Universidad de las Ciencias Informáticas

Facultad 15

Título:

Ingeniería de Requisitos del Módulo Reportes del Sistema Nacional Público para el Seguimiento de Inversiones y Sectores (SINAPSIS).

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autora: Dalgys La Rosa Morales.

Tutor: Ing. Abdiel Matos Nieto.

Consultante: Msc. Ana M. Chaviano Hernández.

Ciudad de la Habana, 2010.

"Año del 52 Aniversario del Triunfo de la Revolución"

FRASE

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio conscientes de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte".

Ernesto Ché Guevara de la Serna.

DECLARACIÓN DE AUTORÍA

Declaro ser la única autora de este trabajo y autorizo a la facultad 15 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los __ días del mes de _____ del año 2010.

Dalgys La Rosa Morales	Ing. Abdiel Matos Nieto
_____	_____
(Autora)	(Tutor)

AGRADECIMIENTOS

A la Revolución Cubana, a Fidel y a Raúl por permitirme llegar aquí y seguir avanzando.

A la Universidad de las Ciencias Informáticas por todos los conocimientos y la preparación que me ha brindado.

A mis padres, familiares y amigos, por haberme apoyado y haber depositado su confianza en mí.

A mi novio Carlos por estar siempre ahí para brindarme su apoyo, por ser tan comprensivo y quererme tanto.

A todas las personas que he conocido acá en la universidad, los que se han ganado un lugarcito en mi corazón y con las que he compartido inolvidables momentos.

A mi tutor y a todos mis maestros y profesores que desde mis inicios han aportado un granito de arena a mi formación como estudiante.

A todos les estaré eternamente agradecida.

DEDICATORIA

A mis padres Marbelis y Juan Carlos por apoyarme siempre, darme ánimo en los momentos que más lo necesité, por ser guías y ejemplo, por confiar en mí y quererme tanto, los amo con todo mi corazón.

A mis hermanos Diley, Abel y Daril por apoyarme siempre, los quiero, adoro y admiro muchísimo.

A mis abuelitos Eneida y Carlos, por ser los mejores abuelos del mundo, los quiero mucho.

A mis segundos padres Monona, Margarita, Guilarte e Isnoel, por considerarme su hija, por ser tan lindos y quererme tanto, los adoro.

A Carlos por ser lo mejor y más lindo que me ha pasado, por respetarme y apoyarme siempre, por estar a mi lado en los momentos buenos y malos, por amarme tanto, te amo con todo mi corazón.

A mi familia querida, sobre todo a los que han estado siempre ahí para mí y me han demostrado su cariño, los quiero.

A todos mis amigos y compañeros de la universidad, con los que he podido compartir momentos inolvidables, los quiero y siempre estarán presentes aún en la distancia.

RESUMEN

El presente trabajo de diploma aborda básicamente cómo se obtuvieron las necesidades de los clientes y se transformaron en requisitos de software del módulo Reportes del proyecto SINAPSIS.

Dicho proyecto surge por la necesidad de llevar a cabo de forma automatizada, la gestión de presupuestos por proyectos en el sector público venezolano, por lo que este sistema facilitará el registro, seguimiento y control de los proyectos que forman parte del presupuesto anual de la nación. La gestión de presupuestos por proyectos es un proceso que se lleva a cabo en la administración pública de la República Bolivariana de Venezuela, como parte de la gestión de proyectos que se realiza todos los años en este país.

A partir del módulo antes mencionado, se obtendrá la información de la ejecución de los proyectos a través de reportes, de ahí la importancia de su automatización de manera efectiva.

Se tiene como objetivo general la aplicación de las etapas de elicitación, análisis, especificación y validación de requisitos, para transformar las necesidades de los clientes, en requisitos de software del módulo Reportes del proyecto SINAPSIS, facilitando de esta forma el entendimiento entre clientes y desarrolladores.

Como resultado de la aplicación de las etapas de la Ingeniería de Requisitos mencionadas, se obtuvieron como artefactos principales, la Especificación de requisitos de software y el Modelo de sistema.

Finalmente se validaron los artefactos obtenidos, en función de la calidad, lo cual se realizó a partir de la aplicación de métricas combinadas con un conjunto de listas de chequeo. La validación de los artefactos demostró que los mismos cumplen con los aspectos de calidad requeridos.

PALABRAS CLAVE

Ingeniería de Requisitos, reportes, software, artefactos, métricas, proyecto, listas de chequeo.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

TABLA DE CONTENIDO

INTRODUCCIÓN

En la actualidad, la industria del software es la cuna de la economía en gran parte del mundo, debido al gran desarrollo que ha alcanzado hasta el momento, constituyendo el software el motor impulsor que conduce a la toma de decisiones comerciales.

La República Bolivariana de Venezuela desde el año 2006, lleva a cabo la gestión del presupuesto por proyectos. A través de esta, el gobierno proporciona bienes y servicios para atender las necesidades de la comunidad, paga a proveedores y contratistas, al tiempo que resuelve problemas de pasivos laborales y contractuales con sus trabajadores.

Para llevar a cabo la gestión del presupuesto por proyectos, el Ministerio del Poder Popular para la Planificación y Desarrollo (MPPPD), conjuntamente con el Ministerio del Poder Popular de Economía y Finanzas (MPPEF), Petróleos de Venezuela S.A. (PDVSA) y la Vicepresidencia de la República (VPR), decidieron desarrollar un sistema informático que facilitara el registro, seguimiento y control de todos los proyectos que formarían parte del presupuesto anual de la nación, denominado *"Nueva Etapa*".

