Universidad de las Ciencias Informáticas

Levantamiento de Requisitos y Diseño de los Módulos Administración y Configuración del Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba – Venezuela.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Janier Acosta Aragón.

Tutores: Ing. Yurisleidys Leiva Zuñiga.

Ing. José Sevila Hidalgo

Ciudad de La Habana, Cuba

2009-2010.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Janier Acosta Aragón	**Yurisleidys Leiva Zuñiga**	José Sevila Hidalgo
_____	_____	_____
Firma del Autor	Firma del Tutor	Firma del Tutor

AGRADECIMIENTOS

A los amigos que han contribuido a la realización de este trabajo y me han dado mucho apoyo y muchas fuerzas para seguir adelante.

A los profesores que formaron parte de mi formación como estudiante y siempre me inculcaron valores positivos siendo este trabajo la culminación de todo un proceso docente.

A los tutores por su abnegada y constante preocupación por este trabajo, por sus experiencias y profesionalismo a la hora de inculcarme los conocimientos.

A mi novia por ser un factor fundamental en el apoyo que necesité para realizar este trabajo y ser la persona que ha estado en los malos y buenos momentos dándome mucho aliento.

A mis padres agradecerles su apoyo, su preocupación y darme esa confianza que me ha servido para vencer y lograr el objetivo de realizar este trabajo.

DEDICATORIA

Dedicarles este sueño alcanzado a mis abuelos que no están presentes, pero están en mi memoria. Aunque estén ausentes, aquí está el resultado de lo que siempre añoraron que fuera, un profesional.

A mis padres por ser las personas que han influido de forma directa en mi formación académica, por servirme de inspiración desde pequeño para lograr este sueño y darme todo el apoyo necesario para seguir los pasos de ustedes. A mi hermana, mi novia y el resto de mi familia les agradezco mucho por siempre estar enfocados en mi desarrollo como estudiante, sin el apoyo de todos ustedes nunca había sido posible la realización de este trabajo y poder alcanzar ese fruto que siempre mis padres quisieron que obtuviese. Me siento feliz y agradecido por tener la posibilidad de tener a mi lado personas que me han ayudado a realizar este trabajo.

RESUMEN

A partir de la creación de un sistema informático donde se realizan los procesos de elaboración, concepción, contratación y seguimiento de los proyectos presentados surge la necesidad de crear mecanismos de Administración y Configuración, ya que los procesos antes mencionados por si solos no poseen elementos que le permitan al sistema ser configurable y que este pueda ejecutarse en distintos escenarios

En el presente trabajo se realiza el levantamiento de requisitos y diseño de los Módulos Administración y Configuración con el fin de contribuir a la configuración, la flexibilidad y la escalabilidad del sistema. Con el levantamiento de requisitos y diseño se logra un sistema bien especificado. Para ello se realizó un estudio de las metodologías, lenguaje de modelado y herramientas para el desarrollo de software, las métricas de software, especificación de requisitos y de los patrones de casos de uso y de diseño. También se identificaron y especificaron los requerimientos y casos de uso del sistema. Además se generaron los artefactos correspondientes al modelo de diseño: subsistemas, paquetes, modelo de datos, diagramas de clases y diagramas de interacción exactamente los diagramas de secuencia. Finalmente se validan los artefactos obtenidos, tanto de la especificación de los requisitos como del modelo de diseño en función de la calidad.

Palabras claves:

Administración, configuración, requisitos, diseño, artefactos.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La República de Cuba y la República Bolivariana de Venezuela mantienen desde hace algunos años estrechos lazos de amistad entre sus pueblos y excelentes relaciones entre sus gobiernos. En el marco de la Alternativa Bolivariana para las Américas (ALBA), ambas naciones, conscientes de su interés común por promover y fomentar el progreso de sus respectivas economías, y las ventajas recíprocas que resultan de una cooperación que tenga resultados efectivos en el avance económico y social de ambos países y la integración de América Latina y el Caribe, se comprometieron a elaborar de común acuerdo, programas y proyectos de cooperación. Para esto existe un Convenio donde ambos países firman acuerdos para desarrollar proyectos conjuntos que beneficien a ambas partes. Los proyectos se firman cada cierto tiempo durante las Comisiones Mixtas. Las Comisiones Mixtas son el mecanismo financiero para los acuerdos a nivel de gobierno, que traen como consecuencia pasar a una fase posterior.

Para la ejecución de estos programas y proyectos de cooperación, se considera la participación de organismos y entidades de los sectores públicos de ambos países. Cuando es necesario, también se incluye la participación de las universidades, organismos de investigación y de organizaciones no gubernamentales.

Existen por la parte cubana y la venezolana toda una estructura administrativa que permite la coordinación para que fluyan los proyectos aprobados en las Comisiones Mixtas. El primer nivel está constituido por los Directivos del Gobierno seguido de los Directivos del Convenio. En un nivel inferior están las Secretarías Técnicas, que se componen de un coordinador de la Secretaría Técnica por cada una de las partes y un grupo de trabajo de esta. Dicho grupo está formado por un jefe de grupo, especialistas que atienden a los diferentes Ministerios u Organismos, un especialista que atiende la parte financiera y uno que atiende las misiones. Subordinados a este nivel de Secretaria Técnica se encuentran los Ministerios correspondientes a cada país, en estos se incluye tanto el ministro como el responsable ejecutor. Así mismo, los Ministerios incluyen a un conjunto de entidades que se desempeñan como ejecutoras de los proyectos, estos entes ejecutores se componen de un director y un responsable que atiende la cooperación en el marco de la Comisión Mixta. Las entidades dirigen a un grupo de equipos que desarrollan cada uno de los proyectos, siendo este el último nivel en la jerarquía.

El Convenio ha crecido y ganado en diversidad. Ha provocado un impacto positivo en importantes sectores del desarrollo económico y social de Cuba y Venezuela.

Actualmente este proceso se realiza de forma manual, lo que hace más engorroso y difícil el trabajo de ambos países en cuanto a la ejecución de los proyectos de cooperación. Además, esto trae como consecuencia que no se tenga un control estricto y centralizado de los proyectos que se han firmado, ni del monto en efectivo relativo a estos. Por todo lo anteriormente expuesto ha sido propuesto iniciar en la Universidad de las Ciencias Informáticas, un proyecto para automatizar las actividades que se efectúan antes, durante y después de realizada la Comisión Mixta de cada año. Para esto, se desarrollará una aplicación web donde puedan acceder las dos partes y de esta forma mejorar el modo en que se manejan los proyectos firmados dentro del marco del Convenio. Esta aplicación web consta de los módulos: presentación, contratación y seguimiento donde se realizan los procesos de elaboración concepción, contratación y seguimiento de proyectos presentados.

La República Bolivariana de Venezuela es un país que está en constante cambio, por lo que los proyectos de colaboración pueden sufrir modificaciones. Los módulos antes mencionados por sí solos no cuentan con un mecanismo que le permita al sistema ser configurable, que esté a expensas de realizar algún cambio y estos cambios puedan estar adaptados al escenario en que se quiere realizar la modificación, por lo que dicho sistema puede ejecutarse en distintos escenarios.

A partir de esta información surge el siguiente problema:

La falta de la definición de los requisitos y el diseño de Módulos de Administración y Configuración en el Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba – Venezuela provoca que no se contribuya a la configuración, la flexibilidad y la escalabilidad del sistema.

Objeto de estudio

Proceso de desarrollo del software.

Campo de acción

Levantamiento de requisitos y diseño de un software

Objetivo:

Realizar el levantamiento de requisitos y el diseño de los Módulos Administración y Configuración del Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba – Venezuela (CCV), de forma que contribuya a la configuración, la flexibilidad y la escalabilidad del sistema.

Objetivos Específicos:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Realizar el levantamiento de requisitos y diseño de los Módulos Administración y Configuración.
- ✓ Validar los requisitos y el diseño propuesto.

Hipótesis:

Si se realiza el levantamiento de requisitos y diseño de los Módulos Administración y Configuración del Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba – Venezuela, entonces se contribuye a la configuración, la flexibilidad y la escalabilidad del sistema.

Métodos y técnicas de investigación a utilizar

Métodos Científicos

Teóricos

- Histórico –Lógico: Permitió realizar el estudio de las diferentes técnicas de análisis y diseño de software, de patrones de caso de uso y de diseño, de las distintas herramientas CASE para el análisis y diseño y métricas de diseño.
- Modelación: Para la creación de modelos y diagramas que reflejen la lógica de los Módulos Administración y Configuración en el levantamiento de requisitos y diseño.
- Hipotético – deductivo: Para la confección de la hipótesis.

Empíricos:

Entrevista: Para interactuar con los clientes e identificar los requerimientos del sistema.

Resultados Esperados

- Diagrama de clases conceptuales del dominio
- Especificación de clases del dominio
- Especificación de los requisitos
- Especificación de los casos de uso
- Diagrama de clases del diseño
- Diagrama de realización de casos de diseño

Estructura del trabajo

El trabajo está compuesto por tres capítulos los cuales se describen a continuación:

Capítulo 1. Fundamentación Teórica.

En el capítulo 1 se tratan aspectos teóricos y técnicos vinculados a la ingeniería de requisitos y diseño de software, así como términos vinculados a patrones de casos de uso, patrones de diseño y métricas de software. Otros aspectos que se tratan son sobre algunas características de las herramientas y metodologías más importantes para realizar el modelado. Finalmente se pone de manifiesto cuales son las técnicas, herramientas y metodologías utilizadas en el desarrollo del sistema.

Capítulo 2. Solución propuesta. Levantamiento de Requisitos y Diseño del sistema.

En el capítulo 2 se presenta la solución de la investigación partiendo del levantamiento de requisitos y diseño de los Módulos Administración y Configuración del sistema CCV. Debido a que se hace difícil por parte de los analistas delimitar los procesos de negocio se determinó que era preciso llevar a cabo un modelo del dominio. Además se analizan los casos de uso, sus especificaciones y se describen los requerimientos funcionales y no funcionales para modelar los artefactos imprescindibles para obtener un modelo de diseño.

Capítulo 3. Análisis de los Resultados.

En el capítulo 3 se hace un análisis de los resultados de la especificación de los requisitos de software. A la hora de lograr una mayor exactitud en la medición del software es necesario poner práctica algunas de las métricas tanto para la especificación de requisitos como para el diseño. Las métricas de diseño son utilizadas para ayudar a que el diseño evolucione a un nivel superior de calidad.

CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se aborda acerca de la evolución del desarrollo de software y las tendencias en la actualidad, resaltando las dificultades a que se enfrenta la industria del software a nivel mundial. Se tratan aspectos teóricos y técnicos vinculados a la ingeniería de requisitos y diseño de software. Otros aspectos que se tratan son los patrones de casos de uso, patrones de diseño y las métricas de software. También se abordan las principales características de algunas herramientas y metodologías para el desarrollo del sistema. Finalmente se pone de manifiesto cuales son las técnicas, herramientas y metodologías seleccionadas para el desarrollo del sistema.

1.2 Ingeniería de software

Hoy en día la industria del software ha logrado una gran hegemonía a escala mundial, ya que al surgir esta nueva aérea de la ingeniería, ha sido un catalizador a la hora de desarrollar un software de calidad que resuelva problemas de todo tipo. Sin embargo, un conjunto de problemas relacionados con el software han persistido a través de la evolución de sistemas basados en computadoras y estos problemas continúan aumentando. Podemos señalar entre las principales dificultades que los avances del software continúan dejando atrás nuestra habilidad de construir software para alcanzar el potencial de hardware y también la habilidad de soportar y mejorar los programas existentes se ve amenazada por diseño pobres y recursos inadecuados. La destreza de construir nuevos programas no pueden ir al mismo ritmo de la demanda de nuevos programas, ni se puede construir programas lo suficientemente rápido como para cumplir las necesidades del mercado y de los negocios. Se puede decir también que el uso extenso de computadoras ha hecho de la sociedad cada vez más dependiente de la operación fiable del software. Cuando el software falla, pueden ocurrir daños económicos enormes y ocasionar sufrimiento humano.

Existen varias personas que han expuesto diferentes definiciones sobre la Ingeniería de Software afirmando que:

- ✓ Ingeniería de Software es el estudio de principios y metodologías para su desarrollo y mantenimiento de sistemas de software. (Zelkovitz, 1976)

- ✓ Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación necesaria requerida para desarrollar, operar (funcionar) y mantenerlos. (Bohem, 1976)
- ✓ Ingeniería del Software trata del establecimiento de los principios y métodos de la Ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales. (Bauer, 1972)

La ingeniería del software trata con áreas muy diversas de la Informática y de las Ciencias de la Computación, tales como construcción de compiladores, sistemas operativos o desarrollos de Intranet/Internet, etc. La evolución del software está condicionada al desarrollo de las tecnologías de la información, viéndose esto como un factor determinante para que pueda lograrse un sistema de software rápido, eficaz y que sobretodo satisfaga la necesidades del cliente.

1.3 Proceso de desarrollo de software

Para construir la ingeniería de software adecuadamente, se debe de definir un proceso de desarrollo de software. (Pressman, 2005). Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Una perspectiva utilizada para determinar los elementos del proceso de desarrollo de software es establecer las relaciones entre elementos que permitan responder **Quién** debe hacer **Qué**, **Cuándo** y **Cómo** debe hacerlo (Letelier, 2003).

Un **proceso de desarrollo de software** es un método de organizar las actividades relacionadas con la creación, presentación y mantenimiento de los sistemas de software. (Larman, 1999)

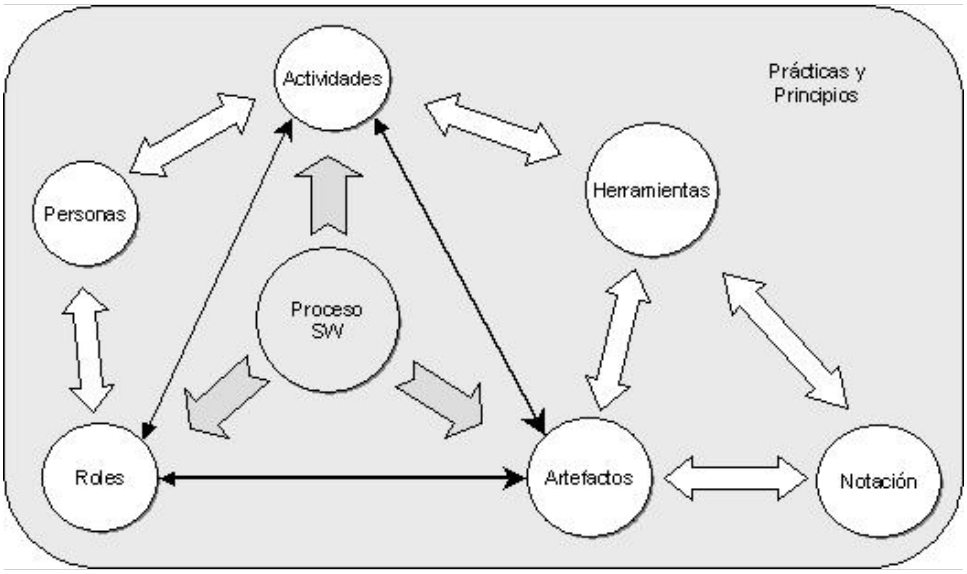

Figura 1 Relación entre los elementos del proceso de software. (Valencia, 2003)

Una persona o un equipo de desarrollo que esté vinculada a un proceso de desarrollo de software se ven afectadas, ya que estas personas son las que están implicadas en el modo como se gestiona y organiza un software. Dentro de las afectaciones podemos describir conceptos como: (Larman, 1999)

- ✓ Viabilidad del proyecto: Las personas que están implicadas a soluciones complejas de sistemas informáticos no disfrutan de ese trabajo.
- ✓ Gestión de riesgos: Cuando el análisis de los riesgos no se ha efectuado las personas se sienten incómodas. Una buena exploración de los riesgos en la primera fase contribuye a la no proliferación de este problema.
- ✓ Estructura de los equipos: El trabajo en equipo se hace más eficaz con número de personas en un rango de seis a ocho no es necesario tener mucho más participantes en un equipo.
- ✓ Planificación de un proyecto: La planificación de un proyecto cuando no está adecuada a la realidad que se quiere demostrar, las personas que están implicadas se sienten desmotivadas.
- ✓ Facilidad de compresión del proyecto: A las personas le gusta saber lo que hace en cada momento, quiere tener una compresión global.
- ✓ Sensación del cumplimiento: Un ritmo de trabajo rápido combinado con conclusiones frecuentes aumenta la sensación del progreso de las personas.

1.3.1 Metodologías de desarrollo de software

Dentro del proceso de desarrollo de software las metodologías de desarrollo de software constituyen una guía de cómo llevar a cabo la producción de un software con el fin de cumplir los requisitos iníciales y minimizar las pérdidas de tiempo. Los integrantes del equipo de desarrollo tienen que estar de acuerdo con las tareas comprendidas en la metodología seleccionada.

