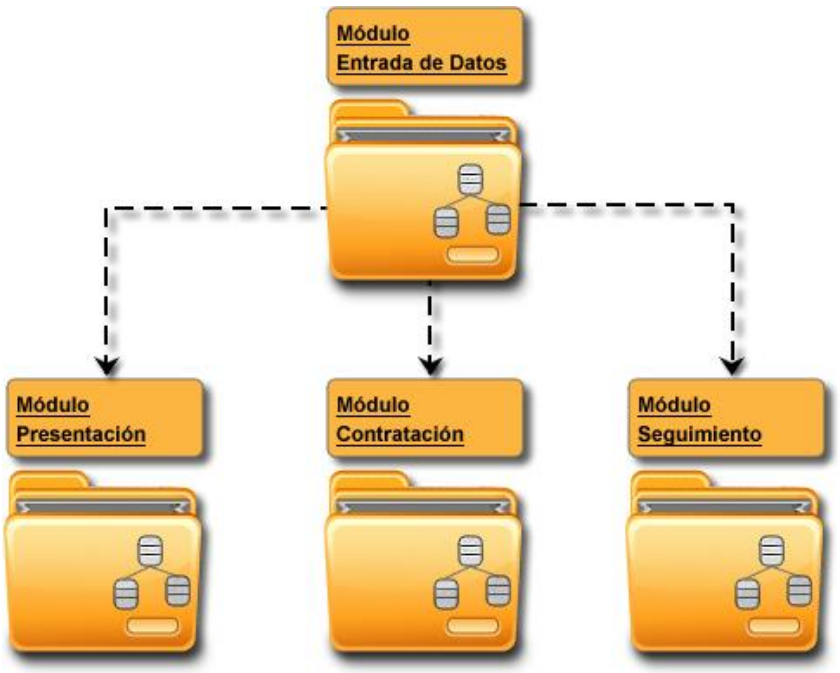


Figura 15. Módulo Entrada de Datos y los módulos descendientes.

$S(i)$ = 3^2 = 8
$D(i)$ = 87 / [3+1] = 21.25
$C(i)$ = 21.25 + 3 = 24.25

Según los valores obtenidos en el análisis realizado y partiendo de los umbrales definidos con anterioridad, se obtuvieron los resultados ilustrados en la siguiente gráfica:

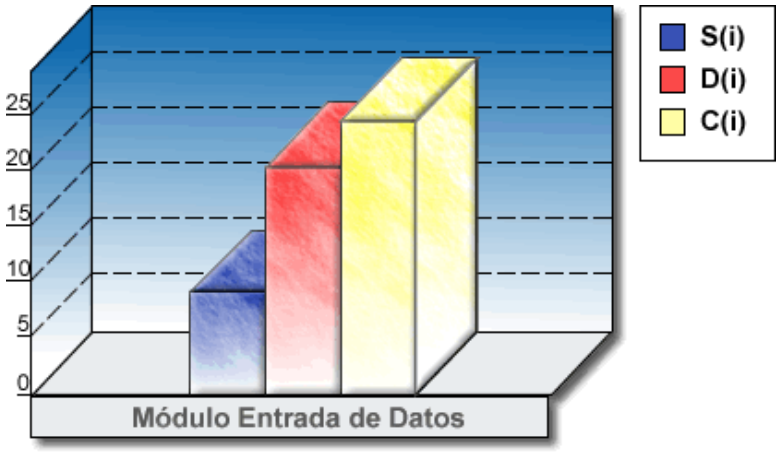


Figura 16. Niveles de Complejidad

Resultado

Como se observa en la gráfica y con el apoyo de la Tabla 10, los niveles de complejidad del módulo "Entrada de Datos" según los valores de S(i), D(i), C(i) se definen de la siguiente manera:

Tipo de Complejidad	Clasificación
Estructural	No complejo
De Datos	Muy complejo
Del Módulo	Poco complejo

Tabla 12. Niveles de complejidad.

4.4.2 Resultado de las métricas orientadas a clases

Métricas propuestas por Lorenz y Kidd. Aplicación al modelo.

A continuación se aplican algunas de las métricas antes mencionadas, para determinar el grado de calidad y fiabilidad del diseño propuesto en este trabajo.