El sistema antes mencionado no fue desarrollado utilizando una metodología de desarrollo de software, por lo que no cuenta con la documentación necesaria que facilite la comprensión y optimización del mismo. A pesar de ser una aplicación Web, no está disponible para sus usuarios durante todo el año, por lo que se formulan y planifican los proyectos con estimaciones de costos y tiempos irreales.

Por otra parte, la obtención de información íntegra de la ejecución de los proyectos a través de reportes, se realiza de forma manual o semiautomática, es decir, se hace mediante vías no formales (Llamadas telefónicas, documentos, emails, etc.), lo cual provoca que el proceso sea lento e ineficiente, ya que no se realiza en tiempo real. Por todo lo anteriormente expuesto, *Nueva Etapa* no satisface todas las expectativas de los clientes y usuarios.

Dada la problemática planteada se decide desarrollar el proyecto SINAPSIS, un sistema informático que sustituirá completamente el anterior y que se divide en 7 módulos, entre los que se encuentra Reportes. Este módulo permitirá obtener la información íntegra de la ejecución de los proyectos a través de reportes, con el objetivo de apoyar la toma de decisiones, lo cual se realizará mediante la generación de reportes personalizados y estructurados por roles, en tiempo real. La automatización de este proceso de obtención

de información además de apoyar la toma de decisiones, posibilitará que el mismo en sí se realice rápidamente y de forma eficiente.

En correspondencia con lo abordado hasta el momento es válido resaltar, el hecho de que en general, existe una tendencia errónea en los informáticos, de plantear soluciones antes de comprender en profundidad los problemas o necesidades de los clientes, lo cual conlleva a que se desarrollen proyectos de manera ineficiente y que no cumplen todas las expectativas de los clientes y usuarios. Es por esto que del módulo Reportes del proyecto SINAPSIS se hace imprescindible obtener las necesidades de los clientes y usuarios, y lograr un acuerdo común y un mejor entendimiento entre clientes y desarrolladores, para contribuir de esta forma al desarrollo del mismo.

Sobre la base de los elementos expuestos hasta el momento se formula el siguiente **Problema**: ¿Cómo transformar las necesidades de los clientes, en requisitos de software del módulo Reportes del proyecto SINAPSIS, para facilitar el entendimiento entre clientes y desarrolladores? Siendo el **Objeto de Estudio** la Ingeniería de Requisitos cuyo **Campo de Acción** queda enmarcado en la Elicitación, Análisis, Especificación y Validación de requisitos de software.

Para dar solución al problema planteado, se define el siguiente **Objetivo General:** Realizar la Elicitación, Análisis, Especificación y Validación de requisitos, para transformar las necesidades de los clientes, en requisitos de software del módulo Reportes del proyecto SINAPSIS. Planteándose la siguiente **Idea a defender:** Si se realizan la Elicitación, Análisis, Especificación y Validación de requisitos, se podrán transformar las necesidades de los clientes, en requisitos de software del módulo Reportes del proyecto SINAPSIS, facilitando así el entendimiento entre clientes y desarrolladores.

Para dar cumplimiento al objetivo trazado, se establecen las siguientes **Tareas de Investigación:**

- Elaboración del marco teórico de la investigación a partir de la sistematización de las definiciones, etapas y metodologías de desarrollo del software en el marco de la Ingeniería de Requisitos del Módulo Reportes del proyecto SINAPSIS.

- Selección de los métodos, y técnicas a emplear, así como la descripción de la metodología empleada.

- Descripción de la solución propuesta a partir de los resultados obtenidos en la Elicitación, Análisis y Especificación de requisitos en el Módulo Reportes del proyecto SINAPSIS.

- Análisis de los resultados obtenidos a través de métricas y algunas técnicas propuestas por la Ingeniería de Requisitos.

Durante el transcurso del trabajo de diploma y para su desarrollo se emplearán varios **Métodos de Investigación,** para dar cumplimiento a las tareas planteadas anteriormente, los cuales se exponen a continuación.

Métodos teóricos.

Histórico - Lógico: Permitirá realizar el análisis de la trayectoria de la Ingeniería de Requisitos y para expresar teóricamente sus elementos fundamentales.

Análisis-Síntesis: Para el estudio del objeto de la investigación, que en este caso se refiere a la Ingeniería de requisitos, imposible de abordar sin la desintegración epistémica de cada una de sus partes, para luego realizar la conformación del todo.

Ascenso de lo abstracto a lo concreto: Parte de los fundamentos teóricos sobre la Ingeniería de requisitos y sus diversos componentes, hasta llegar a resultados de la praxis investigativa.

Modelación: Para el análisis y modelación del sistema ya que es necesario la elaboración de diagramas, figuras y otros artefactos importantes.

Métodos empíricos.

Entrevista: Permitirá realizar entrevistas no estandarizadas con el cliente para obtener la mayor cantidad de información posible, además ayudará a determinar las características de la propuesta de solución teniendo en cuenta las necesidades del cliente.

Estructura del trabajo de diploma.

El presente trabajo de diploma consta de 3 capítulos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el capítulo se hace un estudio de diferentes metodologías para el desarrollo de software, lenguajes de modelado y herramientas que pueden utilizarse para modelar los artefactos que proponen dichas metodologías. Además se abordan las etapas de la Ingeniería de Requisitos, las métricas para la especificación de requisitos y se presentan patrones de casos de uso.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

En el capítulo se muestra la solución que se propone con la aplicación de las etapas Elicitación, Análisis y Especificación de requisitos, al módulo Reportes del proyecto SINAPSIS. Se obtienen un conjunto de requisitos funcionales y no funcionales, para ello se aplicaron técnicas de ingenierías de requisitos. Se muestran además los actores y diagrama de casos de uso del sistema y la descripción de los mismos.

CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS

En el capítulo se lleva a cabo la etapa de validación de requisitos, con la cual se analizan los resultados obtenidos. Para ello se aplican un conjunto de métricas, además de listas de chequeo y una matriz de trazabilidad, que demuestran que los artefactos generados cuentan con la calidad requerida.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1- Introducción

Durante el proceso de desarrollo del software se aplican una serie de metodologías, patrones, métricas y se hace uso de las herramientas que permiten llevar a cabo el proceso. Es por esto que en el presente capítulo se realiza un estudio de la Ingeniería de Requisitos y sus etapas, así como de las metodologías de desarrollo de software y las herramientas que pueden utilizarse para modelar los artefactos que proponen dichas metodologías. Se presentan además algunos de los lenguajes de modelado, patrones de casos de uso, que son de importancia para lograr una correcta modelación de estos y las métricas que serán empleadas para la validación de los resultados obtenidos.

1.2- Ingeniería de Requisitos (IR). Generalidades

El desafío de la ingeniería del sistema y de los ingenieros del software es importante: ¿Cómo se puede asegurar que se ha especificado un sistema que recoge las necesidades del cliente y satisface sus expectativas? No hay una respuesta segura a esta difícil pregunta, pero un sólido proceso de Ingeniería de Requisitos es la mejor solución de que se dispone actualmente. (Pressman, 2002)

La ingeniería de requisitos es la disciplina de la ingeniería de software que comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requisitos de los inversores, que pueden entrar en conflicto entre ellos.

Sin embargo, no se puede iniciar un estudio del proceso si no se comprende primeramente qué es un Requisito.

Existen numerosas definiciones al respecto. Para una mejor comprensión se abordará la definición de la IEEE[1], considerando que es la más abarcadora y comprensible, de acuerdo con los estudios realizados sobre el tema.

Así, la IEEE 1233-1998 define requisito como:

I. Una condición o capacidad que necesita el usuario para resolver algún problema o alcanzar un objetivo.

[1] Institute of Electrical and Electronics Engineers.

II. Condición o capacidad que debe cumplir o poseer un sistema o componente del sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto.

III. Una representación documental de una condición o capacidad como en I o en II (IEEE, 1998).

Resulta válido entonces exponer las definiciones de Ingeniería de Requisitos estudiadas de los autores Donald Reifer y Sommerville:

El primero describe este proceso de la siguiente forma:
"La IR es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un costo reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario." (Pressman, 2002)

Por su parte Sommerville refiere que:
"La ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretender comunicar las necesidades del sistema del cliente a los desarrolladores del sistema". (Sommerville, 2005)

De las dos definiciones anteriormente expuestas, se considera, coincidiendo con el criterio de otros especialistas de la comunidad científica estudiosos del tema, que la primera es la más completa, ya que según los estudios realizados se aborda prácticamente todo lo relacionado con el proceso en sí.

A la IR se le confiere gran importancia, ya que facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional.

Esta disciplina cumple un papel primordial en el proceso de construcción y producción de un software, es decir, que está basado en función de las necesidades planteadas por los clientes en un nivel muy general, donde se descubren, documentan, analizan y definen los servicios o componentes de lo que se desea producir, además de las restricciones que debe tener el producto o software.

Los diferentes autores estudiados coinciden en que su principal tarea consiste en la definición del proceso a seguir en la construcción de un software, y de facilitar la comprensión de lo que el cliente requiera. La obtención correcta de los requisitos puede llegar a describir con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento de un sistema, ya que para que un proyecto de desarrollo de software pueda tener éxito es de vital importancia realizar una comprensión total de los requisitos del software a diseñar.

De tal manera que, basarse en la obtención de requisitos y sobre todo que sean correctos, contribuye en gran medida a minimizar los problemas asociados al desarrollo del software, con lo que se logra reducción de tiempo en la construcción, reducción de errores, y los más importante, no sólo para el cliente, sino también para el desarrollador, evita gastar más dinero que el planeado en el proyecto inicialmente.

1.2.1- Etapas de la Ingeniería de Requisitos.

Dorfman y Thayer plantean que la Ingeniería de Requisitos incluye tareas de elicitación, análisis, especificación, validación y administración de requisitos de software, considerando la “administración de requisitos de software” como la planificación y control de todas esas actividades relacionadas (Dorfman M. y Thayer, 1997).

Según Wiegers la disciplina de Ingeniería de Requisitos se divide en Desarrollo de Requisitos y Gestión de Requisitos. El Desarrollo de Requisitos incluye la elicitación, el análisis, la especificación y la validación de los requisitos del proyecto de software (Wiegers, 2003).

Sin embargo, existen autores como Sommerville, que consideran que el proceso de Ingeniería de Requisitos puede ser descrito en seis pasos distintos: Identificación de Requisitos, Análisis de Requisitos y Negociación, Especificación de Requisitos, Modelado del Sistema, Validación de Requisitos y Gestión de Requisitos (Sommerville, 2005).