Un proceso de software detallado y completo suele denominarse "Metodología". Las metodologías se basan en una combinación de los modelos de procesos genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, entre otras. (Valencia, 2003)

1.3.1.1 Rational Unified Process (RUP)

El proceso unificado es un proceso de desarrollo de software y está basado en componentes, lo cual quiere decir que el sistema de software en construcción está formado por componentes interconectados a través de interfaces. (Jacobson, 2000)

El proceso unificado está caracterizado por tres aspectos fundamentales, los cuales son:

- ✓ **Dirigidos por casos de uso:** Los casos de usos no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Los casos se especifican, se diseñan y los casos de uso finales son la fuente indispensable para realizar la implementación de los mismos.
- ✓ **Centrado en la arquitectura:** Los arquitectos modelan el sistema para darle una arquitectura. Esta arquitectura es la que debe diseñarse para que el sistema evolucione desde su desarrollo inicial hasta las futuras generaciones.
- ✓ **Iterativo e incremental:** Debido al gran esfuerzo que se hace necesario para desarrollar un software es vital dividir el trabajo en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración del cual se obtiene un incremento que produce un crecimiento en el producto.

  El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina.

La eliminación de una de estas tres ideas que caracterizan a RUP reduciría drásticamente el valor del proceso unificado. (Jacobson, 2000)

RUP está definido en función de flujos de trabajos y fases (**figura 2**).

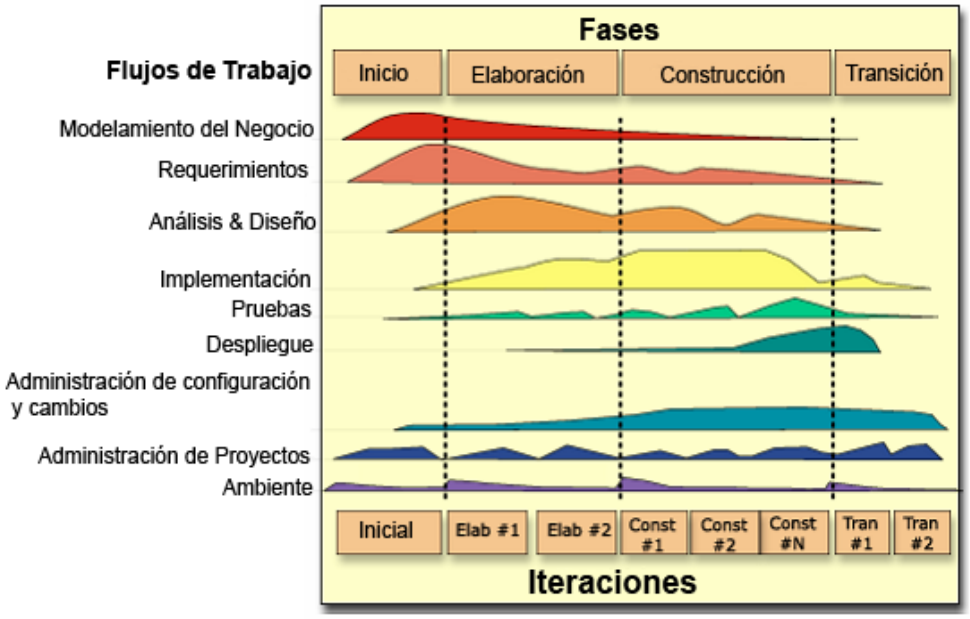

Figura 2 Proceso unificado de desarrollo de software (fases y flujos de trabajos).

Las fases del ciclo de vida de RUP antes mencionada serán descritas a continuación:

- ✓ **Inicialización:** Durante esta fase se desarrolla una descripción del producto final a partir de una buena idea de que es lo que se quiere hacer. Surgen algunas interrogantes las cuales son: ¿Cómo sería la arquitectura candidata del sistema?, ¿Cuál sería el plan de proyecto y el costo de estimación del proyecto?
- ✓ **Elaboración:** Se especifican los casos de usos y se diseña la arquitectura del sistema. Se obtiene también una aplicación ejecutable que responde a los casos de uso que lo comprometen.
- ✓ **Construcción:** Se obtiene un producto listo, creciendo la línea base hasta convertirse en un sistema completo.

- ✓ **Transición:** El producto se convierte en una versión beta. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras surgidas en una versión general. El producto está listo para su instalación en condicionales reales.

Como ilustra la **figura 2** en RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

1.3.1.2 Extreme Programming (XP)

Esta metodología de desarrollo de software forma parte del grupo de metodologías ágiles, las cuales se caracterizan porque el proceso de desarrollo de software sea **incremental**, **cooperativo**, **sencillo** y **adaptable**.

Específicamente esta metodología ágil centrada en fomentar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (Valencia, 2003)

Según (Beck, 1999) los roles de XP son:

- ✓ Programador.
- ✓ Cliente
- ✓ Encargado de pruebas (Tester).
- ✓ Encargado de seguimiento (Tracker).
- ✓ Entrenador (Coach).
- ✓ Consultor.
- ✓ Gestor (Big boss).

El ciclo de desarrollo a grandes rasgos consiste en: (Jeffries, 2001)

- ✓ El cliente define el valor de negocio a implementar.
- ✓ El programador estima el esfuerzo necesario para su implementación.
- ✓ El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

- ✓ El programador construye ese valor de negocio.

El ciclo de vida ideal de XP está compuesto por seis fases: Exploración, Planificación de la Entrega (Reléase), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

1.3.1.3 Microsoft Solution Framework (MSF)

Microsoft Solution Framework (MSF) es un grupo de guías para lograr que una solución en sistemas de información pueda ser finalizada exitosamente, rápidamente y reduciendo la cantidad de personas y riesgos.

El modelo de proceso MSF, propone una secuencia generalizada de actividades para la construcción de soluciones empresariales. Este proceso es flexible y se puede adaptar al diseño y desarrollo de una amplia gama de proyectos de una empresa. (Ávila, 2005)

Las principales características de MSF son:

- ✓ Adaptable.
- ✓ Flexible.
- ✓ Escalable.
- ✓ Agnóstico a tecnologías.

El Modelo de proceso MSF consta de cinco fases distintas:

- ✓ Previsión.
- ✓ Planeamiento.
- ✓ Desarrollo.
- ✓ Estabilización.
- ✓ Implementación.

Este proceso incorpora las prácticas probadas desarrolladas en Microsoft con respecto a los requerimientos, diseño, seguridades, rendimiento y pruebas (testing).

- ✓ Administración de Proyectos. Esta sección contiene archivos en blanco de Project y Excel que muestran la documentación que se necesita para la Administración de Proyectos.
- ✓ Requerimientos. Esta sección contiene documentos de ejemplo y plantillas para definir personas, escenarios y un documento de visión.

- ✓ Seguridad. Esta sección contiene una hoja de Excel para cuestiones relacionadas con seguridades y una plantilla de Word para definir la gestión de riesgos.
- ✓ Pruebas. Esta sección contiene una hoja para definir la orientación de las pruebas y una plantilla en Word para crear más de ellas. (Larrea, 2005)

Se puede afirmar que Microsoft Solutions Framework proporciona las mejores prácticas para planear, diseñar, convertir y desarrollar exitosas soluciones empresariales.

1.3.2 Selección de la metodología de desarrollo de software

Luego de la realización de un estudio de las metodologías más usadas en la actualidad, se puede concluir que RUP es la más indicada para el desarrollo de un sistema de gestión como el que se implementará, ya que esta metodología se encarga de asegurar la producción de un software de alta calidad que reúna las necesidades de los usuarios finales dentro de un plan y un presupuesto predecible, proveer un enfoque disciplinado para asignar tareas y responsabilidades dentro del desarrollo del sistema . Esta metodología constituye uno de los estándares internacionales que más aceptación ha tenido en los últimos años en el desarrollo informático. Además, varias herramientas soportan dicha metodología, permitiendo el trabajo en equipo y con la capacidad de generar código en distintos lenguajes de programación a partir de un diseño UML.

1.4 Ingeniería de Requisitos

El proceso de ingeniería de requisitos es un conjunto organizado de actividades que transforman entradas en salidas. En el proceso de desarrollo de un sistema, el equipo de desarrollo se enfrenta al problema de la identificación de requisitos. La definición de las necesidades del sistema es un proceso complejo, pues en él hay que identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales y de los clientes. (Escalona, 2002)

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. (Association, 2004)

Las etapas de la ingeniería de requisitos son: la identificación de los requisitos, análisis de requisitos y negociación, especificación de los requisitos, modelado del sistema, validación de los requisitos y gestión de requisitos. (Sornerville, 1997)

Identificación de requisitos

La captura de requisitos es la actividad en la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema (A., 2001).Según (Jacobson, 2000) la captura de requisitos es un acto de descubrimiento. Es el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir.

El proceso de captura de requisitos resulta complejo si principalmente el entorno de trabajo es desconocido para el equipo de analistas (Escalona, 2002). Las personas que intervienen en este proceso juegan un papel fundamental, producto a la gran responsabilidad y complejidad que requiere esta actividad. Es importante llevar a cabo una buena captura de requisitos, porque sino el cliente no estará satisfecho con el producto.

Existen un conjunto de técnicas que han sido utilizadas para esta actividad como son:

- ✓ **Entrevistas:** Resulta una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. La estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista). (Escalona, 2002)
- ✓ **JAD (Joint Application Development/Desarrollo conjunto de aplicaciones):** Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes (IBM, 1997). Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación (lo que ves es lo que obtienes).
- ✓ **Brainstorming (Tormenta de ideas):** Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El objetivo principal es lograr un consenso sobre los requisitos. Es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre (Raghavan, 1994). Ayuda a la participación de todos los involucrados y permite pensar en otras ideas.
- ✓ **Concept Mapping:** Son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan

con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar (Pan, 2001). Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste.

- ✓ **Sketches y Storyboards:** Esta técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (sketches). Estos sketches pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (storyboard). (Escalona, 2002)
- ✓ **Casos de Uso:** Aunque inicialmente se desarrollaron como una técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para las captura de requisitos (Pan, 2001). La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades. (Insfrán, 2002)
- ✓ **Cuestionarios y Checklists**: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (Checklist). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista. (Escalona, 2002)

Análisis y Negociación de requisitos

Una vez recopilados los requisitos, el producto obtenido configura la base del análisis de requisitos. Los requisitos se agrupan por categorías y se organizan en subconjuntos, se estudia cada requisito en relación con el resto, se examinan los requisitos en su consistencia, completitud y ambigüedad, y se clasifican en base a las necesidades de los clientes y usuarios. (Pressman, 2005)

Especificación de Requisitos

La especificación de los requisitos del software se produce en la culminación de la tarea del análisis. La Introducción a la especificación de los requisitos establece las metas y objetivos del software, describiéndolo en el contexto del sistema. En muchos casos la especificación de requisitos del software

puede estar acompañada de un prototipo ejecutable (que en algunos casos puede sustituir a la especificación), un prototipo en papel o un manual de usuario preliminar. (Pressman, 2005)

Modelado del sistema

Para especificar correctamente lo que se va a desarrollar se necesita un modelo con toda su información. Con el modelo será relativamente fácil asegurar la eficiencia en el trabajo. Para desarrollar el modelo de sistema, se emplea un esquema del modelado del sistema. El ingeniero de sistemas asigna elementos a cada una de las cinco regiones de tratamiento del esquema: (1) interfaz de usuario. (2) entrada, (3) tratamiento y control del sistema, 4) salida y (5) mantenimiento y autocomprobación. (Pressman, 2005)

Validación de Requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea (Lowe, 1999). Existen algunas técnicas para la validación las cuales son:

- ✓ **Reviews o Walk-throughs:** Está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante. (Escalona, 2002)
- ✓ **Listas de chequeo:** Son frecuentemente usadas en inspecciones o revisiones de artefactos generados en el proceso de producción de software; son listas de aspectos que deben ser completados o verificados.
- ✓ **Prototipos**: Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario (Olsina, 1998). Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.
- ✓ **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo (Durán A., 1999). Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

Gestión de requisitos

La gestión de los requisitos no es más que el proceso de gestión en los cambios de un sistema. Según (Pressman, 2005) la gestión de requisitos es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento. Una vez los requisitos han sido identificados, se desarrollarán un conjunto de matrices para su seguimiento. Entre las posibles matrices de seguimiento citamos las siguientes: (Pressman, 2005)

- ✓ Matriz de seguimiento de características: Muestra los requisitos identificados en relación a las características definidas por el cliente del sistema/producto.
- ✓ Matriz de seguimiento de orígenes*:* Identifica el origen de cada requisito.
- ✓ Matriz de seguimiento de dependencias*.* Indica cómo se relacionan los requisitos entre sí.
- ✓ Matriz de seguimiento de subsistemas*.* Vincula los requisitos a los subsistemas que los manejan.
- ✓ Matriz de seguimiento de interfaces*.* Muestra como los requisitos están vinculados a las interfaces externas o internas del sistema.

1.5 Patrones de casos de uso y patrones de diseño

Los patrones para el desarrollo de software son uno de los últimos avances de la tecnología orientada a objetos, ya que estos se convierten en muy importantes para conseguir la reutilización. Los patrones son una forma literaria para resolver problemas de ingeniería del software, mediante el uso de los patrones es necesario establecer una comunicación exacta entre los desarrolladores para lograr solucionar los problemas en el campo de la Ingeniería de Software.

El objetivo principal de los patrones de casos de uso como los del diseño es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones (Juan, 2002). A continuación se abordan especificaciones de los principales patrones que se deben usar para el desarrollo el levantamiento de requisitos y el diseño de software.

1.5.1 Patrones de casos de uso.

La utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. Un patrón de caso de uso es un diseño probado en un modelo de casos de uso, junto con una descripción del contexto en el cual será usado y las consecuencias que tendrá su aplicación en el modelo (Övergaard G. y., 2004). A continuación se expone una descripción de algunos de ellos. (Övergaard G. y., 2004)

Reglas del Negocio

- ✓ **Definición estática**

  Este patrón es aplicado a todos los casos de uso modelando los servicios que son afectados por las reglas del negocio definidas en la organización. Este patrón no influye en la estructura del modelo de casos de uso. Las reglas son descritas en un documento separado, referenciadas por las descripciones de los casos de usos relevantes. Este patrón es apropiado utilizarlo cuando no hay necesidad de cambiar dinámicamente las reglas del negocio mientras el sistema se esté utilizando.

- ✓ **Modificación dinámica**

  Este modelo del patrón contiene un caso de uso llamado Gestionar regla**,** que se encarga de crear, actualizar y eliminar las reglas del negocio. Este patrón es útil cuando la colección de reglas sea modificada dinámicamente, o sea, estas pueden ser modificadas mientras el sistema este corriendo.

Concordancia

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

- ✓ **Reuso**

  Consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones. De manera que deben existir al menos dos de ellos.

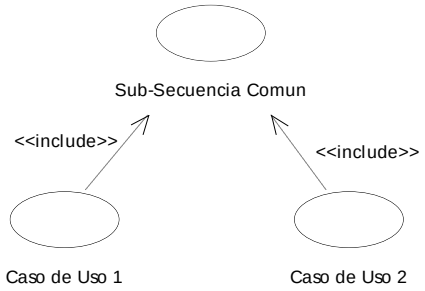

Figura 3 Patrón concordancia, Reuso

- ✓ **Adición**

  En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema.

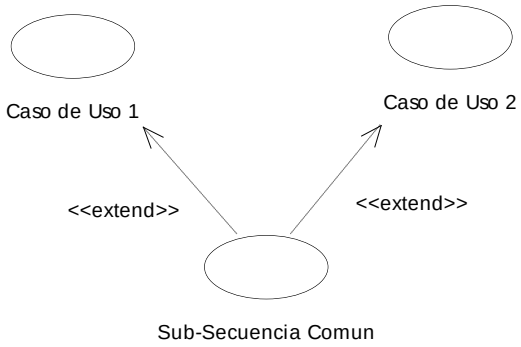

Figura 4 Patrón concordancia, Adición

- ✓ **Especialización**

  Otro patrón de concordancia que contiene casos de uso del mismo tipo. En este caso, estos son modelados como una especialización de casos de uso de tipo de uso común. Todas las acciones en estos casos de uso son heredadas por los casos de uso hijos, donde otras acciones serán adicionadas o acciones heredadas que serán especializadas. Este patrón es aplicable cuando la

utilización de los casos de uso que han sido modelados son del mismo tipo, y este tipo debe hacerse visible en el modelo.

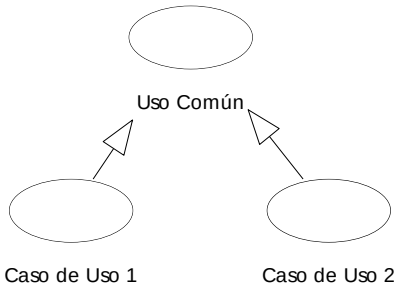

Figura 5 Patrón concordancia, Especialización

- ✓ **Reuso interno**

  Si la subsecuencia de acciones es utilizada en diferentes lugares en un solo caso de uso, no existe la necesidad de extraer la subsecuencia dentro de un caso de uso separado. En cambio, este debe ser descrito en una subsección separada en la descripción del caso de uso. Esta subsección será referenciada desde diferentes partes en la descripción del caso de uso donde las subsecuencia de acciones sean realizadas.

Extensión concreta o inclusión

Se basa en modelar ambos flujos de trabajo como parte de un caso de uso y como separado, completando el caso de uso por sí solo. (Zuñiga, 2008)

- ✓ Extensión: consiste en la existencia de una relación de extensión entre dos casos de uso. El caso de uso extendido puede ser o no instanciado por el caso de uso base. El caso de uso base puede ser concreto o abstracto. Este patrón se utiliza cuando un flujo puede extender el flujo de otro caso de uso o bien puede ejecutarse dentro de este.