Tamaño de clase (TC)

Esta métrica consiste en medir el tamaño de una clase a partir de las siguientes medidas:

1. Total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
2. Número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.
3. Promedio general de las dos métricas anteriores para el sistema en general.

La métrica TC fue aplicada solamente a la capa de negocio porque es la más compleja dentro de las capas y la que engloba la mayoría de la lógica del negocio. Un TC grande afecta los indicadores de calidad definidos para esta métrica por los especialistas, ver la Tabla 13.

Parámetros de calidad	A valores grande de TC
Reutilización	Reduce la reutilización de la clase
Implementación	Complica la implementación
Complejidad de las pruebas	Hace compleja las pruebas del sistema
Responsabilidad	La clase debe tener bastante responsabilidad

Tabla 13. Parámetros de calidad para valores grandes de TC.

Las medidas o umbrales para los parámetros de calidad han sido una polémica a nivel mundial en el diseño de sistemas.

Los umbrales que plantean algunos especialistas para estas métricas se muestra en la Tabla 14, estos fueron los aplicados en el diseño para el Módulo Entada de Datos.

No de Operaciones y/o Atributos	
TC	**Umbral**
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 14. Umbrales para TC.

Resultado

Los resultados para esta métrica en el diseño del módulo fueron los siguientes:

Total de Clases	Promedio de Atributos	Promedio de Operaciones
33	2.78	9.09

Tabla 15. Total de clases del negocio y los promedios de atributos y operaciones.

La capa de negocio presenta 33 clases pequeñas

Umbral	Tamaño	Cantidad de Clases
<=20	Pequeño	33

Tabla 16. Cantidad de clase por tamaño.

Como se puede observar en la tabla todas las clases están clasificadas en pequeñas; para un resultado positivo según los parámetros de calidad propuesto para la métrica TC.

4.4.3 La serie de métricas CK. Aplicación al modelo.

Dentro de las métricas OO (Orientadas a Objetos) considerablemente referenciadas, se conocen las propuestas por Chidamber y Kemerer. Normalmente nombradas como la serie de métricas CK, los autores han propuesto un conjunto de métricas basadas en clases para este tipo de sistema.
Se seleccionó una de estas métricas para ser aplicada en el siguiente epígrafe.

4.4.3.1 Árbol de profundidad de herencia (APH)

Esta métrica se define como <<la máxima longitud del nodo a la raíz del árbol >>. A medida que el APH crece, es posible que clases de más bajos niveles hereden muchos métodos. Esto conlleva dificultades potenciales, cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda (el APH es largo) también conduce a una complejidad de diseño mayor. Por el lado positivo, los valores APH grandes implican un gran número de métodos que se reutilizarán. (Pressman, 2002)
Por su parte, algunos autores sugieren un umbral de 6 niveles como indicador de un abuso en la herencia en distintos lenguajes de programación.

Resultado

De acuerdo con los datos obtenidos, luego de aplicar al módulo la métrica APH se obtuvo un valor de 2 como el nivel más alto de herencia, lo cual se encuentra dentro del umbral definido para determinar que el diseño no es complejo, no existe un alto acoplamiento y no es de difícil mantenimiento.

4.5 Beneficios tangibles e intangibles

El desarrollo de CICCV es un proyecto de cooperación que aporta en esta primera etapa un considerable beneficio económico al país. No obstante el mayor aporte de la misma se enfoca a la disponibilidad de un eficaz sistema informacional para los gobiernos de Cuba y Venezuela. Siendo así el mayor beneficio tanto para las relaciones entre ambos países como para la Universidad por su considerable presencia dentro del convenio, también por ser un producto basado en la creación de modelos flexibles y robustos que permiten utilizarlo en múltiples entornos empresariales, de manera sencilla y amigable, al poder conocer y gestionar de forma precisa y en el menor tiempo posible la información de interés de los procesos que rigen el funcionamiento de una Corporación o Empresa al llevar a cabo proyectos de forma bilateral.