Luego de realizado un estudio sobre el tema, se ha podido apreciar que varios autores resumen las etapas de la IR en: elicitación, análisis, especificación, validación y gestión de requisitos. En el trabajo serán desarrolladas solo las cuatro primeras, por ser las que se consideran que involucran actividades de reunir, documentar y evaluar los requisitos. Además de que se decidió desarrollar la gestión de requisitos una vez obtenida la línea base de los requisitos del sistema y entregada la primera versión del producto, puesto que es el momento en el que se pueden dar nuevos cambios en los requisitos debido a la lejanía de los clientes que impide un constante intercambio con los desarrolladores.

A continuación se exponen las definiciones de cada una de estas etapas.

1. **Elicitación o Identificación de requisitos.**

Constituye en síntesis la obtención de requisitos. En correspondencia con los criterios de Pressman “En principio, parece bastante simple preguntar al cliente, a los usuarios y a los que están involucrados en los objetivos del sistema o producto y sean expertos, investigar cómo los sistemas o productos se ajustan a las necesidades del negocio, y finalmente, cómo el sistema o producto va a ser utilizado en el día a día. Esto que parece simple, es muy complicado” (Pressman, 2002).

Existen una serie de problemas que ayudan a comprender por qué la elicitación de requisitos es costosa, estos son:

- **Problemas de alcance:** El límite del sistema está mal definido o los detalles técnicos innecesarios, que han sido aportados por los clientes/usuarios, pueden confundir más que clarificar los objetivos del sistema.

- **Problemas de comprensión:** Los clientes/usuarios no están completamente seguros de lo que necesitan, tienen una pobre compresión de las capacidades y limitaciones de su entorno de computación, no existe un total entendimiento del dominio del problema, existen dificultades para comunicar las necesidades al ingeniero del sistema, la omisión de información por considerar que es obvia, especificación de requisitos que están en conflicto con las necesidades de otros clientes/usuarios, o especificar requisitos ambiguos o poco estables.

- **Problemas de volatilidad:** Los requisitos cambian con el tiempo (Christel, et al., 1992).

Asimismo, se sugieren un conjunto de actuaciones para esta etapa, que están descritos en las tareas siguientes:

- Valorar el impacto en el negocio y la viabilidad técnica del sistema propuesto.

- Identificar las personas que ayudarán a especificar requisitos y contrastar su papel en la organización.

- Definir el entorno técnico (arquitectura de computación, sistema operativo, necesidades de telecomunicaciones) en el sistema o producto a desarrollar e integrar.

- Identificar restricciones de dominio (características específicas del entorno de negocio en el dominio de la aplicación) que limiten la funcionalidad y rendimientos del sistema o producto a construir.
- Definir uno o más métodos de obtención de requisitos (entrevistas, grupos de trabajo, equipos de discusión).
- Solicitar la participación de muchas personas para que los requisitos se definan desde diferentes puntos de vista, asegurarse de identificar lo fundamental de cada requisito registrado.
- Identificar requisitos ambiguos como candidatos para el prototipado.
- Crear escenarios de uso para ayudar a los clientes/usuarios a identificar mejor los requisitos fundamentales (Sommerville, 2005).

Cada uno de los productos obtenidos debe ser revisado por las personas que hayan participado en la obtención de sus requisitos.

El proceso de elicitación/identificación de requisitos puede resultar complejo y costoso debido a numerosos problemas. Muchas veces los clientes no están seguros de lo que realmente necesitan, se les hace muy difícil comunicarle al analista sus necesidades, omiten información suponiendo que es obvia o hacen una mala definición del límite del sistema.

¿Cómo hacer entonces para garantizar la calidad de este proceso? Por su complejidad, la Ingeniería de Requisitos propone técnicas o métodos que permiten hacerlo de una forma más eficiente y precisa.

A continuación se exponen algunas de ellas:

- **Entrevistas.**

Es una de las técnicas más usadas en la captura de requisitos. Consiste en establecer una conversación entre personas de ambas partes. Las entrevistas son dirigidas normalmente por el personal más experto del equipo, quienes junto al equipo de profesionales de otras áreas, como la psicología y el derecho, son los encargados de orientar las entrevistas de forma que la información obtenida a través de ellas sea relevante al proceso. Al analizar las características del sistema con el personal seleccionado cuidadosamente por sus conocimientos sobre este, los analistas pueden conocer los datos que no están

disponibles en ninguna otra forma. En esta técnica se pueden identificar tres fases: la preparación, la realización y el análisis de la información obtenida.

- **Tormenta de ideas (Brainstorming).**

El propósito de esta técnica es la generación de ideas en un ambiente libre de críticas o juicios. La intención de su aplicación es la de generar la mayor cantidad posible de requisitos para el sistema. Esta técnica tiene la ventaja de que es muy fácil de aprender y requiere poca organización.
Por otro lado, al ser un proceso poco estructurado, puede no producir resultados con la misma calidad o nivel de detalle que otras técnicas.

- **Prototipos**

Durante la actividad de extracción de requisitos, puede ocurrir que algunos requisitos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requisitos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final (Durán, et al., 2000) (Escalona, et al., 2002).

Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, se construyen para ayudar a los desarrolladores, usuarios y clientes en la obtención de un mejor entendimiento del sistema, en especial de los requisitos que están menos claros. Existe un fuerte solapamiento de su uso para la elicitación de requisitos respecto a la validación de requisitos. El prototipado es un medio común para la validación de la interpretación de los ingenieros de software de los requisitos de software, así como también para la elicitación de los mismos". (Adriano, 2006)

2. Análisis de Requisitos.

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

Al iniciarse la actividad del análisis de requisitos se plantean las siguientes cuestiones:

- ¿Cada requisito es consistente con los objetivos generales del sistema/producto?
- ¿Tienen todos los requisitos especificados el nivel adecuado de abstracción? Es decir, ¿algunos requisitos tienen un nivel de detalle técnico inapropiado en esta etapa?