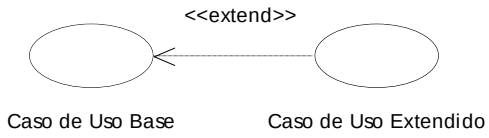

Figura 6 Patrón extensión concreta

- ✓ Inclusión: existe una relación de inclusión del caso de uso base con el caso de uso incluido. Este último puede ser instanciado como el mismo. El caso de uso base puede ser concreto o abstracto. Este patrón se utiliza cuando un flujo puede ser incluido en el flujo de un caso de uso y también puede ejecutarse dentro de este.

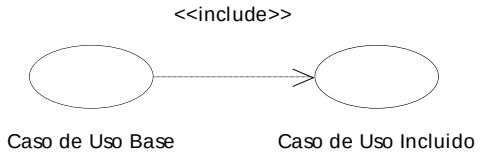

Figura 7 Patrón inclusión concreta.

CRUD (Crear, Leer, Actualizar, Eliminar)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

- ✓ **Completo**

  Consta de un caso de uso, llamado Gestionar información modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

Figura 8 Patrón CRUD completo.

- ✓ **Parcial**

  Modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras.

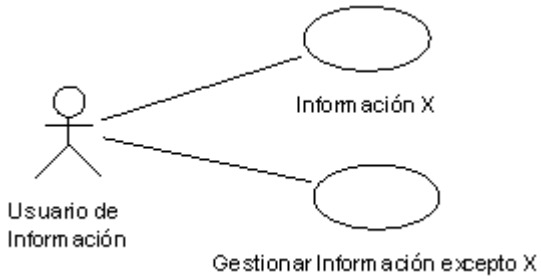

Figura 9 Patrón CRUD parcial

Múltiples actores

- ✓ **Roles diferentes**

  Captura la concordancia entre actores manteniendo roles separados. Consiste de un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso.

Figura 10 Patrón multiples actores , Roles diferentes

- ✓ **Roles comunes**

  Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso.

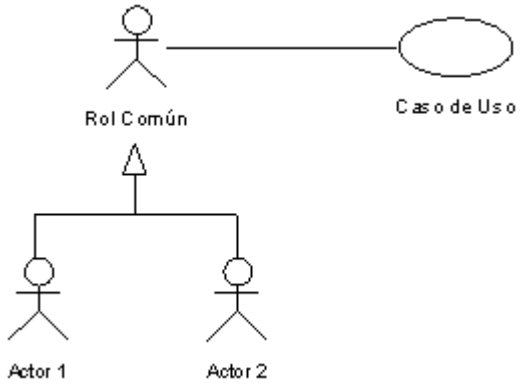

Figura 11 Patrón mutiples actores , Roles comunes

1.5.2 Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño estructurado, común, que lo hace útil para la creación de diseños orientados a objetos reutilizables. Los patrones de diseño identifican las clases participantes y las instancias, sus papeles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se enfoca sobre un particular diseño. (Juan, 2002). Estos patrones solucionan problemas específicos del diseño y hacen los diseños orientados a objetos más flexibles, elegantes y por último reutilizables. Los patrones de diseño ayudan a los diseñadores a reutilizar con éxito diseños para obtener nuevos diseños (Albacete, 2003). Las cualidades de un patrón de diseño son: (Albacete, 2003)

- ✓ **Encapsulación y abstracción**: Cada patrón encapsula un problema bien definido y su solución en un dominio particular.
- ✓ **Extensión** y **variabilidad**. Cada patrón debería ser abierto por extensión o parametrización por otros patrones, de tal forma que pueden aplicarse juntos para solución un gran problema.
- ✓ **Generatividad** y **composición**. Cada patrón una vez aplicado genera un contexto resultante, el que concuerda con el contexto inicial de uno o más de uno de los patrones del catálogo.
- ✓ **Equilibrio**. Cada patrón debe realizar algún tipo de balance entre sus efectos y restricciones.

Los patrones de diseño se clasifican en: (Albacete, 2003)

Patrones de creación: Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones.

- ✓ **Abstract Factory**: Proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta.
- ✓ **Builder**: Permite a un objeto construir un objeto complejo especificando sólo su tipo y contenido.
- ✓ **Factory Method**: Define una interfaz para crear un objeto, dejando a las subclases decidir el tipo específico.

Patrones estructurales: Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros**.**

- ✓ **Adapter**: Convierte la interfaz que ofrece una clase en otra esperada por los clientes.
- ✓ **Bridge**: Desacopla una abstracción de su implementación y les permite variar independientemente.
- ✓ **Composite**: Permite construir objetos complejos mediante composición recursiva de objetos similares.

Patrones de comportamiento: Se utilizan para organizar, manejar y combinar comportamientos.

- ✓ **Chain of Responsability**: Evita el acoplamiento entre quien envía una petición y el receptor de la misma.
- ✓ **Command**: Encapsula una petición de un comando como un objeto.
- ✓ **Interpreter**: Dado un lenguaje, define una representación para su gramática y permite interpretar sus sentencias.

Los patrones GRASP (General Responsibility Asignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Estos patrones se aplican durante la elaboración de los diagramas de interacción. (Larman, 1999)
A continuación se describen los cinco patrones GRASP: (Larman, 1999)

- ✓ **Experto**: Asignar una responsabilidad al experto en información: la clase cuenta con la información necesaria para cumplir la necesidad.
- ✓ **Creador**: Asignarle a la clase B la responsabilidad de crear una instancia de la clase A
- ✓ **Alta cohesión**: Asignar una responsabilidad de modo que la cohesión siga siendo alta.
- ✓ **Bajo Acoplamiento**: Asignar la responsabilidad para mantener bajo acoplamiento.
- ✓ **Controlador**: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.

Los patrones seleccionados para asignar las responsabilidades a los objetos y diseñar la colaboración entre ellos durante la elaboración de los diagramas de interacción fueron el patrón bajo acoplamiento, porque que no existe mucha dependencia entre las clases de los Módulos Administración y Configuración, el diseño de estas es más independiente, lo que reduce el impacto de los cambios y el patrón alta cohesión fue puesto en práctica por el número reducido de operaciones que presenta las clases y con una importante funcionalidad relacionada, lo que soporta un aumento de la capacidad de reutilización.

1.6 Métricas de Software

Un elemento importante de cualquier proceso de ingeniería es la medición. Cuando se planifica un proyecto se tiene que obtener estimaciones del costo y esfuerzo humano requerido por medio de las mediciones de software que se utilizan para recolectar los datos cualitativos acerca del software y sus procesos para aumentar su calidad. Las métricas de software permiten al ingeniero descubrir y corregir problemas potenciales antes de que se conviertan en defectos catastróficos (Pressman, 2005). En los

siguientes subepígrafes se realiza un estudio de las métricas de la calidad de la especificación de requisitos y de las métricas del modelo de diseño.

1.6.1 Métricas de la calidad de la especificación de requisitos

(Pressman, 2005) Se proponen una lista de características que pueden emplearse para valorar la calidad del modelo de análisis y la correspondiente especificación de requisitos: Especificidad (ausencia de ambigüedad, corrección, compleción, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, traza habilidad, capacidad de modificación, exactitud y capacidad de reutilización. Además apuntan que las especificaciones de alta calidad deben estar almacenadas electrónicamente, ser ejecutables o al menos interpretables, anotadas por importancia y estabilidad relativas, con su versión correspondiente, organizadas, con referencias cruzadas y especificadas al nivel correcto de detalle.

Aunque muchas de las características anteriores pueden ser de naturaleza cuantitativa, Davis sugiere que todas puedan representarse usando una o más métricas. Por ejemplo asumimos que hay requisitos en una especificación, tal como

$$\mathrm{nr} = n_f + nn_f$$

Donde n_f es el número de requisitos funcionales y nn_f es el número de requisitos no funcionales (por ejemplo, rendimiento).

Para determinar la especificidad de los requisitos, Davis sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

$$Q1 = \mathrm{nu_i}/\mathrm{n_r}$$

Donde $\mathrm{nu_i}$ es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. Cuanto más cerca de uno esté el valor de Q_1 menor será la ambigüedad de la especificación. La compleción de los requisitos funcionales puede terminarse calculando la relación:

$$Q2 = \mathrm{n_u}/(\mathrm{n_i} * \mathrm{n_s})$$

Donde n_u es el número de requisitos de función únicos, ni es el número de entradas (estímulos) definidos o implicados por la especificación y n_s es el número de estados especificados. La relación Q_2 mide porcentaje de funciones necesarias que se han especificado para un sistema, sin embargo, no trata los requisitos no funciónales. Para incorporarlos a una métrica global completa, debemos considerar el grado de validación de los requisitos:

$$Q3 = n_c/(n_c * nn_v)$$

Donde n_c es el número de requisitos que se han validados como correctos y nn_vel número de requisitos que no se han validado todavía.

1.6.2 Métricas de diseño

Es inconcebible que cuando se lleve a cabo el diseño de un producto se realice sin definir las medidas del diseño, sin determinar las métricas para varios aspectos de la calidad del diseño y usarlas para guiar la evolución del diseño (Pressman, 2005). A continuación se expondrá las métricas más comunes del modelo del diseño.

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 las medidas de 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 (Pressman, 2005). Dentro de las métricas de diseño a nivel de componentes se encuentran las métricas de cohesión y la de acoplamiento las cuales serán descriptas a continuación.

Métricas de cohesión

(Bieman, 1994) Definen una colección de métricas que proporcionan una indicación de la cohesión de un módulo. Las métricas se definen con cinco conceptos y medidas:

- ✓ Porción de datos. Dicho simplemente, una porción de datos es una marcha atrás a través de un módulo que busca valores de datos que afectan a la localización del módulo en el que empezó la marcha atrás. Debería resaltarse que se pueden definir tanto porciones de programas (que se centran en enunciados y condiciones) como porciones de datos.

- ✓ Símbolos léxicos (tokens) de datos. Las variables definidas para un módulo pueden definirse como señales de datos para el módulo.
- ✓ Señales de unión. El conjunto de señales de datos que se encuentran en uno o más porciones de datos.
- ✓ Señales de súper unión. Las señales de datos comunes a todas las porciones de datos de un módulo.
- ✓ Cohesión. La cohesión relativa de una señal de unión es directamente proporcional al número de porciones de datos que liga (Pressman, 2005)

Bieman desarrolla métricas para cohesiones funcionales fuertes (CFF), cohesiones funcionales débiles (CFD) y pegajosidad (el grado relativo con el que las señales de unión ligan juntas porciones de datos). La cohesión funcional fuerte y la pegajosidad se obtienen cuando la métrica de Bieman toma un valor máximo de 1. Para ilustrar el carácter de estas métricas, considere la métrica para la cohesión funcional fuerte:

$$\mathrm{CFF(i)} = \mathrm{SU(SA(i))/muestra(i))}$$

Donde SU (SA (i)) denota muestra de súper unión (el conjunto de señales de datos que se encuentran en todas las porciones de datos de un modulo i). Como la relación de muestras de súper unión con respecto al número total de muestras en un modulo i aumenta hasta un valor máximo de 1, la cohesión funcional del módulo también aumenta.

Métricas de acoplamiento

El acoplamiento de módulo proporciona una indicación de la "conectividad" de un módulo con otros módulos, datos globales y el entorno exterior. (Dhama, 1995) Se propone una métrica para el acoplamiento del módulo que combina el acoplamiento de flujo de datos y de control: acoplamiento global y acoplamiento de entorno. Las medidas necesarias para calcular el acoplamiento de módulo se definen en términos de cada uno de los tres tipos de acoplamiento apuntados anteriormente. (Pressman, 2005)

Para el acoplamiento de flujo de datos y de control*:*

- ✓ d_i = número de parámetros de datos de entrada
- ✓ c_i = número de parámetros de control de entrada
- ✓ d_0 = número de parámetros de datos de salida

- ✓ c_0 = número de parámetros de control de salida

Para el acoplamiento global:

- ✓ g_d = número de variables globales usadas como datos
- ✓ g_c = número de variables globales usadas como control

Para el acoplamiento de entorno:

- ✓ w = número de módulos llamados (expansión) r = número de módulos que llaman al módulo en cuestión (concentración).

Usando estas medidas, se define un indicador de acoplamiento de módulo, m_c de la siguiente manera:

$$m_c = k/M$$

Donde *k* = 1 es una constante de proporcionalidad.

$$M = d_i + a * c_i + d_0 + b * c_0 + g_d + c * g_c + w + r$$

Donde:

$$a = b = c = 2$$

Cuanto mayor es el valor de m_c, menor es el acoplamiento de módulo.

Métricas orientadas a clases

La clase es la unidad principal de cualquier sistema orientado a objeto. Esto dice, que tanto las medidas y métricas para una clase individual, la jerarquía de clases y las colaboraciones de las clases resultarán de gran utilidad al ingeniero que desee estimar la calidad de un diseño. (Heras, 2009)

Tamaño de clase (TC)

El tamaño general de una clase se puede determinar siguiendo los planteamientos a continuación: (Heras, 2009)

- ✓ El número total de operaciones (tanto operaciones heredadas como operaciones privadas de la instancia) que están encapsuladas dentro de la clase.
- ✓ El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

Si existen valores grandes de TC estos estarán demostrando que una clase puede tener demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación y la

prueba. De forma contraria sucede si los valores TC son de menor valor. Por otra parte es necesaria una evaluación concreta de las métricas mediante los umbrales. Algunos especialistas plantean la clasificación de la siguiente manera:

Tabla 1 Clasificación de las clases

Clasificación	Valores de los umbrales
Pequeño	<=20
Medio	>20<=30
Grande	>30

1.7 UML. Lenguaje de Modelado

UML es un lenguaje de modelado que permite visualizar, especificar y construir artefactos de sistemas de software. Este lenguaje está hecho para el desarrollo de software que utilizan conceptos de programación orientada a objetos.

UML es un lenguaje para construir modelos, no guía al desarrollador en la forma de realizar el análisis y diseño orientado a objetos ni le indica cuál proceso de desarrollo adoptar. (Larman, 1999).

El lenguaje UML se compone de tres elementos básicos, los bloques de construcción, las reglas y algunos mecanismos comunes.
Los bloques de construcción se dividen en tres partes:

- ✓ Relaciones: unen elementos entre sí.
- ✓ Elementos: son las abstracciones de primer nivel.
- ✓ Diagramas: son agrupaciones interesantes de elementos

Existen cuatro tipos de elementos en UML, dependiendo del uso que se haga de ellos: elementos estructurales, elementos de comportamiento, elementos de agrupación y elementos de anotación.
También existen cuatro tipos de relaciones entre elementos de un modelo: dependencia, asociación, generalización y realización.

Los diagramas en UML se pueden clasificar en : diagrama de estructura estática, de comportamiento y de implementación.

Diagrama de estructura estática

- ✓ Diagramas de casos de uso.
- ✓ Diagrama de clases.
- ✓ Diagrama de objetos.

Diagrama de comportamiento

- ✓ Diagrama de interacción.
- ✓ Diagrama de secuencia.
- ✓ Diagrama de colaboración.
- ✓ Diagrama de actividad.
- ✓ Diagrama de estados.

Diagrama de implementación

- ✓ Diagrama de componentes.
- ✓ Diagrama de despliegue.

1.8 Herramientas CASE para el Modelado

Los lenguajes de modelado son la notación en la que se basan las herramientas Computer Aided Software Engineering (CASE) para crear modelos de software. Las herramientas para realizar el modelado capacitan al ingeniero del software para crear modelos del sistema que haya que construir. Al efectuar una comprobación de la consistencia y validez del modelo, las herramientas de modelado ayudan al ingeniero a eliminar errores antes de que se propaguen al diseño, o lo que es peor, a la propia implementación. Hoy en día muchas empresas han extendido la adquisición de herramientas CASE, con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema. A continuación se exponen diferentes características de las herramientas CASE más utilizadas en la actualidad.

1.8.1 Rational Rose

Rational ofrece un Proceso Unificado (RUP) para el desarrollo de los proyectos de software, Rose es la herramienta de Rational para la etapa de análisis y diseño de sistemas, es también una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas que posee

Rational Rose es que utiliza el lenguaje de modelado (UML), lo que permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común.

Esta herramienta posee la capacidad de crear, ver, modificar y manipular los componentes de un modelo en la notación UML. (Mestras, 2004)

Entre las características que posee esta herramienta CASE se encuentran:

- ✓ Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en “Design Patterns: Elements of Reusable Object-Oriented Software”.
- ✓ Capacidad de análisis de calidad de código.
- ✓ Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- ✓ Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- ✓ Integración con otras herramientas de desarrollo de Rational.
- ✓ Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

1.8.2 Visual Paradigm

Es una herramienta CASE que utiliza UML: como lenguaje de modelado. Soporta el ciclo de vida completo de desarrollo de software, ayuda a una rápida construcción de aplicaciones de calidad y a un menor coste. Permite modelar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (International., 2005)

Visual Paradigm se caracteriza por lo siguiente: (Zapata, 2005)

- ✓ Soporta aplicaciones web.
- ✓ Las imágenes y reportes generados no son de muy buena calidad.
- ✓ Generación de código para Java y exportación como HTML.
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones.

Sin importar la IDE de preferencia, Visual Paradigm SDE ofrece los siguientes beneficios:

- ✓ Navegación intuitiva entre código y el modelo.
- ✓ Poderoso generador de documentación y reportes UML PDF/HTML/MS Word.
- ✓ Demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente.
- ✓ Superior entorno de modelado visual.