Por tanto además de los beneficios tangibles que representan la obtención del producto en sí se generan beneficios intangibles como son:

- Disminución del tiempo y el esfuerzo que se invierte en la masificación de las informaciones referentes a los proyectos dentro del los niveles jerárquicos, empresariales y gubernamentales de ambos países que se realizaban de formas disimiles y sin ningún tipo de estándar.
- Descentralización de la generación de información por ministerios y sus respectivos entes ejecutores garantizando que a las instancias superiores llegue la información precisa en el momento adecuado.
- Disponibilidad de flujos de trabajo dentro de la herramienta que abarcan todo el proceso de creación control y seguimiento de un proyecto.
- Fácil detección de problemas y procesamiento de la información.

4.6 Análisis de costos y beneficios

El costo actual para la construcción de un producto informático es elevado. Para poder llevar a cabo el mismo depende en gran medida de los beneficios que reportaría. Estos pueden ser de orden económico y o social. La solución propuesta aporta ambos beneficios con la visión de elevar el nivel de información, control y seguimiento de los proyectos enmarcados en las mixtas, por lo que mejora considerablemente el trabajo en las entidades donde se implante.

El despliegue de esta aplicación mejorará la eficiencia de los servicios y recursos informacionales que se brindan dentro de ambos países, al disminuir el tiempo necesario para la gestión de proyectos. Además sienta las bases para el desarrollo de posteriores versiones y evolución del sistema. La tecnología utilizada para el desarrollo del sistema es totalmente libre, apartando los gastos por conceptos de licencias. El sistema es portable por lo que un cambio de plataforma para la implantación del mismo es viable y factible.

Analizando los numerosos beneficios que reporta tanto económicos como sociales, detallados con anterioridad, se puede concluir que su implementación es realmente factible.

4.7 Conclusiones

En este capítulo se describió el estudio de factibilidad del sistema propuesto mediante la utilización de1 Método de Punto de Casos de Uso. También se aplicaron métricas para la evaluación del diseño. Se determinó que el diseño propuesto no presenta una alta complejidad de sistema en general; permitiendo

que las pruebas no sean complejas y no exista un gran esfuerzo para integrar los módulos. Las métricas como “Tamaño de Clase” evidencian que la mayoría de las clases son reutilizables, y que la implementación no es complicada; se observa además la profundidad de los niveles de herencia y las mismas están acordes con los umbrales definidos por algunos autores permitiendo el bajo acoplamiento. Por lo que se considera que el análisis y el diseño propuesto poseen un alto nivel de factibilidad.

CONCLUSIONES GENERALES

Después de concluir el presente trabajo y haber logrado el cumplimiento del los objetivos propuestos al inicio del mismo, se arribaron a las siguientes conclusiones:

- Se obtuvo una definición sobre conceptos claves para el objeto de estudio.
- Se hizo un análisis de alguno de los productos actuales desarrollados como antecedentes y base para la aplicación.
- Quedó definida la metodología, tecnología y herramientas para desarrollar el análisis y diseño del módulo.
- El modelamiento del negocio, los requerimientos y la especificación de casos de uso del sistema constituyeron el punto de partida básico para la elaboración del trabajo, permitiendo de esta forma ajustar el diseño a los requisitos del sistema y cumplir las expectativas de los usuarios; siguiendo las restricciones estipuladas en las Reglas de Negocio.
- A partir de la fundamentación teórica, luego de un previo estudio sobre los conceptos del análisis y principios del diseño, se estableció la estrategia para lograr un diseño flexible y escalable, aplicando el uso de patrones.
- Durante diseño se definió la Arquitectura Lógica del Sistema, tomando como base los patrones aplicados y la propuesta de solución mediante frameworks para facilitar la implementación y funcionalidades de las principales capas definidas.
- Se obtuvieron los artefactos necesarios, según la metodología de desarrollo de software seleccionada a partir del estudio del arte realizado (RUP), siguiendo el orden de las fases que propone y la continuidad de cada uno mediante los ciclos de desarrollo para implementar la propuesta de solución.
- Los resultados obtenidos a partir del análisis y diseño del sistema son positivos, tomando como referencia la aplicación de métodos y métricas para validar la solución del mismo.