- ¿El requisito es necesario o representa una característica añadida que puede no ser esencial a la finalidad del sistema?
- ¿Cada requisito está delimitado y sin ambigüedad?
- ¿Cada requisito tiene procedencia. Es decir, ¿existe un origen (general o específico) conocido para cada requisito?
- ¿Existen requisitos incompatibles con otros requisitos?
- ¿Es posible lograr cada requisito en el entorno técnico donde se integrará el sistema o producto?
- ¿Se puede probar el requisito una vez implementado?

Es corriente en clientes y usuarios solicitar más de lo que puede realizarse, consumiendo recursos de negocio limitados. También es relativamente común en clientes y usuarios el proponer requisitos contradictorios, argumentando que su versión es esencial por necesidades especiales.

El ingeniero del sistema debe resolver estos conflictos a través de un proceso de negociación. Los clientes, usuarios y el resto de intervinientes deberán clasificar sus requisitos y discutir los posibles conflictos según su prioridad. Los riesgos asociados con cada requisito serán identificados y analizados. Se efectúan estimaciones del esfuerzo de desarrollo que se utilizan para valorar el impacto de cada requisito en el coste del proyecto y en el plazo de entrega. Utilizando un procedimiento iterativo, se irán eliminando requisitos, se irán combinando y/o modificando para conseguir satisfacer los objetivos planteados (Pressman, 2002).

En la etapa de análisis de requisitos se trata, precisamente, de analizar la información recibida desde los usuarios, para distinguir las necesidades de tareas, los requisitos funcionales, atributos de calidad, soluciones sugeridas e información extraña (Wiegers, 2006.).

Se define a la etapa de análisis de requisitos como la actividad de transformar requisitos informales en requisitos técnicos mediante el aseguramiento de que los mismos reflejan los atributos de calidad que necesitan y que expresan las necesidades de los clientes. El análisis es una actividad reiterativa. Los pasos del proceso deberán ser repetidos una cierta cantidad de veces, existe una constante comunicación entre la consulta de los clientes, los usuarios finales y los desarrolladores (Young, 2004).

Algunas técnicas para esta etapa son:

- **Glosario y ontologías**: La diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Esta necesidad se vuelve más patente en los sistemas de información web puesto que el equipo de desarrollo en ellas suele ser más interdisciplinario.

- **Plantillas o patrones**: Esta técnica, recomendada por varios autores, tiene como objetivo describir los requisitos mediante el lenguaje natural pero de una manera estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando, usando para ello el lenguaje del usuario.

- **Escenarios:** La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo.

- **Lenguajes Formales:** Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista (Escalona, et al., 2002).

3. Especificación de Requisitos.

En el contexto de un sistema basado en computadoras y software, el término especificación significa distintas cosas para diferentes personas. Estudiosos del tema sugieren que debe desarrollarse una plantilla estándar y usarse en la especificación del sistema, argumentando que así se conseguirán requisitos que sean presentados de una forma más consistente y más comprensible. No obstante, en muchas ocasiones es necesario buscar la flexibilidad cuando una especificación va a ser desarrollada. Para grandes sistemas, un documento escrito, combinado con descripciones en lenguaje natural y modelos gráficos puede ser la mejor alternativa. En cualquier caso, los escenarios a utilizar pueden ser tanto los requeridos para productos de tamaño pequeño o los de sistemas que residan en entornos técnicos bien conocidos.

La Especificación del Sistema es el producto final sobre los requisitos del sistema obtenido por el ingeniero. Sirve como fundamento para la ingeniería del hardware, ingeniería del software, la ingeniería de bases de datos y la ingeniería humana. Describe la función y características de un sistema de computación y las restricciones que gobiernan su desarrollo. Esta delimita cada elemento del sistema. Además describe la información (datos y control) que entra y sale de este (Pressman, 2002).

Los clientes y usuarios utilizan la Especificación de Requisitos para comparar si lo que se está proponiendo, coincide con las necesidades de la empresa. Los analistas y programadores la utilizan para determinar el producto que debe desarrollarse. El personal de pruebas elaborará las pruebas funcionales y de sistemas en base a este documento. Para el administrador del proyecto sirve como referencia y control de la evolución del sistema (Sanchez, 2002).

4. Validación de Requisitos.

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto.

El primer mecanismo para la validación de los requisitos es la revisión técnica formal. El equipo de revisión incluye ingenieros del sistema, clientes, usuarios, y otros intervinientes que examinan la especificación del sistema buscando errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información incompleta, inconsistencias (es un problema importante), requisitos contradictorios, o requisitos imposibles o inalcanzables.

Aunque la validación de requisitos puede guiarse de manera que se descubran errores, es útil chequear cada requisito con un cuestionario. Las siguientes cuestiones son las que propone Pressman de las preguntas que pueden plantearse:

- ¿Está el requisito claramente definido? ¿Puede interpretarse mal?
- ¿Está identificado el origen del requisito (por ejemplo: persona, norma, documento)? ¿El planteamiento final del requisito ha sido contrastado con la fuente original?
- ¿El requisito está delimitado en términos cuantitativos?
- ¿Qué otros requisitos hacen referencia al requisito estudiado? ¿Están claramente identificados por medio de una matriz de referencias cruzadas o por cualquier otro mecanismo?