- ✓ Soporte completo de notaciones UML.
- ✓ Diagramas de diseño automático sofisticado.
- ✓ Análisis de texto y soporte de tarjeta CRC.

1.8.3 Enterprise Architect

Enterprise Architect (EA) es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multiusuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. (Ltd, 2000-2008)

Algunas de las características que posee EA son: (Ltd, 2000-2008)

- ✓ Ayuda a manejar la complejidad.
- ✓ EA proporciona generación de documentos poderosa y herramientas de reporte con un completo editor de plantillas WYSIWYG.
- ✓ Soporta la generación y la ingeniería reversa de código fuente para muchos lenguajes populares, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP.
- ✓ Soporta transformaciones de Modelo avanzado de Arquitectura Conducida (MDA) usando plantillas fáciles de desarrollar y editar.
- ✓ Facilidad de Exportación e Importación XMI.

1.8.4 Selección de la herramienta de modelado

Visual Paradigm for UML 6.0 Enterprise Edition, es la herramienta CASE seleccionada para modelar todos los artefactos que se obtendrán a partir del análisis del negocio y el sistema. Esta herramienta es de gran utilidad para el proyecto, por su nivel de integración con el entorno de desarrollo Eclipse y por la posibilidad de trabajar ambos sobre la plataforma libre GNU/Linux, sistema operativo sobre el cual se desarrolla el proyecto.

1.9 Conclusiones Parciales

En este capítulo se realizó un estudio sobre las diferentes metodologías de desarrollo de software más utilizadas y se seleccionó RUP, ya que es una metodología que se encarga de asegurar la producción de un software de alta calidad teniendo presente las necesidades del cliente. De acuerdo con el estudio que

se llevó a cabo sobre la ingeniería de requisitos se aplicarán las etapas de identificación de requisitos, análisis y negociación de requisitos, modelado del sistema, validación y gestión de requisitos, aplicando cada una de las actividades que estas proponen y seleccionando como técnica para la identificación de requisitos la entrevista por ser esta una forma directa para identificar los requisitos. Con la realización de la entrevista el equipo de trabajo se acerca al problema de una forma natural, precisa y se establece una comunicación más directa con el cliente. Se reflejaron los patrones de caso de uso y algunos de diseño, ya que los de caso de uso brindarán gran ayuda a la identificación y la confección de los diagramas de caso de uso de los Módulos Administración y Configuración y los patrones de diseño que se utilizaron para asignar responsabilidades a los objetos fueron el patrón bajo acoplamiento, porque no existe mucha dependencia entre las clases, el diseño de estas es más independiente, lo que reduce el impacto de los cambios y el patrón alta cohesión fue puesto en práctica por el número reducido de operaciones que presenta las clases y con una importante funcionalidad relacionada, lo que soporta un aumento de la capacidad de reutilización. Después de estudiar el lenguaje de modelado UML, se utilizará este por permitir todo el modelado necesario para levantamiento de requisitos y diseño de los Módulos Administración y Configuración. El hecho de cultivarse sobre las herramientas CASE para el modelado, permite utilizar Visual Paradigm por su nivel de integración con el entorno de desarrollo eclipse y por la posibilidad de trabajar ambos sobre la plataforma libre GNU/Linux.

CAPÍTULO 2: SOLUCIÓN PROPUESTA

2.1 Introducción

En el presente capítulo se presenta la solución de la investigación partiendo del levantamiento de requisitos y diseño de los Módulos Administración y Configuración del sistema CCV. Debido a que se hace difícil por parte de los analistas delimitar los procesos de negocio se determinó que era preciso llevar a cabo un modelo del dominio, especificándose así las clases del dominio y representando el diagrama de clases conceptuales. También en este capítulo 2 se describen los requerimientos funcionales y no funcionales. Además se muestran las especificaciones de los casos de uso y de los artefactos imprescindibles del modelo de diseño como: diagrama de clases del diseño con estereotipos web, diagrama de secuencia, paquetes de diseño, subsistema de diseño y el modelo de datos. Dentro de los diagramas de interacción se utilizó el diagrama de secuencia, ya que el centro de atención principal es encontrar las secuencias de acciones detalladas y ordenadas en el tiempo.

2.2 Modelo de Dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen en los eventos que suceden en el entorno en el que se trabaja el sistema. El objetivo del modelo de dominio es comprender y describir las clases más importantes dentro del contexto del sistema (Jacobson, 2000).

La realización del modelo del dominio ayudó a que se comprendiera la estructura de la organización de los tipos de objetos más importantes que intervienen en el desarrollo del Convenio Cuba-Venezuela y que son de interés para los Módulos Administración y Configuración. Fue necesario resaltar que durante la realización de este modelo entre los clientes y el equipo de desarrollo se llegó a un entendimiento común sobre el funcionamiento del negocio.

Como parte del modelo del dominio se obtuvo el diagrama de clases del dominio y la definición de las conceptuales. El modelado del dominio constituyó una entrada fundamental para el flujo de trabajo levantamiento de requisitos.

2.2.1 Especificación de las clases conceptuales del modelo del dominio

Descripción de la clase: Convenio Cuba Venezuela

Acuerdo de cooperación integral firmado entre la República Cuba y la República Bolivariana de Venezuela, con el objetivo de elaborar de común acuerdo, programas y proyectos que beneficien a ambos países.

Descripción de la clase: Comisión Mixta

Reunión que se realiza cada año con representantes de ambos países, con el fin de servir como mecanismo para el cumplimiento y seguimiento de las acciones de cooperación previstas en el Convenio. Está presidida por la parte cubana por el Ministerio para la Inversión Extranjera y la Colaboración Económica (MINVEC) y por la parte venezolana por el Ministerio del Poder Popular para la Energía y Petróleo (MENPET). Cada una de las partes puede proponer a la otra en cualquier momento, nuevos sectores y proyectos específicos de cooperación para su estudio y aprobación. Así mismo, las partes pueden convocar de común acuerdo y cuando lo consideren necesario, reuniones extraordinarias de la Comisión Mixta.

Descripción de la clase: País

Representa las partes involucradas en el Convenio, en este caso Cuba o Venezuela.

Descripción de la clase: Secretaría Técnica

Encargadas de coordinar en ambas partes, todas las actividades necesarias para el buen desarrollo de cada proyecto. Llevan un control estadístico de la evolución de cada uno de los proyectos, aceptan las propuestas y cambios en la ejecución de estos.

Descripción de la clase: Ministerio

Controlan y aprueban las acciones relacionadas con los proyectos, dirigen las actividades de sus entes ejecutores y por tanto aceptan las propuestas que se presenten.

Descripción de la clase: Ente Ejecutor

Es una empresa o institución, que forma parte de los responsables del proyecto. Son los encargados de coordinar y ejecutar cada una de las acciones relacionadas con los proyectos correspondientes a esa entidad.

Descripción de la clase: Persona

Individuos que realizan las actividades correspondientes a cada uno de los entes, ministerios y secretarías, que están involucrados en el Convenio.

Descripción de la clase: Responsable

Personas que realizan actividades de dirección o se encuentran al frente de cada uno de los entes, ministerios y secretarías, que están involucrados en el Convenio.

Descripción de la clase: Proyecto

Acuerdo que cumple con un conjunto de actividades que siguen un objetivo específico y son ejecutadas entre dos o más empresas o instituciones. Estos proyectos son firmados en el marco del Convenio durante las Comisiones Mixtas y aportan beneficios tanto sociales como económicos a ambos países.

Descripción de la clase: Usuario

El usuario es el individuo que se crea para acceder al sistema.

Descripción de la clase: Rol

Conjunto de expectativas de conducta asociadas a una persona, un patrón de comportamiento que se espera de quién desempeñe cada puesto, con cierta independencia de la persona que sea.

Descripción de la clase: Funcionalidad

Describe la funcionalidad que tendrá el rol del usuario para acceder al sistema.

Descripción de la clase: Nivel

Se corresponde con el nivel que tendrá el usuario en el sistema.

2.2.2 Descripción general del dominio

Entre Cuba y Venezuela existe un convenio de cooperación donde se ejecutan cada año una gran cantidad de proyectos con un importante impacto social, político y económico. Cada cierto tiempo se realizan comisiones mixtas en las que se firman acuerdos para desarrollar de forma conjunta estos programas y proyectos de cooperación, que son ejecutados por organismos y entidades de los sectores públicos de ambos países.

Para toda la coordinación, administración y ejecución de los proyectos aprobados en estas comisiones mixtas, existe un conjunto de personas que son los responsables de todo lo que acontece y que está relacionado con los proyectos y con cada tema que tenga que ver con la colaboración. En cada uno de los países hay una secretaría técnica, que son las encargadas de coordinar todas las actividades necesarias para el buen desarrollo de cada proyecto. Subordinados a ellas, están los ministerios correspondientes a cada país, que a su vez controlan un grupo de entidades que se desempeñan como entes ejecutores de estos proyectos.

A las personas que forman parte de la ejecución de los proyectos dentro de la aplicación web es necesario crearle un usuario, el cual este tendrá un nivel y un conjunto de roles que tendrá determinadas funcionalidades para poder acceder a alguna parte del sistema, definiendo así los niveles de seguridad para cada usuario.

Como parte del modelo de dominio, el diagrama de clases conceptuales quedó estructurado como se muestra a continuación:

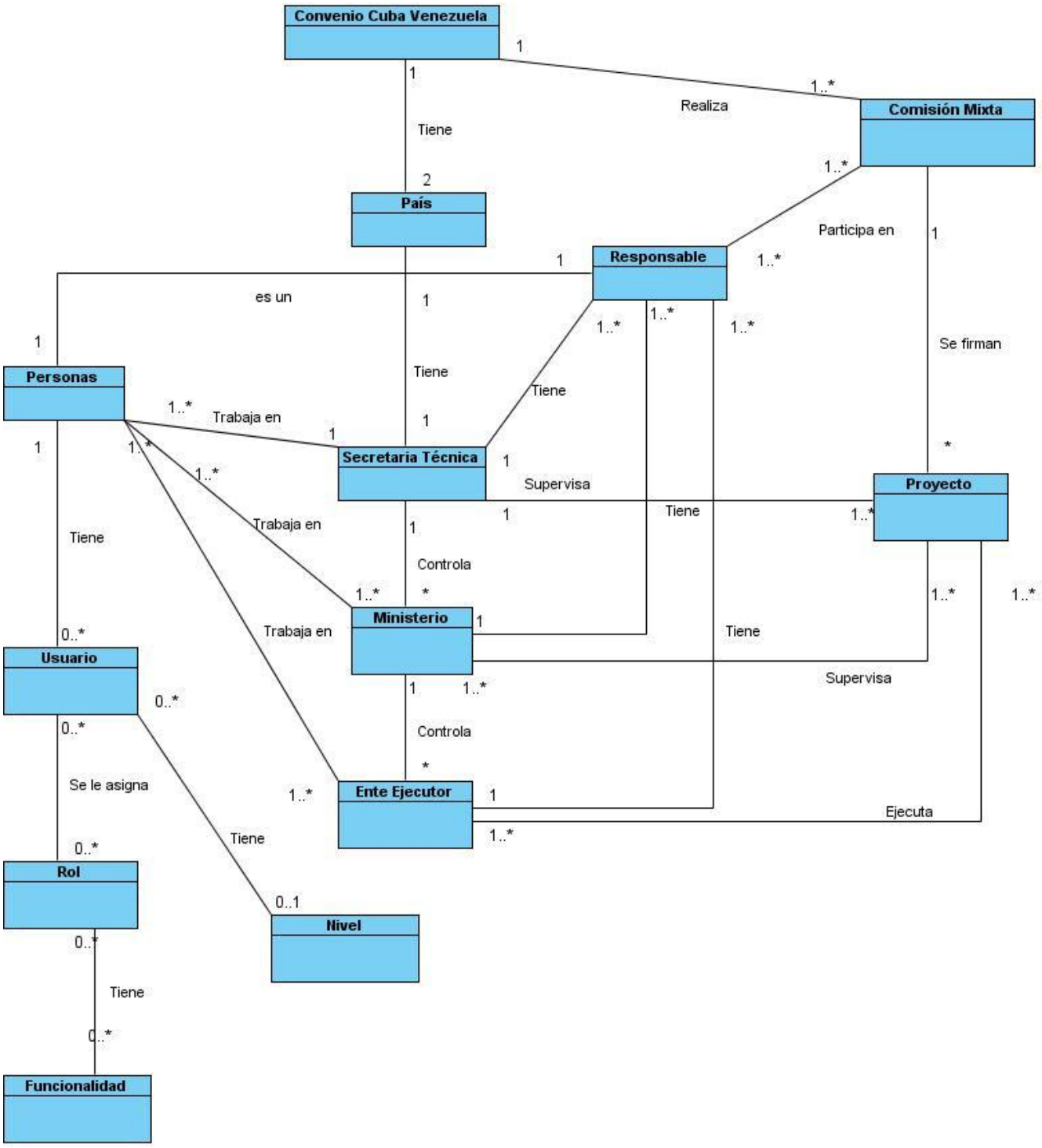

Figura 12 Diagrama de clases conceptuales del dominio

2.3 Requisitos de Software

La obtención de los requisitos de software es resultado del trabajo de captura de requisitos. El mismo recoge los requisitos que debe cumplir la aplicación a desarrollar, que han sido identificados a partir de las necesidades reales de los usuarios y de las demandas del cliente. Los requisitos se agruparon en dos categorías, funcionales y no funcionales, además de llevar un proceso de control de la calidad de la especificación de los mismos. La especificación de los requisitos sirve de guía para llegar a la implementación.

2.3.1 Requisitos Funcionales

Se obtuvieron del Módulo Administración las siguientes condiciones o capacidades que el sistema debe cumplir:

RF01.01 Gestionar Rol

El sistema permitirá gestionar los roles de los usuarios, para ello se deben cumplir con los siguientes requisitos.

RF01.01.01 Crear Rol

El sistema permitirá que se puedan crear nuevos roles, los cuales serán asignados a los usuarios que se creen. Los datos que conforman un rol son los siguientes:

- ✓ Nombre del Rol.
- ✓ Niveles (Se define que niveles debe tener el usuario al cual se le asignará el rol en cuestión).
- ✓ Funcionalidades (Los roles van a estar determinados por las funcionalidades definidas en el sistema que le sean asignadas, lo cual permitirá el acceso a la aplicación)

RF01.01.02 Modificar Rol

- ✓ El sistema permitirá que se pueda modificar un rol creado. Los datos modificables son**:**
- ✓ Nombre del Rol.
- ✓ Niveles.
- ✓ Funcionalidades.

RF01.01.03 Eliminar Rol

El sistema permitirá que se pueda eliminar un rol creado.

RF0101.04 Listar Rol

El sistema permitirá que se puedan listar los roles que estén creados. Los datos que se muestran son:

- ✓ Nombre del Rol.
- ✓ Nivel(es).

RF0101.05 Listar Funcionalidades

El sistema permitirá que se listen las funcionalidades definidas para que sean asignadas a un rol específico que se esté creando.

RF01.01.06 Asignar Funcionalidades a un Rol

El sistema permitirá que se le puedan asignar funcionalidades definidas a un rol.

RF01.02 Gestionar Usuario

El sistema permitirá que se puedan gestionar los usuarios, para ello se deben cumplir los siguientes requisitos:

RF01.02.01 Crear Usuario

El sistema permitirá que se pueda crear un nuevo usuario a una persona determinada, podrán ser creados para los EE, M, ST y Embajada. A continuación se listan los datos de un nuevo usuario.

- ✓ Nombre de Usuario.
- ✓ Nivel (Nivel al que pertenecerá el usuario).
- ✓ Ministerio (Este dato se introduce si se selecciona el nivel Ministerio).
- ✓ EE (Este dato se introduce si se selecciona el nivel Ente Ejecutor).
- ✓ Activo (Este dato es para poner al usuario activo o no en la aplicación).
- ✓ Listado de Roles (Los roles se muestran según el nivel que se haya seleccionado para el usuario).
- ✓ Roles Seleccionados (Roles que serán asignados al usuario que se esté creando).

RF01.02.02 Modificar Usuario

El sistema permitirá que se pueda modificar un usuario existente. Los datos modificables son:

- ✓ Nombre de Usuario.
- ✓ Nivel (Nivel al que pertenecerá el usuario).
- ✓ Ministerio (Este dato se entra si se selecciona el nivel Ministerio).
- ✓ EE (Este dato se entra si se selecciona el nivel Ente Ejecutor).
- ✓ Activo (Este dato es para poner al usuario activo o no en la aplicación).
- ✓ Listado de Roles (Los roles se muestran según el nivel que se haya seleccionado para el usuario).
- ✓ Contraseña.

RF01.02.03 Eliminar Usuario

El sistema permitirá que se pueda eliminar un usuario existente.

RF01.02.04 Listar Usuario

El sistema permitirá listar los usuarios según el resultado de la búsqueda.

RF01.02.05 Buscar Usuario

El sistema permitirá buscar los usuarios que se encuentran creados a partir de un determinado criterio de búsqueda. A continuación se listan los datos de entrada del filtro:

- ✓ Nombre de persona.
- ✓ Primer apellido.
- ✓ Segundo apellido.
- ✓ Nombre usuario.
- ✓ Nivel (Se mostrarán los niveles según el administrador que esté autenticado. Si es un Administrador ST se muestra ST, M, EE, Embajada. Si es un Administrador M se muestra M y EE. Si es un EE se muestra el nivel EE).
- ✓ Ministerio.
- ✓ Ente Ejecutor

RF01.02.06 Generar contraseña

El sistema generará la contraseña automáticamente cuando un usuario sea creado.