RECOMENDACIONES

A continuación, algunas sugerencias o recomendaciones que deben tenerse en cuenta para trabajos futuros:

Se recomienda:

- El desarrollo iterativo del trabajo para que sea implementado con éxito.
- Analizar los diferentes procesos que intervienen en los restantes módulos para realizar su posterior integración con el módulo Entrada de Datos.
- Desplegar el sistema tanto en Cuba como en otros países en los que también se establece la creación de proyectos, control y seguimiento de los mismos.

BIBLIOGRAFÍA

HUIDOBRO, E. G.-M. (Diciembre de 2001). *El Profesional de la Información.* Recuperado el 8 de Enero de 2008, de El Profesional de la Información: http://www.elprofesionaldelainformacion.com

PROJECT MANAGEMENT INSTITUTE. (2000). *A Guide to the Project Management Body of Knowledge* (primer edición 2000 ed.). Project Management Institute.
Project Management Institute. (2000). *PMBOK 2000.*

EISNER, H. (2000). *Ingeniería De Sistemas Y Gestión De Proyectos.* Madrid: AENOR 2000.

GONZALEZ, M. G. (2006). *Modelo de indicadorez de Calidad en el Ciclo de Vida de Proyectos.* Barcelona, Eespaña: Universidad Politécnica de Cataluña.

GESTION DE PROYECTOS. (s.f.). Recuperado el 3 de Diciembre de 2007, de Gestión de Proyectos: http://www.getec.etsit.upm.es/docencia/gproyectos/gproyectos.htm

KMKEY. (s.f.). *kmkey knowledge manager*. Recuperado el 5 de Diciembre de 2007, de http://www.kmkey.com: http://www.kmkey.com/productos/kmkey_project

B-KIN. (s.f.). *gestion de proyectos*. Recuperado el 5 de Diciembre de 2007, de www.b-kin.com: http://www.b-kin.com/ES/gestiondeproyectos

RATIONAL. (s.f.). Recuperado el 6 de diciembre de 2007, de www.rational.com: www.rational.com

VISUAL PARADIGM. (s.f.). Recuperado el 6 de Diciembre de 2007, de www.visual-paradigm.com: http://www.visual-paradigm.com/

SUN MICROSISTEMS. (s.f.). Recuperado el 10 de Enero de 2008, de http://java.sun.com

MICROSOFT. (s.f.). Recuperado el 10 de Enero de 2008, de http://msdn2.microsoft.com

GRADY, B. (1996). *Análisis y Diseño Orientado a Objetos con Aplicaciones.* Pearson Prentice Hall .

JBOSS. (s.f.). Recuperado el 9 de Febrero de 2008, de Hibernate: http://www.hibernate.org

MIGUEL, A. d. (1997). *Fundamentos y modelos de bases de datos.* Madrid: RA-MA .

CEIS. (s.f.). *Centro Experimental de Ingenieria de Software*. Recuperado el 13 de Diciembre de 2007, de http://www.ceis.cl/Gestacion/Gestacion

MANUALES DE AYUDA. (s.f.). Recuperado el 12 de Diciembre de 2007, de http://www.manualesdeayuda.com/manuales/bases-de-datos/postgresql

IVAR JACOBSON, G. B., JAMES RUMBAUGH (2000).*"El proceso unificado de desarrollo de software."*.

EXTREME PROGRAMMING. (s.f.). Obtenido de "Extreme Programming" (lecture paper): USFCA-edu-601-lecture

MICROSOFT. (s.f.) *Microsoft*. Recuperado el 15 de Diciembre de 2007, de http://www.microsoft.com/msf; http://agilemanifesto.org/

PRESSMAN, R. (2002). *Ingeniería de Software "Un enfoque práctico"* (5ta edición ed.). Madrid: Graw Hill.

BUSINESS RULES GROUP. (1 de Noviembre de 2005). *Business Rules Group.* (R. G. Ross., Ed.) Recuperado el 8 de Enero de 2008, de www.BusinessRulesGroup.org

LARMAN, C. (2003). *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Prentice Hall.

GAMMA, E. (1995). *Design Pattern*. Addison-Wesley

GIRALDO. (2007). *Métricas, Estimación y Planificación en Proyectos de Software.*

PERALTA, M. (2004). *Estimación del Esfuerzo Basada en Casos de Usos. Reportes Técnicos en Ingeniería del Software.*