- ¿El requisito incumple alguna restricción definida?
- ¿El requisito es verificable? Si es así, ¿se pueden efectuar pruebas (algunas veces llamadas criterios de validación) para verificar el requisito?
- ¿Se puede seguir el requisito en el Modelo de sistema que se ha desarrollado?
- ¿Se puede localizar el requisito en el conjunto de objetivos del sistema/producto?
- ¿Está el requisito asociado con los rendimientos del sistema o con su comportamiento y han sido establecidas claramente sus características operacionales? ¿El requisito está implícitamente definido?

Las preguntas planteadas en la lista de comprobación ayudan a asegurar que el equipo de validación dispone de lo necesario para realizar una revisión completa de cada requisito (Pressman, 2002).

Algunas técnicas que pueden aplicarse en esta etapa 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.

- **Auditorías**: La revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.

- **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo (Durán, et al., 2000). Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

- **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. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (Escalona, et al., 2002).

- **Listas de chequeo:** Permite mediante una serie de preguntas, donde se comprueba la presencia de información en los documentos, detectar puntos a los cuales no se les haya dado solución. (Báez, et al., 2001)

Por último, se hace referencia a la gestión de requisitos que se expone brevemente para la comprensión cabal de las etapas, pero que como ya se ha explicado, no forma parte del campo de acción del presente trabajo.

5. Gestión de requisitos.

Los requisitos del sistema cambian y el deseo de cambiarlos persiste a lo largo de la vida del sistema. 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. Muchas de estas actividades son idénticas a las técnicas de gestión de configuración del software.

El control de los cambios de los requisitos es vital para el buen desarrollo del sistema, por lo que es recomendable además de las actividades expuestas anteriormente para la gestión de los requisitos, tener en cuenta otras de las actividades pertenecientes a la Gestión de Configuración del Software (GCS), que son de gran utilidad para la gestión de los requisitos. (Pressman, 2002)

A pesar de su importancia y utilidad, teniendo en cuenta los objetivos trazados, solamente se llevarán a cabo las etapas de elicitación/identificación, análisis, especificación y validación de requisitos. Para llevar a cabo estas etapas de la IR, se hace necesario seguir una guía que permita planearlas y controlarlas, por lo que es fundamental realizar el estudio de las Metodologías de desarrollo de software.

1.3- Metodologías de desarrollo de software

El desarrollo de software es sin dudas una tarea bastante difícil por todo lo que esta conlleva. Para darle solución a este problema ha surgido una alternativa desde hace mucho: la Metodología. Las metodologías imponen un proceso disciplinado que permite estructurar, planear y controlar el proceso de desarrollo de software con el fin de hacerlo más predecible y eficiente.

Hoy en día existen numerosas propuestas de metodologías que inciden en distintas dimensiones del proceso de desarrollo. Un ejemplo de ellas son las metodologías robustas, centradas específicamente en el control del proceso. Estas han demostrado ser efectivas y necesarias en un gran número de proyectos, sobre todo aquellos proyectos de gran tamaño respecto a tiempo y recursos.

Por otra parte están las metodologías ágiles, que se caracterizan por tener un desarrollo incremental para producir tempranamente pequeñas entregas en ciclos rápidos, y predisposición para el cambio y la adaptación continua. Estas metodologías por lo general se centran en desarrollar productos funcionales más que en conseguir una buena documentación (Figueroa, et al.).

Dentro de las metodologías ágiles se conocen XP (Extreme Programming), SCRUM, FDD (Feature Driven Development) y AUP (Agile Unified Process), entre muchas otras. Esta propuesta de metodología no será utilizada debido a que no se ajusta a los criterios de selección del proyecto SINAPSIS, ya que, este es un proyecto de gran tamaño respecto a tiempo y recursos y se requiere de una documentación detallada del mismo.

Por lo antes planteado se empleará una metodología robusta. A continuación se describen las metodologías Rational Unified Process (RUP) y Microsoft Solution Framework (MSF), que se incluyen dentro de esta clasificación.

1.3.1- Rational Unified Process (RUP).

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP como proceso, en su modelación define como sus principales elementos:

- **Trabajadores (“quién”):** Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

- **Actividades (“cómo”):** Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

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

- **Flujo de actividades ("Cuándo"):** Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable. (Jacobson, et al., 2000)

La metodología RUP propone cuatro fases y nueve flujos de trabajo que se muestran en la Figura 1:

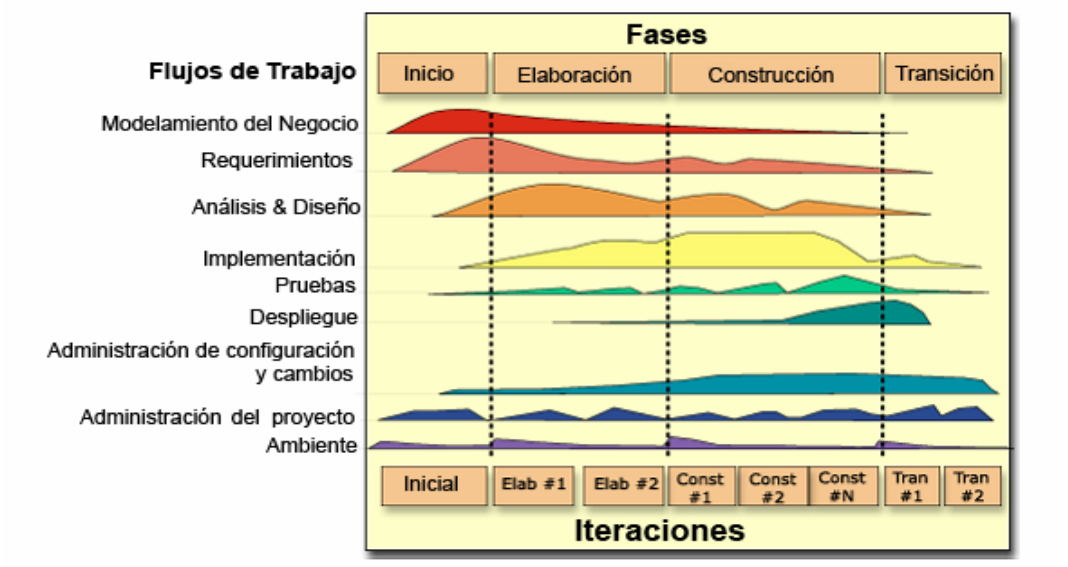


Figura 1. Metodología RUP

El proceso de desarrollo se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo está compuesto por las cuatro fases que se muestran en la Figura 1, el resultado final de un ciclo es una versión del sistema. En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas se llevarán a cabo los flujos de trabajo (Figura 1).

El ciclo de vida de RUP se caracteriza por:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos, a partir de aquí los casos de uso guían el proceso de desarrollo.

- **Centrado en la Arquitectura:** La arquitectura muestra la visión común del sistema completo, describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo

económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU[2] relevantes desde el punto de vista de la arquitectura.

- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o mini-proyectos. Cada mini-proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. (Jacobson, et al., 2000)

Roles que se definen dentro de la metodología RUP.

RUP define roles, que están agrupados por la participación en actividades relacionadas. Estos grupos son:

- Analistas.
- Desarrolladores.
- Gestores.
- Apoyo.
- Especialistas en pruebas.
- Otros roles: stakeholders, revisor, coordinación de revisiones y revisor técnico (Rational Software Corporation, 2003).

1.3.2- Microsoft Solution Framework (MSF)

Microsoft Solutions Framework es un marco de trabajo para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft. Es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. (Microsoft Corporation, 2010)

Algunas características de la metodología son:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

[2] Casos de uso

- **Escalable:** puede organizar equipos tan pequeños entre tres o cuatro personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica**: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología (Sanchez, 2002).

Fases

- Fase Visión y Alcances.
- Fase de Planificación.
- Fase de Desarrollo.
- Fase de Estabilización.
- Fase de Distribución (López Requena, 2006) (Figueroa, et al.).

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación (Sanchez, 2002).

Luego de haberse realizado un estudio de las metodologías RUP y MSF expuestas anteriormente, se tomó la decisión de adoptar RUP como proceso rector de desarrollo para guiar la solución que se aplicará al módulo Reportes del proyecto SINAPSIS, en primer lugar por ser apropiado para proyectos complejos y de larga duración como lo es este, en segundo lugar por tener el equipo de desarrollo del proyecto experiencia en su aplicación, obteniendo resultados positivos con su uso, y en tercer lugar por contar con clientes que no tendrán una relación directa con el equipo de desarrollo. Además, constituye uno de los estándares internacionales que más aceptación ha tenido en los últimos años en el desarrollo informático.

Por otra parte existe un gran inconveniente con la metodología MSF, ya que, es un marco de trabajo para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft, es decir, que esta solo usa herramientas y tecnologías de dicha empresa, lo cual no favorece el trabajo que se quiere realizar, si se tiene en cuenta la necesidad que existe en estos momentos en la universidad de migrar a software libre.

1.4- Lenguajes de Modelado.

En la actualidad, no es posible imaginarse el desarrollo de software sin antes pasar por una etapa de modelado. La utilización de modelos es una metodología aceptada y recomendada tanto en el ámbito académico como dentro del ámbito profesional privado.

Un modelo es una representación, en cierto medio, de algo en el mismo u otro medio, que capta los aspectos esenciales de lo que se está modelando, desde cierto punto de vista, y simplifica u omite el resto. Provee una base común de entendimiento para que todas las partes implicadas puedan comunicarse y estar hablando de las mismas cosas, refiriéndose a los mismos conceptos.
Actualmente, los modelos realizados con el fin de representar sistemas de software son creados a través de lenguajes de modelado. Del mismo modo que los sistemas se desarrollan a través lenguajes de programación.

Algunos de estos lenguajes de modelado son: la Notación de Modelado de Procesos de Negocio (BPMN), el Lenguaje de Modelado de Sistemas (SysML) y el Lenguaje Unificado de Modelado, conocido como UML por sus siglas en inglés. Estos se exponen a continuación.

1.4.1- Notación de Modelado de Procesos de Negocio (BPMN).

BPMN es un estándar de notación desarrollado por Business Process Management Iniciative (BPMI). Esta especificación, representa más de dos años de esfuerzo de BPMI Notation Working Group. Actualmente es mantenida por el Object Management Group (OMG) desde que ambas organizaciones se unieron en el 2005. La versión más actual de BPMN es la 1.2 y un proceso de mayor revisión para BPMN 2.0 está en progreso.

El objetivo principal de BPMN es proveer una notación que sea de fácil comprensión para todos los usuarios del negocio, para los analistas que son los que crean los primeros modelos, para los desarrolladores técnicos, responsables de implementar la tecnología que realizará dichos procesos, y finalmente, para las personas del negocio, los cuales gestionan y monitorean los procesos. Esta notación, crea un puente para la normalización de las diferencias entre el diseño de los procesos de negocio y el proceso de implementación (OMG, 2009).

1.4.2- Lenguaje de Modelado de Sistemas (SysML).