RF01.02.07 Enviar Contraseña por Correo Electrónico

El sistema permitirá enviar la contraseña generada cuando el usuario haya sido creado o se le haya restablecido la contraseña.

RF01.02.08 Buscar Persona

El sistema permitirá buscar una persona determinada para crearle el usuario correspondiente o los usuarios a través de un filtro de búsqueda.

RF01.03 Modificar Contraseña

El sistema permitirá que un usuario autenticado pueda cambiar su contraseña.

RF01.04 Autenticar Usuario

El sistema permitirá que un usuario pueda autenticarse suministrando un usuario y contraseña válidos, creados previamente.

Se obtuvieron del Módulo Configuración las siguientes condiciones o capacidades que el sistema debe cumplir:

RF02.01 Crear tipo de Recurso

El sistema permitirá que se puedan crear nuevos tipos de recursos, los cuales serán utilizados para definir los recursos del plan operativo de los proyectos a partir de los datos que se listan a continuación.

- ✓ Tipo de recurso.
- ✓ Nombre del tipo de recurso.
- ✓ Descripción del tipo de recurso

RF02.02 Modificar tipo de Recurso

El sistema permitirá que se pueda modificar los tipos de recursos definidos. Los datos modificables son:

- ✓ Nombre del tipo de recurso.
- ✓ Descripción del tipo de recurso.

RF02.03 Crear tipo de recursos humanos

El sistema permitirá que se puedan crear nuevos tipos de recursos humanos, los cuales se utilizarán para definir los recursos que estarán asociados a los proyectos. A continuación se especifican los datos de entradas:

- ✓ Tipo de recurso humano.
- ✓ Nombre del tipo de recurso humano.
- ✓ Descripción del tipo de recurso humano.

RF02.04 Modificar tipo de recursos humanos

El sistema permitirá que se pueda modificar los tipos de recursos humanos definidos. Los datos modificables son:

- ✓ Nombre del tipo de recurso humano.
- ✓ Descripción del tipo de recurso humano.

RF02.05 Crear fuentes de financiamientos

El sistema permitirá que se puedan crear nuevas fuentes de financiamientos. A continuación se especifican los datos de entradas:

- ✓ Fuente de financiamiento.
- ✓ Nombre del tipo de fuente de financiamiento.
- ✓ Descripción del tipo de fuente de financiamiento.

RF02.06 Modificar fuentes de financiamientos

- ✓ El sistema permitirá que se pueda modificar las fuentes de financiamientos. Los datos modificables son:
- ✓ Nombre del tipo de fuente de financiamiento.

- ✓ Descripción del tipo de fuente de financiamiento.

RF02.07 Crear modalidad de proyectos

El sistema permitirá que se pueda crear modalidades para clasificar los proyectos en la definición del mismo. A continuación se especifican los datos de entrada.

- ✓ Modalidad.
- ✓ Nombre de la modalidad.
- ✓ Descripción de la modalidad.

RF02.08 Modificar modalidad de proyectos

- ✓ El sistema permitirá modificar las modalidades de proyectos definidas. Los datos modificables son:
- ✓ Nombre de la modalidad.
- ✓ Descripción de la modalidad.

RF02.09 Configurar tasa de cambio

El sistema permitirá que se configure la tasa de cambio del bolívar con respecto al dólar.

RF02.10 Configurar honorarios

El sistema permitirá que se configure el intervalo de honorarios para cada tipo de recurso humano.

RF02.11 Gestionar persona

El sistema permitirá que se puedan gestionar los datos de las personas, para ello es necesario cumplir con los siguientes requisitos:

RF02.11.01 Crear Persona

- ✓ El sistema permitirá crear nuevas personas, los datos que conforman una persona son los siguientes:
- ✓ Nombre.

- ✓ Primer Apellido.
- ✓ Segundo Apellido.
- ✓ Cédula.
- ✓ Correo electrónico.
- ✓ Pasaporte.
- ✓ Cargo.
- ✓ Teléfono.

RF02.11.02 Modificar persona

- ✓ El sistema permitirá modificar los datos existentes de una persona. Se podrán modificar los siguientes campos:
- ✓ Nombre.
- ✓ Primer Apellido.
- ✓ Segundo Apellido.
- ✓ Cédula.
- ✓ Correo electrónico.
- ✓ Cargo.
- ✓ Pasaporte.
- ✓ Teléfono.

RF02.11.03 Eliminar persona

El sistema permitirá eliminar personas existentes mientras no tenga usuarios creados

RF02.11.04 Buscar persona

El sistema permitirá buscar personas a través de un filtro de búsqueda.

RF02.11.05 Listar persona

El sistema permitirá mostrar un listado de las personas encontradas a partir de la búsqueda.

RF02.12 Gestionar datos de ministerios

El sistema permitirá que se puedan gestionar los datos de los ministerios, para ello es necesario cumplir con los siguientes requisitos:

RF02.12.01 Crear ministerio

El sistema permitirá crear nuevos Ministerios. Los datos que conforman los Ministerios son los siguientes:

- ✓ Nombre.
- ✓ Siglas.
- ✓ Coordinador.
- ✓ Teléfono.
- ✓ Fax.
- ✓ Descripción.

RF02.12.02 Modificar ministerio

- ✓ El sistema permitirá modificar ministerios y se van a poder modificar todos los datos del mismo:
- ✓ Nombre.
- ✓ Siglas.
- ✓ Coordinador.
- ✓ Teléfono.
- ✓ Fax.
- ✓ Descripción.

RF02.12.03 Eliminar ministerio

El sistema permitirá eliminar ministerios mientras no tenga usuarios asignados o entes ejecutores.

RF02.12.04 Listar ministerio

El sistema permitirá mostrar un listado de los ministerios de cualquiera de las dos partes, puede ser venezolana o cubana.

RF02.13 Gestionar datos de entes ejecutores

El sistema permitirá que se puedan gestionar los datos de los entes ejecutores, para ello es necesario cumplir con los siguientes requisitos:

RF02.13.01 Crear entes ejecutores

El sistema permitirá crear nuevos entes ejecutores, los datos que conforman los entes ejecutores son los siguientes:

- ✓ Nombre.
- ✓ Siglas.
- ✓ Ministerio.
- ✓ Coordinador.
- ✓ Teléfono.
- ✓ Fax.
- ✓ Descripción.

RF02.13.02 Modificar entes ejecutores

El sistema permitirá modificar entes ejecutores y se van a poder modificar todos los datos del mismo:

- ✓ Nombre.
- ✓ Siglas.
- ✓ Ministerio
- ✓ Coordinador.
- ✓ Teléfono.
- ✓ Fax.
- ✓ Descripción.

RF02.13.03 Eliminar entes ejecutores

El sistema permitirá eliminar entes ejecutores mientras no tenga usuarios asignados o proyectos creados.

RF02.13.04 Buscar entes ejecutores

El sistema permitirá buscar entes ejecutores a través de un filtro de búsqueda.

RF02.13.05 Listar entes ejecutores

El sistema permitirá mostrar un listado de los entes ejecutores encontrados en el sistema.

2.3.2 Requisitos no funcionales

Se obtuvieron de los Módulos Administración y Configuración las siguientes propiedades o cualidades que el producto debe tener:

Requerimientos no funcionales de Seguridad:

- ✓ Autenticación obligatoria y segura.
- ✓ Acceso a la información según el rol.
- ✓ Realizar salvas periódicamente de la información contenida en la base de datos.
- ✓ El sistema debe recuperase ante fallos, ya sea por pérdida de conexión.
- ✓ Las secciones de los usuarios debe espirar después de 10 min de inactividad.

Requerimientos no funcionales de usabilidad:

- ✓ Permitir uso del teclado para realizar operaciones sobre el sistema.
- ✓ Poseer una interfaz agradable para el cliente de acuerdo a los estándares de diseño.
- ✓ Mostar la información de forma lógica y correctamente estructurada.
- ✓ Mostrar los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema en idioma español.
- ✓ Mostrar los valores de los campos numéricos utilizando separadores, según estándares.

Reusabilidad:

- ✓ Definir un modelo tres capas para el sistema.

Eficiencia:

- ✓ Responder en tiempos aceptables las peticiones que se realicen en el sistema.

Requerimientos no funcionales de Disponibilidad:

- ✓ El sistema debe estar accesible desde internet.

Requerimientos no funcionales de Rendimiento (Tiempo de respuesta, capacidad y rendimiento):

- ✓ Las páginas de la aplicación deben cargar en un tiempo inferior a 15 segundos.
- ✓ Debe garantizarse que con 300 usuarios conectados concurrentemente no disminuya el rendimiento y rapidez de la aplicación.
- ✓ El tiempo de carga de la aplicación debe ser de 10 a 25 segundos.

Requerimientos no funcionales de Portabilidad:

- ✓ El sistema debe permitir ser usado en cualquier plataforma.

Capacidad:

- ✓ Considerar características técnicas mínimas para la ejecución en clientes.

2.4 Modelo de Casos de Uso del sistema

El Modelo de Casos de Uso del Sistema sirvió como medio de comunicación entre el cliente y los desarrolladores del sistema en cuanto a las funcionalidades del mismo. Este modelo describe los requisitos funcionales de un actor, en términos de las interacciones que realiza con el sistema. Dichas interacciones se describen por medio de uno o más flujos de eventos que ocurren para llevar a cabo una tarea.

En esta etapa se obtuvieron los Actores del Sistema, el Diagrama de Casos de Uso del Sistema y las Descripciones de los Casos de Uso. En estas últimas se presentó una descripción detallada del flujo de eventos en función del responsable y las condiciones de excepción que contemplan estos flujos, conjuntamente con los prototipos de cada uno. El modelo de casos de uso del sistema proporciona la entrada fundamental para el diseño, la implementación y las pruebas.

Como parte del Modelo de Casos de Uso del Sistema de los Módulos Administración y Configuración los actores identificados fueron:

Tabla 2 Descripción de los actores del sistema

Actores	Descripción
Administrador	El actor es el encargado de administrar el sistema, en la creación, eliminación, modificación de usuarios que accederán a la aplicación.
Administrador ST (Secretaría Técnica)	El actor realiza todas las funcionalidades de **"Administrador"**, podrá gestionar la información de los usuarios a cualquier nivel

	del sistema (ST, M, EE, Embajada), además es el que podrá definir los roles para los diferentes usuarios.
Administrador EE (Entidad Ejecutora)	El actor realiza todas las funcionalidades de **"Administrador"**, podrá gestionar la información de los usuarios a nivel de EE.
Usuario	El actor podrá autenticarse en el sistema además de modificar su contraseña.
Gestor de Persona	Es el actor que se encarga de gestionar los datos de las personas
Gestor de EE (Entidad Ejecutora)	Es el actor que se encarga de gestionar los datos de las personas.

Para la elaboración del diagrama de casos de uso del sistema (DUCS) de los Módulos Administración y Configuración se aplicó el patrón: múltiples actores, roles comunes y CRUD completo. Finalmente quedando el DUCS de los Módulos Administración y Configuración de la siguiente forma:

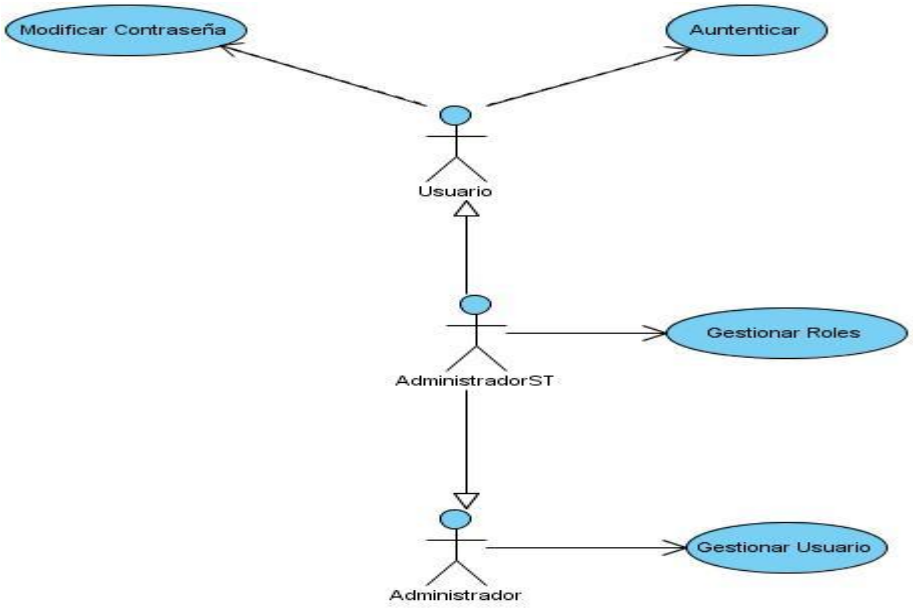

Figura 13 Diagrama de casos de uso del sistema del módulo Administración

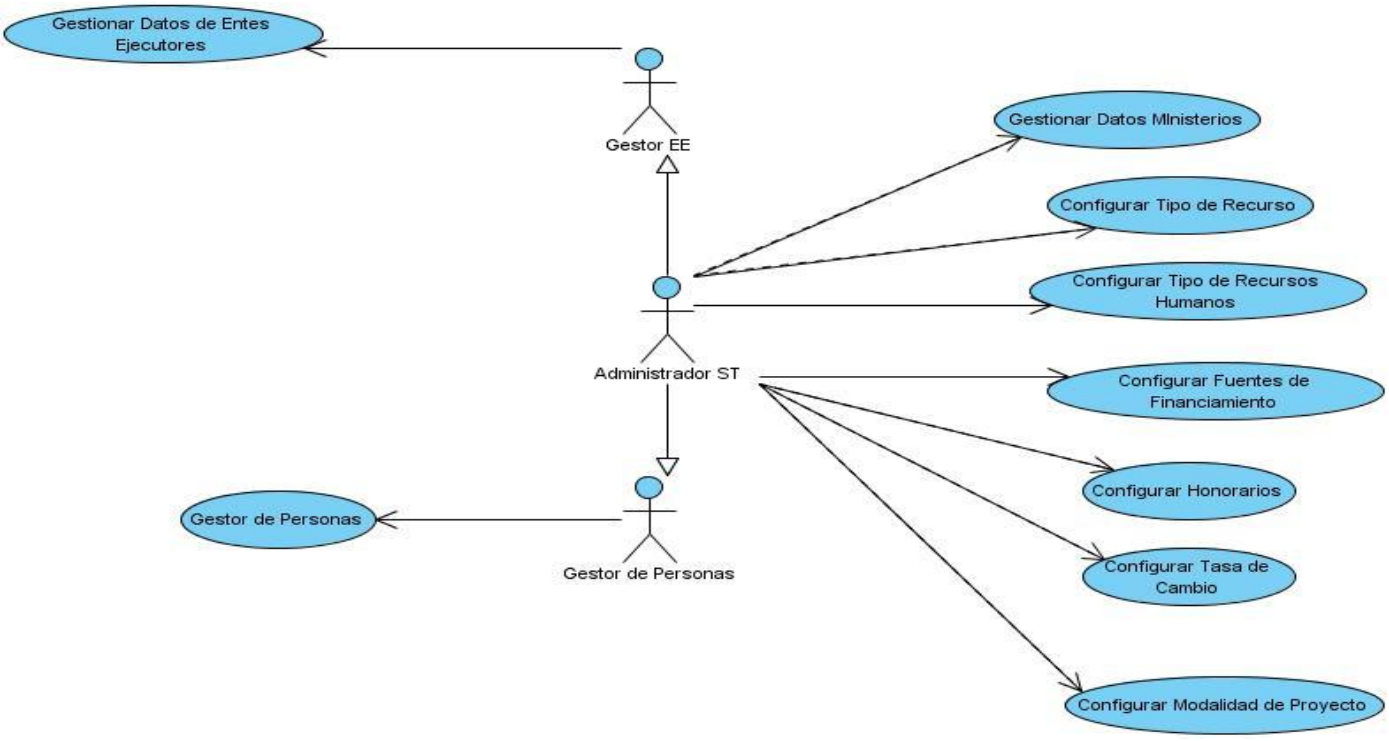

Figura 14 Diagrama de casos de uso del sistema del módulo Configuración

2.4.1 Especificación de los casos de uso del sistema

Se procede a la especificación de los casos de uso del sistema con el objetivo de describir en detalles cada una de las funcionalidades que deben ser implementadas. La descripción de los casos de uso constituye una guía para los desarrolladores y un documento de obligatorio cumplimiento en cuanto al desarrollo de funcionalidades en el sistema. Las especificaciones de los casos de uso de los Módulos Administración y Configuración se pueden ver en el documento modelo de casos de uso del sistema CCV de los Módulos Administración y Configuración. (Janier Acosta Aragón Y. L., 2010)

2.5 Subsistema de diseño

Los subsistemas de diseño constituyen una forma de estructurar los artefactos que conforman el modelo de diseño en estructuras más independientes. Los elementos de un subsistema comparten alguna propiedad en común y se integran para completar la misma función. Con el fin de separar los aspectos del diseño se identificaron los subsistemas de diseño relacionados con los Módulos Administración y Configuración del sistema CCV.

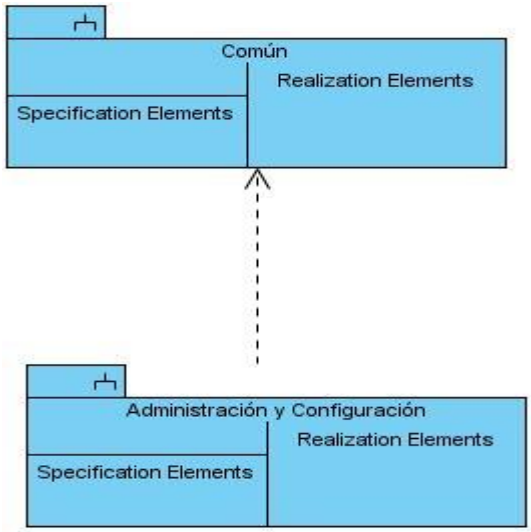

Figura 15 Subsistemas de diseño de los Módulos Administración y Configuración.

2.6 Paquetes de diseño

Para facilitar el desarrollo de un sistema complejo, resulta aconsejable lograr una mayor organización de las distintas partes del sistema para que el trabajo sea más fluido y ágil. La organización de las distintas partes del sistema se lleva a cabo mediante la agrupación de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que se relacionan de alguna manera. Con el objetivo de establecer una mejor organización y especialización de los elementos del diseño se identificaron y definieron los siguientes paquetes para el subsistema de Administración, Configuración de los Módulos Administración y Configuración del sistema CCV.

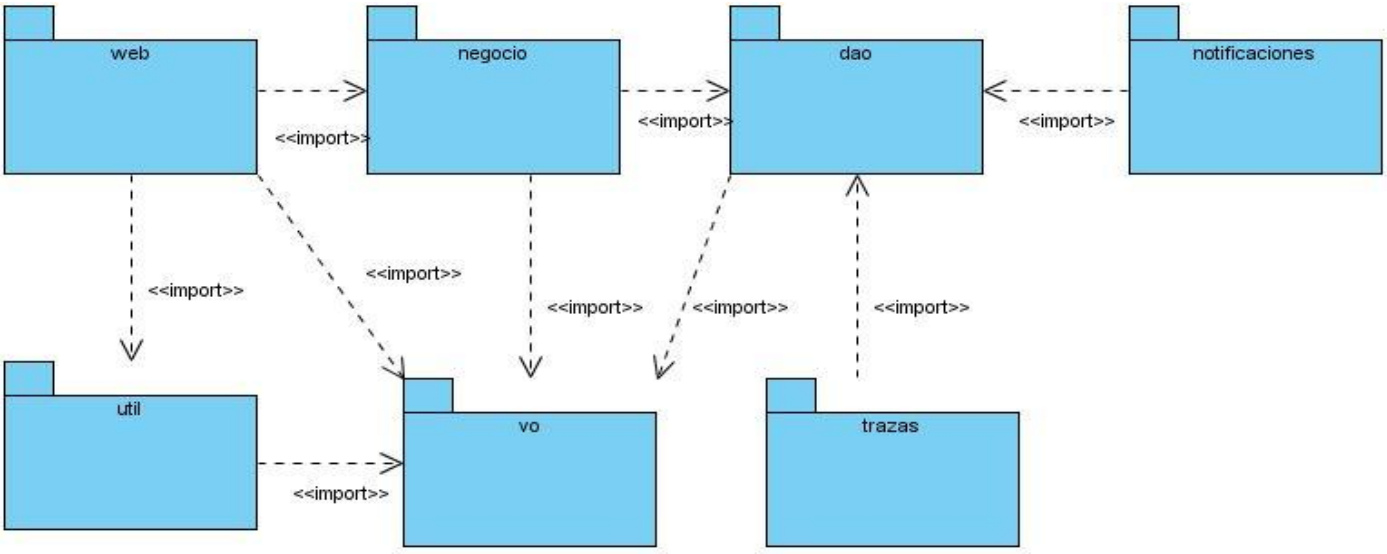

Figura 16 Paquetes de diseño de los Módulos Administración y Configuración.

2.7 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en como los requisitos funcionales y no funcionales tienen impacto en el sistema. En este modelo los casos de uso son realizados por las clases del diseño y sus objetos. Esto se representa por colaboraciones en el modelo de diseño. (Jacobson, 2000). En los siguientes epígrafes se especifican algunos de los artefactos que pertenecen al modelo de diseño.

2.8 Diagrama de clases del diseño

Los diagramas de clases del diseño son los encargados de representar las relaciones entre clases, interfaces así como la colaboración entre ellos. Constituyen la vista del diseño estático de un sistema

A continuación se muestran los diagramas de clases de diseño de algunos de los casos de uso de los Módulos Administración y Configuración del sistema CCV, el resto de los diagramas de clases del diseño se pueden consultar en el documento Modelo de diseño del sistema CCV de los Módulos Administración y Configuración. (Janier Acosta Aragón Y. L., 2010)

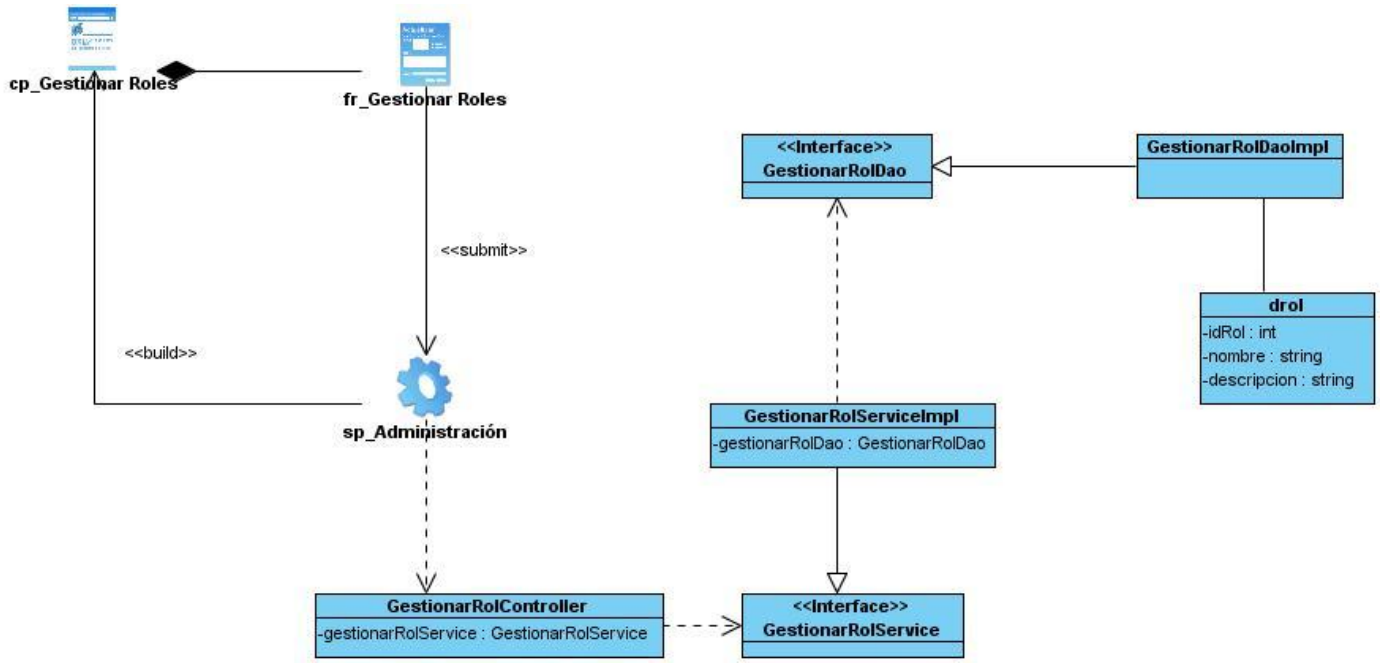

Figura 17 Diagrama de clases del diseño, CU Gestionar Roles

2.9 Diagrama de secuencia

En los diagramas de secuencia se muestran las interacciones entre objetos mediante transferencias de mensajes a objetos o subsistemas. En el diseño se utilizaron los diagramas de secuencia para representar la secuencia de acciones en un caso de uso donde los objetos del diseño implicados interactúan para llevar a cabo los casos de uso.

A continuación se muestran los diagramas de secuencia elaborados para los escenarios de los casos de uso principales de los Módulos Administración y Configuración del sistema CCV, el resto de los diagramas de secuencia del diseño se pueden consultar en el documento modelo de diseño del sistema CCV de los Módulos Administración y Configuración. (Janier Acosta Aragón Y. L., 2010)

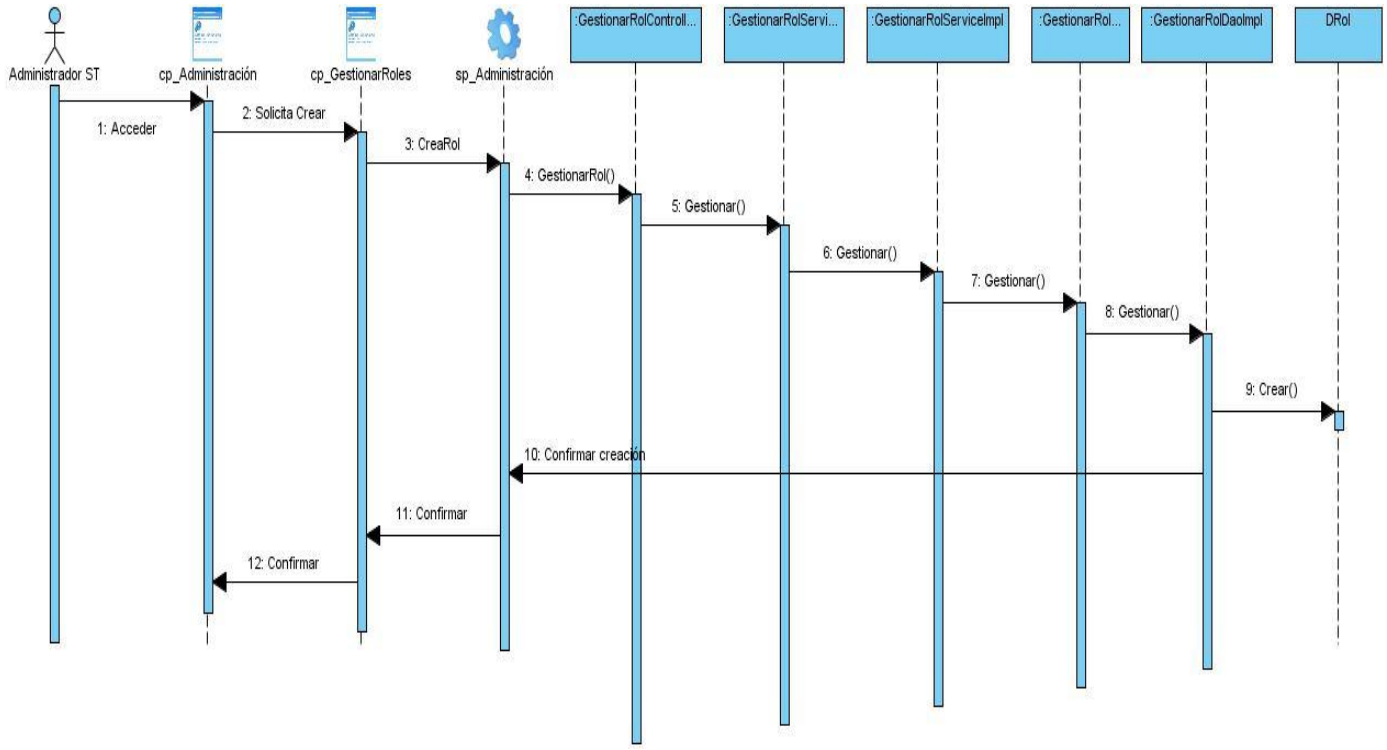

Figura 18 Diagrama de secuencia del CU Gestionar Rol, Sección Crear

2.10 Modelo de Datos

En el modelo de datos se representa la estructura que debe de tener la base datos para almacenar los objetos persistentes del sistema. Con el fin de garantizar la persistencia de los datos se modeló y normalizó el modelo de entidad relación de los Módulos Administración y Configuración del sistema CCV, donde se muestra un fragmento del modelo a continuación.

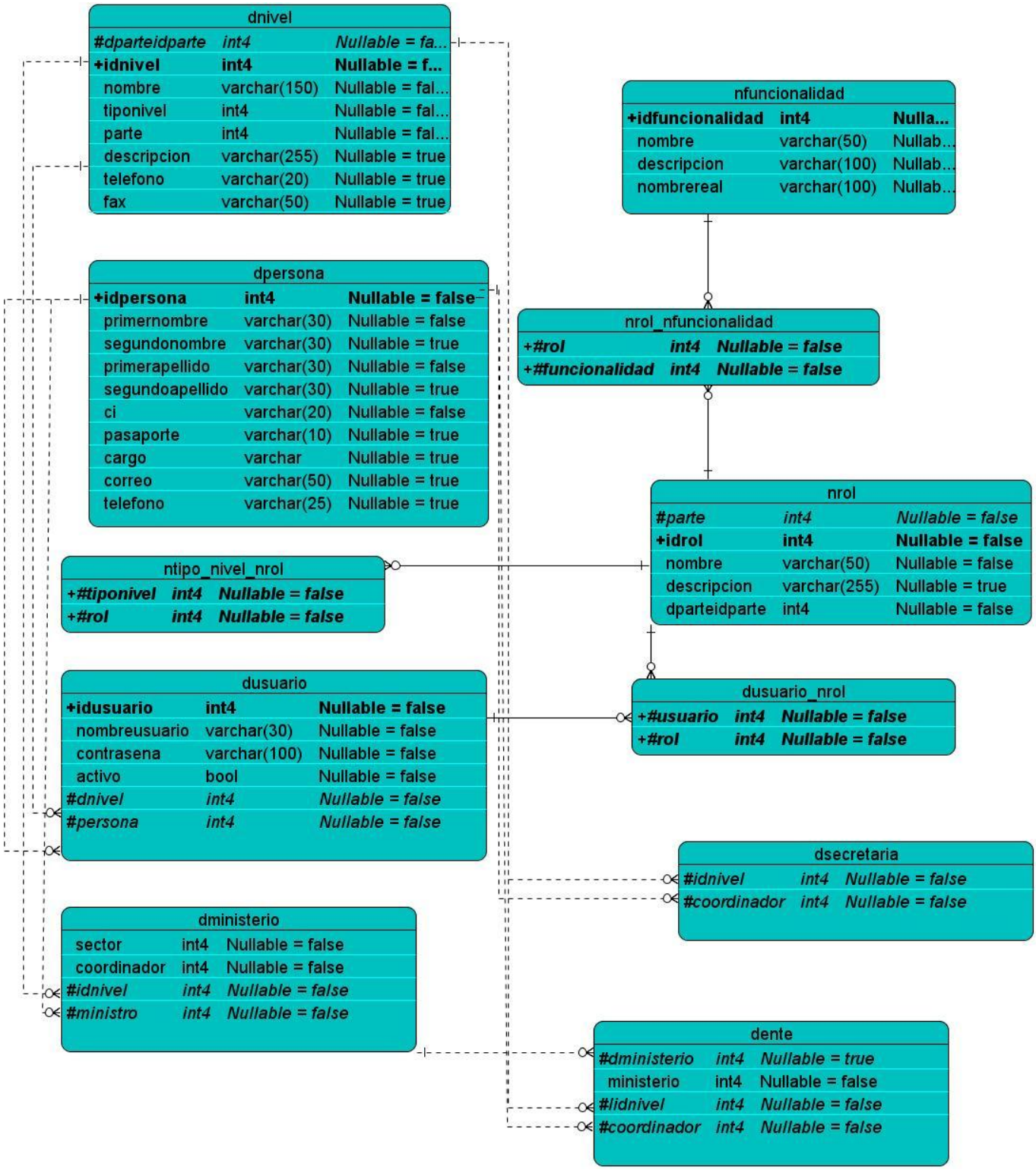
dnivel
#dparteidparte int4 Nullable = fa...
+idnivel int4 Nullable = f...
nombre varchar(150) Nullable = fal...
tiponivel int4 Nullable = fal...
parte int4 Nullable = fal...
descripcion varchar(255) Nullable = true
telefono varchar(20) Nullable = true
fax varchar(50) Nullable = true
nfuncionalidad
+idfuncionalidad int4 Nulla...
nombre varchar(50) Nullab...
descripcion varchar(100) Nullab...
nombrereal varchar(100) Nullab...
dpersona
+idpersona int4 Nullable = false
primernombre varchar(30) Nullable = false
segundonombre varchar(30) Nullable = true
primerapellido varchar(30) Nullable = false
segundoapellido varchar(30) Nullable = true
ci varchar(20) Nullable = false
pasaporte varchar(10) Nullable = true
cargo varchar Nullable = true
correo varchar(50) Nullable = true
telefono varchar(25) Nullable = true
nrol_nfuncionalidad
+#rol int4 Nullable = false
+#funcionalidad int4 Nullable = false
nrol
#parte int4 Nullable = false
+idrol int4 Nullable = false
nombre varchar(50) Nullable = false
descripcion varchar(255) Nullable = true
dparteidparte int4 Nullable = false
ntipo_nivel_nrol
+#tiponivel int4 Nullable = false
+#rol int4 Nullable = false
dusuario
+idusuario int4 Nullable = false
nombreusuario varchar(30) Nullable = false
contrasena varchar(100) Nullable = false
activo bool Nullable = false
#dnivel int4 Nullable = false
#persona int4 Nullable = false
dusuario_nrol
+#usuario int4 Nullable = false
+#rol int4 Nullable = false
dsecretaria
#idnivel int4 Nullable = false
#coordinador int4 Nullable = false
dministerio
sector int4 Nullable = false
coordinador int4 Nullable = false
#idnivel int4 Nullable = false
#ministro int4 Nullable = false
dente
#dministerio int4 Nullable = true
ministerio int4 Nullable = false
#lidnivel int4 Nullable = false
#coordinador int4 Nullable = false

Figura 19 Modelo Entidad Relación

2.11 Conclusiones Parciales

En este Capítulo se analizaron los aspectos que forman parte de solución referente al sistema CCV, específicamente de los Módulos Administración y Configuración donde se arribó a las siguientes conclusiones:

- ✓ El análisis del negocio propuesto, junto a la interacción con los clientes del sistema y aplicando técnicas de la ingeniería de requisitos, propició que se obtuvieran resultados satisfactorios en la identificación de los requerimientos, tanto funcionales como no funcionales que debe cumplir el sistema.
- ✓ La identificación y especificación de los artefactos del modelo del sistema facilitó un mayor entendimiento y un acuerdo común entre los clientes y los desarrolladores, en cuanto a la concepción de las funcionalidades que el sistema debe cumplir.
- ✓ La modelación de los artefactos del diseño, utilizando visual Paradigm y UML como lenguaje de modelado, propició un mayor entendimiento entre los involucrados de los Módulos Administración y Configuración del sistema CCV.
- ✓ La construcción de los artefactos del modelo de diseño permitió obtener cada uno de los elementos que soportan los requerimientos funcionales y no funcionales del sistema, siendo estos la entrada a las actividades de implementación.
- ✓ La aplicación de patrones, tanto de casos de uso como de diseño propiciaron obtener artefactos aceptables.

CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS

3.1 Introducción

La obtención de los artefactos desarrollados durante este trabajo sirvió de entrada al proceso de diseño del software. En el capítulo 3 se hace un análisis de la especificación de los requisitos de software. A la hora de lograr una mayor exactitud en la medición del software es necesario poner en práctica algunas de las métricas tanto para la especificación de requisitos como para el diseño. Para la validación se aplicaron la listas de chequeo para los requerimientos (ver **Anexo 1**). Las métricas de diseño son utilizadas para ayudar a que el diseño evolucione a un nivel superior de calidad.

3.2 Resultados de la Especificación de Requisitos

El levantamiento de requisitos es una de las etapas fundamentales en el desarrollo de un software. Durante esta actividad es importante establecer un acuerdo común entre el cliente y el equipo de desarrollo en cuanto a que es lo que se quiere construir. Para alcanzar los resultados esperados, fue importante primeramente, entender la definición y el alcance del problema que se estaba tratando de solucionar con el sistema. En esta etapa se identificaron cada una de las necesidades del cliente, obteniendo como resultado los requisitos funcionales y los no funcionales del sistema. Una vez terminada la captura de requisitos se procedió a la realización del Modelo de Casos de Uso del Sistema, donde fueron agrupados varios requisitos en un caso de uso hasta conformar el modelo.

3.3 Métricas de la calidad de la especificación

Para complementar la validación de la especificación de requisitos Davis y sus colegas (Pressman, 2005) proponen una lista de características que pueden emplearse para valorar la calidad correspondiente a la especificación de requisitos: especificidad (ausencia de ambigüedad), compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización.

A continuación se muestra los resultados de algunas características que forman parte de la lista de propuesta por (Davis, 1993).

Número de requisitos en la especificación*:* $n_r = n_f + n_{nf}$

n_r : Total de requisitos en una especificación.

n_f : Número de requisitos funcionales.

n_{nf} : Número de requisitos no funcionales.

$n_r = 45 + 27 = 72$

Para determinar la especificidad (ausencia de ambigüedad) de los requisitos. Davis sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

Especificidad: $Q_1 = n_{ui} / n_r$

Q_1 : Especificidad de los requisitos.

N_{ui} : Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

$$Q_1 = \frac{72}{72} = 1.00$$

Cuando más cerca de uno este valor de Q_1 menor será la ambigüedad de la especificación.

La compleción de los requisitos funcionales puede determinarse calculando la relación:

Compleción: $Q_2 = \frac{n_A}{n_A + n_B}$

Q_2 : Compleción de los requisitos.

N_A : Número de requisitos completos.

N_B : Número de requisitos pobremente especificados.

$$Q_2 = \frac{72}{72 + 0} = 1.00$$

La relación Q_2 mide el porcentaje de funciones necesarias que se han especificado para un sistema. Sin embargo, no trata los requisitos no funcionales. Para incorporarlos a una métrica global completa, debemos considerar el grado de validación de los requisitos.

Corrección: $Q_3 = \frac{n_c}{n_c + n_{nv}}$

Q_3 : Corrección de los requisitos.

N_c : Número de requisitos que se han validado como correctos.

N_{nv} : Número de requisitos que no se han validado como correctos todavía.

$$Q_3 = \frac{72}{72+0}$$

$$Q_3 = \frac{72}{72} = 1.00$$

Comprensión: $Q_4 = \frac{n_{cu}}{n_r}$

Q_4 : Comprensión de los requisitos.

N_{cu} : Número de requisitos que todos los revisores entienden.

$$Q_4 = \frac{72}{72} = 1.00$$

Consistencia interna: $Q_6 = \frac{n_u - n_n}{n_u}$

Q_6 : Consistencia interna.

N_u : Número de requisitos especificados.

N_n : Número de requisitos en conflicto con otros requisitos en la especificación.

$$Q6 = \frac{72-0}{72} = 1.00$$

Consistencia externa: $Q_7 = \frac{n_{EC}}{n_r}$

Q_7 : Consistencia externa.

N_{EC} : Número de requerimientos que son consistentes con otros documentos.

$$Q_7 = \frac{72}{72} = 1.00$$

No redundancia: $Q_8 = \frac{n_f}{n_u}$

n_f : Número de requisitos funcionales.

N_u : Número de requisitos funcionales únicos.

$$Q_8 = \frac{72}{72} = 1.00$$

3.4 Métricas del diseño

El diseño sin medición es una alternativa que no se puede aceptar cuando se requiere de métodos para lograr una mayor eficacia en el diseño de un software. Como parte de esos métodos que se toman para obtener una mayor exactitud en la medición se ponen en práctica algunas de las métricas de diseño más comunes.

3.4.1 Métricas de diseño a nivel de componentes

Métricas de acoplamiento

Para medir el nivel de conectividad y dependencia de los Módulos Administración y Configuración con otros módulos se aplica la medida propuesta por Dhama (**ver Capítulo 1**), con el fin de determinar el nivel de independencia y reutilización que posee el módulo según el diseño propuesto. Definiendo:

Acoplamiento de flujos de datos de control:

d_i = 1 (número de parámetros de datos de entrada)

c_i = 1 (número de parámetros de control de entrada)

d_0= 1 (número de parámetros de datos de salida)

c_0 = 1 (número de parámetros de control de salida)

Acoplamiento global:

g_d = 0 (número de variables globales usadas como datos)

g_c = 0 (número de variables globales usadas como control)

Acoplamiento de entorno:

w = 1 (número de módulos llamados **Figura 20**)

r = 3 (número de módulos que llaman a los Módulos Administración y Configuración **Figura 21)**

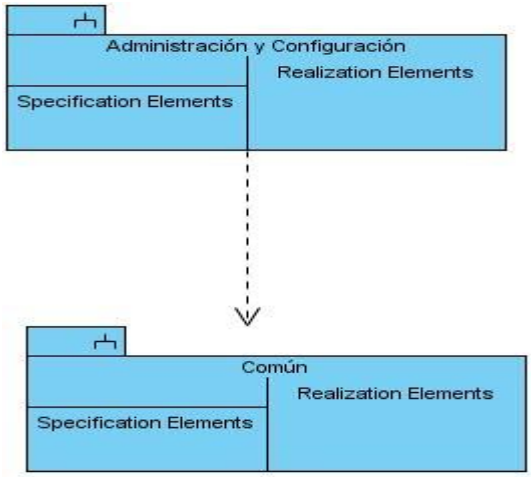

Figura 20 Expansión de los Módulos Administración y Configuración

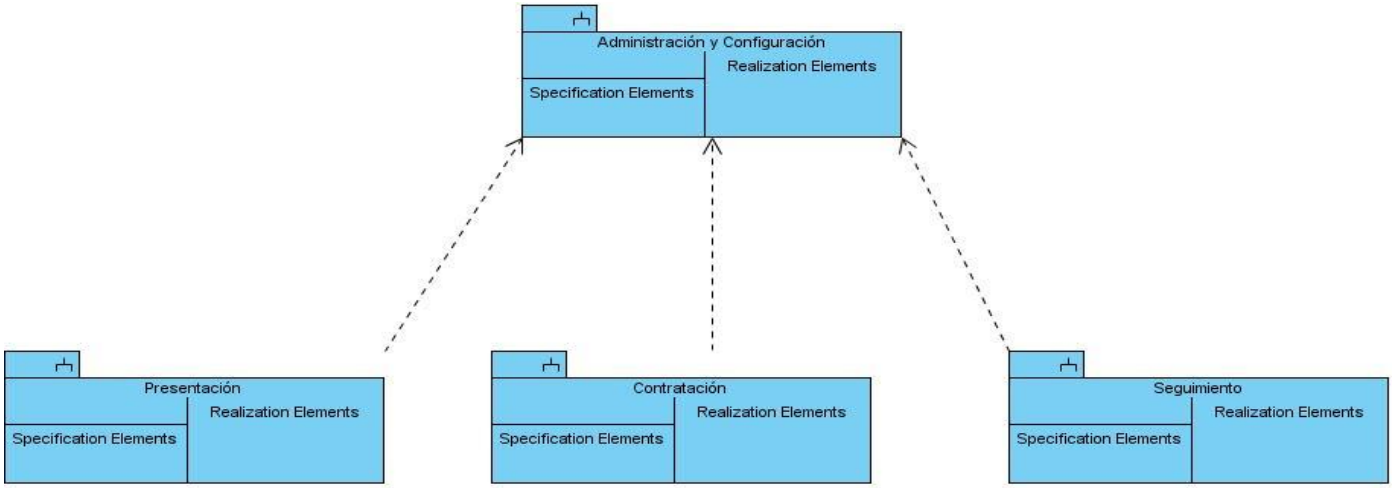

Figura 21 Concentración de los Módulos Administración y Configuración

Indicador de acoplamiento del módulo:

m_c= K / M

Donde K = 1, constante de proporcionalidad según el autor

$M = d_i + axc_i + d_0 + bxc_0 + g_d + cxg_c + w + r$

a=b=c=2

$m_c = 1/(1 + 2x1 + 1 + 2x1 + 0 + 2x0 + 1 + 3$

m_c= 0.1

El valor de 0.1 del indicador de acoplamiento de los Módulos Administración y Configuración nos sugiere el grado de acoplamiento del mismo, lo que sugiere que existe dependencia con otros módulos para su funcionamiento.

Métricas de cohesión

Una buena práctica para el diseño lo constituye el lograr una alta cohesión. Para medir el índice de cohesión de los elementos que conforman el diseño de los Módulos Administración y Configuración se aplicó la métrica de Bieman y Ott, la cual se detalla en el **Capítulo 1**.

Para proceder al análisis de cada uno de estos conceptos se recogieron los datos que ilustran el nivel de uso de las principales clases diseñadas de los Módulos Administración y Configuración (ver **Tabla 3)**.

Tabla 3 Nivel de uso de las principales clases diseñadas

Clases	Usabilidad
1.GestionarRolController	1
2.GestionarUsuarioController	1
3.GestionarPersonaController	1
4.GestionarEntesController	1
5.GestionarMinisteriosController	1
6.FuenteFinanciamientoController	1
7.TipoRecursoController	1
8.TipoRecursoHumanoController	1
9.AdministracionController	0
10.ConfiguraciónController	0
11.ModalidadProyectosController	1
12.HonorarioController	1
13.CambiarPasswordController	1
14.GestionarRolServiceImpl	1
15.GestionarUsuarioServiceImpl	1
16.GestionarPersonaServiceImpl	1
17.FuenteFinanciamientoServiceImpl	1
18.TipoRecursoServiceImpl	1
19.TipoRecursoHumanoServiceImpl	1
20.ModalidadProyectosServiceImpl	1

21.HonorariosServiceImpl	1
22.GestionarEntesServiceImpl	1
23.GestionarMinisteriosServiceImpl	1
24.GestionarRolDaoImpl	1
25.GestionarPersonaDaoImpl	1
26.GestionarUsuarioDaoImpl	1
27.GestionarEntesDaoImpl	1
28.GestionarMinisteriosDaoImpl	1
29.ConfigurarFuenteFinanciammientoDaoImpl	1
30.ConfigurarTipoRecursoHumanoDaoImpl	1
31.ConfigurarTipoRecursoDaoImpl	1
32.ConfigurarHonorariosDaoImpl	2
33.ConfigurarModalidadProyectosDaoImpl	1
34.CambiarPasswordDaoImpl	2

La cohesión funcional se determina con dos enfoques:

1. Determinando la cohesión funcional fuerte (CFF) y la pegajosidad: se obtienen cuando el resultado de la métrica es de 1.

Se define como:

CFF = número de súper adhesivos (i) / número de elementos (i)

2. Determinando la cohesión funcional débil (CFD):

Se define como:

CFD = número de adhesivos (i)/ número de elementos (i)

Adhesivo. Se le llamará adhesivo a un elemento que aparece en una o más rebanadas.

Súper adhesivo. Se denomina súper adhesivo a un elemento que está en todos los elementos de un módulo. **(i)** Se define como la muestra.

Según los datos de las clases analizadas se tiene que:

Número de elementos = 34

Número de súper adhesivos (i) = 0

Número de adhesivos (i) = 32

CFD = número de adhesivos (i) / número de elementos (i)

CFD = 32/ 34

CFD = 0.94

CFF = número de súper adhesivos (i) / número de elementos (i)

CFF = 0 / 34

CFF = 0

Los resultados demuestran que no hay una cohesión funcional fuerte, pero la relación del número de clases adhesivas con el número total de elementos de la muestra, determinados por la CFD = 0.94, está cercana a 1, lo que demuestra que el diseño de las clases de los Módulos Administración y Configuración poseen una cohesión funcional con un 94% de fortaleza.

3.4.2 Métricas orientadas a clase. Tamaño de la clase (TC)

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

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

Los umbrales que plantean algunos especialistas para estas métricas se muestra en la **Tabla 4**, estos fueron los aplicados en el diseño para los Módulos Administración y Configuración.

Tabla 4 Umbrales para TC

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

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

Se presentó un total de 34 clases para un promedio de atributos de 0.67 y un promedio de operaciones de 6.14.

Los módulos Administración y Configuración presentan 34 clases pequeñas de acuerdo con los umbrales que se presentan en la **Tabla 5**

Tabla 5 Resultados

Umbral	**Tamaño**	**Cantidad de Clases**
<=20	Pequeño	34

3.5 Conclusiones Parciales

En este Capítulo se realizó un análisis de los requisitos que se obtuvieron durante la fase de desarrollo del sistema CCV. La aplicación de métricas para validar el sistema arribó a las siguientes conclusiones:

- ✓ La métrica de especificación de requisitos utilizada permitió que se hicieran varias iteraciones hasta obtener con valor uno a las características de especificidad (ausencia por ambigüedad), compleción, corrección, comprensión, no redundancia y consistencia interna y externa.
- ✓ El indicador de acoplamiento de los Módulos Administración y Configuración determinó que existe dependencia con otros módulos para su funcionamiento.
- ✓ El diseño de las clases de los Módulos Administración y Configuración posee una cohesión funcional con índice de fortaleza de un 94%, lo que demuestra que sus elementos están altamente cohesionados.
- ✓ La métrica de tamaño de clase utilizada sirvió para evidenciar que la mayoría de las clases son reutilizables y no poseen mucha responsabilidad, lo que provoca que se lleve a cabo la implementación de una forma más fácil.
- ✓ Las métricas de diseño aplicadas son parte de que el diseño mantenga altos valores de calidad para así contribuir a la implementación exitosa del sistema.

CONCLUSIONES GENERALES

Al concluir el presente trabajo se arriba a las siguientes conclusiones:

- ✓ El estudio y selección de la metodología de desarrollo de software y de los aspectos vinculados a la ingeniería de requisitos, patrones de diseño y de casos de uso permitió la modelación del sistema teniendo presente las necesidades del cliente.
- ✓ La metodología RUP, la herramienta Visual Paradigm y el lenguaje de modelado UML fueron utilizados para modelar los artefactos correspondientes, definiendo las clases conceptuales del dominio, los actores, los casos de uso del sistema y los artefactos del modelo de diseño.
- ✓ La realización del modelo de dominio estableció una mayor estructuración en la organización de los tipos de objetos más importantes en el contexto del sistema.
- ✓ La especificación de los requisitos de software permitió definir las funcionalidades con que debería contar el sistema y logró establecer las metas y objetivos del software.
- ✓ El modelo de casos de uso del sistema posibilitó obtener los Actores del Sistema, el Diagrama de Casos de Uso del Sistema y las Descripciones de los Casos de Uso con una descripción detallada del flujo de eventos de estos casos de uso.
- ✓ Los artefactos generados del modelo del diseño permitió obtener los elementos que deberán ser implementados.
- ✓ La aplicación de métricas para medir el diseño y la especificación de los requisitos demostró altos valores de calidad para contribuir a una implementación exitosa del sistema.

RECOMENDACIONES

Con el objetivo de lograr mejores resultados en las siguientes fases del desarrollo del software se recomienda:

- ✓ Elaborar los artefactos restantes pertenecientes al diseño, que son necesarios para continuar con el desarrollo de los Módulos Administración y Configuración.
- ✓ Aplicar ingeniería inversa de código utilizando Visual Paradigm, para mantener actualizado los diagramas de clases del diseño.
- ✓ Desarrollar la implementación de los Módulos Administración y Configuración del sistema CCV.

BIBLIOGRAFÍA

A., D. (2001). *IRqA y el desarrollo de proyectos: Experiencias Prácticas. I Jornadas de Ingeniería de Requisitos.* Seville, Spain.

Alarcón, R. (2000). *Diseño Orientado a Objetos con UML.* Obtenido de http://www.scribd.com/doc/490192/Diseno-Orientado-a-Objetos-con-UML-by-Raul-Alarcon

Albacete, E. P. (2003). *Diseño y Programación Orientada a Objetos.* Obtenido de http://www.info-ab.uclm.es/asignaturas/42579

Association, I. S. (2004). *IEEE Standards Association.*

Ávila, S. J. (15 de 7 de 2005). *http://www.mentores.net*. Obtenido de http://www.mentores.net/articulos/intro_microsoft_sol_frame.htm

Bauer, F. L. (1972). *oftware Engineering Information Processing.* Amsterdam: North Holland Publishing Co.

Beck, K. (1999). *Extreme Programming Explained.* Addison Wesley.

Bieman, J. y. (1994). Measuring Functional Cohesion. IEEE Trans. Software Engineering,.

Bohem, B. W. (1976). *Software Engineering.*

Card, D. y. (1990). *Measuring Software.* Prentice-Hall.

Carlos, U. R. (s.f.). *http://kybele.escet.urjc.es.* Obtenido de http://kybele.escet.urjc.es/Documentos/ISI/Ingenieria%20de%20Requisitos.pdf

Dapena, M. D. (s.f.). Obtenido de www.inf.udec.cu

Davis, A. (1993). *Identifying and Measuring Quality in a Software Requirements Specification.* Proc. 1st.

Dhama, H. (1995). *Quantitative Models of Cohesion and oupling in Software.* Journal of Systems and Software.

Díez, A. (2001). *RqA y el desarrollo de proyectos: Experiencias Prácticas.I Jornadas de Ingeniería de Requisitos.* España.

Durán A., B. B. (1999). *A Requirements Elicitation Approach Based in Templates and Patterns. Workshop de Engenharia de Requisitos.* Buenos Aires, Argentina.

Escalona, M. J. (2002). *Ingeniería de Requisitos en Aplicaciones para la Web –Un estudio comparativo.* Obtenido de http://www.ldc.usb.ve/~abianc/materias/ci4712/Ingenieria%20de%20Requisitos.pdf

Henry, S. y. (1981). SoftwareS tructure Metrics. En S. y. Henry, *SoftwareS tructure Metrics* (págs. 510-518). IEEE Trans. Software Engineering.

Heras, M. V. (2009). *Análisis y diseño del módulo Presentación del sistema del Convenio Integral de Cooperación Cuba-Venezuela.* La Habana.

IBM. (1997). *Developing Object Oriented Software. IBM Object Oriented Technology Center.* Prentice-Hall.

Insfrán, E. P. (2002). Requirements Engineering-Based Conceptual Modeling. Requirements. En E. P. Insfrán, *Requirements Engineering-Based Conceptual Modeling. Requirements* (pág. 1).

International., V. P. (2005). *http://www.visual-paradigm.com/product/vpuml/.*

Jacobson, I. B. (2000). *El Proceso Unificado de Desarrollo de Software.* Madrid, España: Addison Wesley, 2000. ISBN 84-7829-036-2.

Janier Acosta Aragón, Y. L. (2010). *Modelo de casos de uso del sistema de los módulos Administración y Configuración del sistema CCV.* La Habana.

Janier Acosta Aragón, Y. L. (2010). *Modelo de diseño de los módulos Administración y Configuración del sistema CCV.* La Habana.

Jeffries, R. A. (2001). *Extreme Programming Installed.* Addison Wesley.

Juan, F. J. (2002). *GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO.* Obtenido de http://www.willydev.net: http://www.willydev.net/InsiteCreation/ElFlecha.Aspx?q=patrones

Koch, N. (1999). *A comparative study of methods for Hypermedia Development. Technical Report 9905.* Munich, Germany.: Ludwig-Maximilian-University.

Kruchten, P. (2000). *The Rational Unified Process An Introduction, Second Edition.* Addison Wesley,ISBN 0-201-70710-1.

Larman, C. (1999). *UML y Patrones. Introducción al Análisis y Diseño Orientado a Objetos.* Mexico: Addison Wesley. ISBN 970-17-0261-1.

Larrea, J. X. (2005). *http://www.mentores.net/*. Obtenido de http://www.mentores.net/default.aspx?tabid=104&type=art&site=183&parentid=32

Letelier, P. (2003). *Proyecto Docente e Investigador.*

López Barrio, C. (2005). *Metodología de desarrollo: Programación Extrema . s.l. : Programa de Doctorado Ingeniería de Sistemas Electrónicos para Entornos Inteligentes.*

Lowe, D. H. (1999). *Hypermedia and the Web. An Engineering approach.* John Wiley & Son.

Ltd, S. S. (2000-2008). *http://www.sparxsystems.es.*

McCabe, T. y. (1994). Software Complexity,,. En T. y. McCabe. Crosstalk, vol. 7, n." 12.

Mestras, J. P. (2004). *http://www.fdi.ucm.es.* Obtenido de ttp://www.fdi.ucm.es: http://www.fdi.ucm.es/profesor/jpavon

Olsina, L. (1998). *Building a Web-based information system applying the hypermedia flexible process modeling.*

Övergaard, G. y. (2004). *Use Cases Patterns and Blueprints.* Addison Wesley Professional.

Övergaard, G. y. (2004). *Use Cases Patterns and Blueprints.* Addison Wesley Professional.

Pan, D. Z. (2001). *Requirements Engineering Techniques. Internal Report. Department of.* University of Calgary. Canada.

Peña, R. (1998). *Diseño de programas. Formalismo y abstracción.*

Pressman, R. S. (2005). *Ingeniería del Software. Un Enfoque Práctico. Quinta Edición.*

Raghavan, S. Z. (1994). *Lectures Notes of Requirements Elicitacitation.* Educational Materials.

Sears, A. (1993). LayouAt ppropriateness: A Metric for Evaluating User Interface Widget Layout,,,. IEEE Trans. Software Engineering.

Sornerville, I. y. (1997). *Requirements ngitwe-ring.* Wiley.

Valencia, U. P. (28 de 7 de 2003). *http://www.dsic.upv.es.* Obtenido de www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc

Vilain, P. S. (2000). *A diagrammatic Tool for Representing User Interaction in UML.* York, England.: Lecture Notes in Computer Science. Proc. UML'2000.

Weidenhaupt, K. P. (1999). Scenarios in Systems Development: Current Practice. En K. P. Weidenhaupt, *Scenarios in Systems Development: Current Practice.* (págs. 34-35). IEEE Software.

Zapata, L. G. (2005). *Herramientas de desarrollo de ingenieria de software para LINUX.*

Zelkovitz, M. V. (1976). *rincipies of Software Engineering and Design.* Prentice Hall.

Zuñiga, Y. L. (2008). *Análisis del Módulo de Soporte del Proyecto Informatización del Convenio Integral de Cooperación Cuba-venezuela.* La Habana.

ANEXOS

Anexo 1- Lista de chequeo para la especificación de requisitos.

Evaluación
Claridad
¿Los requisitos se escriben en lengua comprensible para el usuario/cliente?
¿Hay requisitos que tienen más de una interpretación?
¿Cada requisito característico del producto final se describe con una terminología única?
¿Hay un glosario en el cual los requisitos significativos y específicos están en términos definidos y claros?
¿Se entienden los requisitos para ponerlos en ejecución por un grupo independiente?
Completo
¿El requisito tiene un contenido específico?
Están especificados los cambios posibles a los requisitos.
La probabilidad de cambios está especificada para cada requisito.
¿Se definen todos los términos en cada uno de los requisitos determinados?
¿Tiene el artefacto un índice bien definido?
¿Existen áreas donde está incompleta la información debido a que el desarrollo aún no ha comenzado a especificarse?
¿La información que falta se define en el requisito?
¿Algún requisito necesita una especificación detallada?
¿Algún requisito necesita ser menos especificado?
¿Todos los requisitos se describen ellos mismos?
¿Están incluidos todos los requisitos relacionados con la funcionalidad?
¿Hay requisitos que produzcan inquietud?
¿Están incluidos todos los requisitos relacionados con el funcionamiento?
¿Están incluidos todos los requisitos relacionados con sus cualidades?
¿Están incluidos todos los requisitos relacionados con las interfaces externas?
¿Están incluidos todos los requisitos relacionados con las bases de datos?
¿Están incluidos todos los requisitos relacionados con el software?
¿Están incluidos todos los requisitos relacionados con el hardware? (No todavía)
¿Están incluidos todos los requisitos relacionados con las entradas de datos?
¿Están incluidos todos los requisitos relacionados con las salidas datos?
¿Están incluidos todos los requisitos relacionados con los mensajes de error?
¿Están incluidos todos los requisitos relacionados con la seguridad? (No todavía)
¿Están incluidos todos los requisitos relacionados con la capacidad de mantenimiento del software? (No todavía)
¿Están incluidos todos los requisitos relacionados con la instalación? (No todavía)
Consistencia (Duda)
¿Hay requisitos que describen el mismo objeto que estén en conflicto con otros requisitos con respecto a la terminología?
¿Hay requisitos que describen al mismo objeto que estén en conflicto con respecto a las características?

¿Hay requisitos que describan dos o más acciones que estén en conflicto temporalmente?
Rastreabilidad
¿Los requisitos fueron rastreados en todos los módulos?
Comprobabilidad
¿Existen requisitos que sean imposibles de cumplir?
Modificalidad
¿En el documento los requisitos se organizan de manera clara y lógica?
Contenido General
¿Cada requisito es relevante al problema y a su solución?
¿El artefacto cumple con la plantilla especificada por calidad interna?
Prototipo de Interfaz
¿Se especifican todas las interfaces utilizadas?
¿Se especifican todas las interfaces del hardware?
¿Se especifican todas las interfaces del software?
¿Se especifican todas las interfaces de comunicaciones?
¿Se especifican todos los requisitos del diseño de interfaz?
Hardware
¿Se especifica el tipo de hardware que necesita la aplicación?
Software
¿Se especifica el software requerido y el sistema operativo?
Trazabilidad
Cada requisito obedece a una necesidad específica de usuario.
Cada requisito tiene su origen en una fuente (documento o persona) específica.
Cada requisito se puede rastrear hacia delante su incorporación en determinados módulos.
Verificabilidad
Cada requisito es implementable.
Para cada requisito existe un procedimiento que, ejecutado por una persona o máquina, permite verificar si se cumple
Hay algún requisito que se va a expresar en términos verificables más adelante.

Anexo 2- Acta de aceptación de la documentación del módulo Administración.

albet INGENIERÍA Y SISTEMAS | PROYECTOS

ACTA DE ACEPTACIÓN

Producto: Sistema de Gestión para Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela.

Categoría: Aceptación CU Módulo de Administración.

Fecha de la conciliación: 28/07/2008

Involucrados en el proceso:

Por la parte del Cliente (MENPET): Sandra Cortés
Por la parte del Suministrador (ALBET): Ing. Nahuel Massón

Observaciones del proceso:

Por acuerdo entre las partes involucradas en el proceso se Acepta el Documento CU del Módulo de Administración de la versión v.1.0 del Sistema de Gestión para Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela, con fecha 28 de julio de 2008.

Para que conste la aceptación de la descripción de los CU y requerimientos del Módulo de Administración, dando fe al acuerdo, firman la presente los principales representantes de las Partes.

Sandra Cortés
Representante MENPET

Ing. Nahuel Massón
Representante ALBET

Referencia: CV-SW-CC-005

ALBET, S.A.
Centro de Negocios Ministro Fidel Castro Barcelona. Oficina 502. Avenida 5ta. e/ 76 y 78. Miramar. Playa. Ciudad Habana. Cuba
Tel/Fax: +53 (7) 837 2407
E-mail: albet@albet.cu

Anexo 3- Acta de aceptación de la documentación del módulo Configuración.

albet INGENIERÍA Y SISTEMAS | PROYECTOS

ACTA DE ACEPTACIÓN

Producto: Sistema de Gestión para Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela.

Categoría: Aceptación CU Módulo de Configuración.

Fecha de la conciliación: 28/07/2008

Involucrados en el proceso:

Por la parte del Cliente (MENPET): Sandra Cortés
Por la parte del Suministrador (ALBET): Ing. Nahuel Massón

Observaciones del proceso:

Por acuerdo entre las partes involucradas en el proceso se Acepta el Documento CU del Módulo de Configuración de la versión v.1.0 del Sistema de Gestión para Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela, con fecha 28 de julio de 2008.

Para que conste la aceptación de la descripción de los CU y requerimientos del Módulo de Configuración, dando fe al acuerdo, firman la presente los principales representantes de las Partes.

Sandra Cortés
Representante MENPET

Ing. Nahuel Massón
Representante ALBET

Referencia: CV-SW-CC-010

ALBET, S.A.
Centro de Negocios Miramar, Edificio Barcelona, Oficina 302, Avenida 5ta e/ 76 y 78, Miramar, Playa, Ciudad Habana, Cuba
Tel/Fax: +53 (7) 837 2407

Anexo 4 - Acta de aceptación de los Módulos Administración y Configuración del sistema CCV.

Acta de Aceptación

albet INGENIERÍA Y SISTEMAS

Producto: Sistema de Gestión para el seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba Venezuela (CCV)

Categoría de las pruebas: Revisión de Módulos Presentación, Administración y Configuración del CCV versión 1.0.

Fecha de conciliación: 20 de agosto de 2008

Involucrados en el proceso:

- **Por la parte del Cliente (CCV)**: Sandra Cortés
- **Por la parte desarrolladora (ALBET)**: Ing. Nahuel Massón Padilla
- **Observador independiente (CALISOFT)**: Ing. Dayami Rodríguez Brito
- **Observador independiente (CALISOFT)**: Ing. Roig Calzadilla Diaz

Observaciones del proceso:

Durante el proceso de pruebas de aceptación se identificaron un conjunto de cambios y mejoras necesarias que quedaron registrados adecuadamente en el correspondiente documento de Respuestas a las No Conformidades detectadas, con sus respectivas observaciones. Teniendo en cuenta que las No Conformidades han sido debidamente respondidas, ejecutadas por el Equipo de Desarrollo y validada la eficacia de los mismos por los clientes, se ha tomado el acuerdo de aceptar el **"Sistema de Gestión para el seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba Venezuela"** para la versión 1.0, con fecha 20 de agosto de 2008, el cual se anexa a esta Acta, estableciendo de esta forma la condición necesaria y suficiente para su despliegue.

Para que conste la aceptación de los resultados de las pruebas y por tanto la aceptación **del "Sistema de Gestión para el seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba Venezuela"**, dando fe del acuerdo firman la presente los principales representantes de las partes.

Sandra Cortés
(Por la parte del Cliente)

Ing. Nahuel Massón Padilla
(Por la parte Cubana - ALBET)

Ing. Roig Calzadilla Diaz
(Observador IndependienteCALISOFT)

GLOSARIO

Actor: Conjunto coherente de roles que los usuarios de casos de uso desempeñan cuando interaccionan con estos.

Artefactos: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Captura de Requisitos: Proceso durante el cual se identifica un problema y se especifican los requisitos que debe cumplir un producto de software.

Caso de Uso: Secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de las secuencias.

Cliente: Persona o empresa que contrata al desarrollador de software.

Diagrama: Representación gráfica de un conjunto de elementos, usualmente representado como un grafo conectado de vértices (elementos) y arcos (relaciones).

Ente Ejecutor: Empresa o institución que lleva a cabo proyectos que se firman en las Comisiones Mixtas.

Ministerio: Dirigen las actividades de los entes ejecutores y controlan los proyectos que estos ejecutan.

Microsoft: es una empresa multinacional estadounidense dedicada al sector de la informática. Microsoft desarrolla, fabrica, licencia y produce software y equipos electrónicos.

Prototipo: Maqueta visual funcional o no de la futura aplicación. Este puede ser una imagen o una aplicación software que simule funcionalidades del software.

Requerimientos: Condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.

Secretaría Técnica: Entidad en cada país encargada de coordinar todas las actividades necesarias para el buen desarrollo de cada proyecto.

Usuario: Persona o grupo de personas dentro de la empresa que utilizan el software desarrollado.