El Lenguaje de Modelado de Sistemas (SysML, System Modeling Language) es un nuevo lenguaje de modelado de propósito general para la ingeniería de sistemas. Soporta la especificación, análisis, diseño, verificación y validación de un amplio rango de complejos sistemas; que pueden incluir hardware, software, información, procesos, personal y facilidades. Puede ser adaptado para modelar aplicaciones de un dominio específico, como automovilísticas, aeroespaciales, comunicaciones y sistemas de información. (Milestone Consulting)

Es efectivo particularmente al especificar requisitos, estructura, comportamiento y restricciones de las propiedades del sistema. El lenguaje está orientado a soportar múltiples procesos y enfoques como el estructurado, orientado a objetos y otros, pero cada metodología puede imponer restricciones adicionales en como un tipo de artefacto o diagrama debe ser usado.

SysML reutiliza un subconjunto de UML 2.0, manteniendo intactos algunos diagramas y redefiniendo otros, a su vez que incorpora nuevos diagramas. Entre los más importantes se encuentra el diagrama de requisitos, en el cual se pueden representar los requisitos en formato gráfico, tabular o estructura de árbol. (Giraldo, et al., 2009)

1.4.3- Lenguaje Unificado de Modelado (UML).

UML ("Unified Modeling Language") es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de los sistemas de software. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios, así como en herramientas interactivas de modelado visual que tengan generadores de código y generadores de informes. Es el lenguaje de modelado que define RUP. Ayuda al usuario a entender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el coste y el tiempo empleado en la construcción de las piezas que constituirán el modelo (Jacobson, et al., 2000). Fue originalmente concebido por tres de los más prominentes metodologistas en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh e Ivar Jacobson (International, 2008).

Hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mientras más complejo es el sistema que se desea crear más beneficios presenta el uso de UML, las razones de esto son evidentes:

- Posibilita la descripción de sistemas, simplificando la complejidad de estos y sin pérdida de información, haciendo posible la comprensión del sistema tanto para usuarios como desarrolladores.
- Ofrece varios diagramas para el modelado de sistemas.
- Brinda facilidades para el diseño, documentación, reutilización de código y detecciones de fallas.
- Facilita la comunicación entre desarrolladores, permite ahorrar tiempo en el desarrollo del software y hace más sencillas las modificaciones que se vayan a realizar (Tarazona, et al., 2006).

Luego de estudiados los lenguajes de modelado expuestos anteriormente, se tomó la decisión de utilizar UML para el modelado del sistema del módulo Reportes del proyecto SINAPSIS, ya que, es de fácil comprensión para su aplicación por parte del equipo de desarrollo, con su uso resulta más fácil encontrar dificultades y dependencias en los sistemas y menos engorroso realizar los cambios una vez que se haya comenzado a desarrollar el software. Además facilita la comunicación entre los desarrolladores, por lo que permite el ahorro de tiempo en el desarrollo del software.

En los días de hoy, los lenguajes de modelado tienen una estrecha relación con las herramientas de desarrollo de software, por lo que no tiene sentido realizar el estudio de un tema sin estudiar el otro.

1.5- Herramientas de desarrollo de software.

Las Herramientas CASE (Computer Aided Software Engineering, del español Ingeniería de Software Asistida por Computadoras) son aplicaciones de apoyo al desarrollo, mantenimiento y documentación automatizados de software. Permiten aplicar la metodología de análisis y diseño orientados a objetos. Cuanto más grande es un proyecto, más importante es utilizar una herramienta CASE, ya que, estas contribuyen a aumentar la productividad en el desarrollo de software reduciendo el coste en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software. Al hacer uso de estas, los analistas pueden representar los requisitos del sistema con un modelo de casos de uso (Paradigm, 2007).

Algunas de las herramientas CASE más usadas en el mundo para el desarrollo de software son Rational Rose, Enterprise Architect y Visual Paradigm for UML, las cuales se describen a continuación.

1.5.1- Rational Rose Enterprise Edition

Rational Rose Enterprise Edition es una de las herramienta CASE más utilizadas en la actualidad. Esta permite el diseño detallado del software y la generación de código fuente (de programas y bases de datos) e ingeniería inversa (obtención del diseño a partir del código fuente), basado en modelos con soporte UML, promueve el trabajo en equipo y el desarrollo iterativo. Es una forma de ayuda para la comprensión del sistema y de sus distintos componentes.

Otras características que presenta la herramienta son:

- Software propietario.
- Capacidad de crear definiciones de tipo documento XML para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad de análisis de calidad de código.
- Sistemas operativos apropiados: Windows 2000, Windows NT, Windows XP (GSInnova, 2007).

1.5.2- Enterprise Architect (EA)

Enterprise Architect es una herramienta CASE para el diseño y construcción de sistemas de software. EA soporta la especificación de UML 2.0. Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio.

Esta plataforma proporciona al ingeniero de software, concretamente al ingeniero de requisitos, un marco de trabajo dentro del cual puede definir, almacenar y analizar los requisitos, objetivos, tareas, términos y demás artefactos que vaya recopilando durante su labor dentro de la fase de elicitación y análisis de proyectos software. Además, facilita la generación de reportes o informes, modelado y diseño de artefactos UML y transformación de dichos artefactos en plantillas de código fuente en múltiples lenguajes de programación, Java, C#, PHP, etc. (Systems)

Algunas de las características y funcionalidades claves de Enterprise Architect son:

- Crea elementos del modelo UML para un amplio alcance de objetivos.
- Ubica esos elementos en diagramas y paquetes.
- Crea conectores entre elementos.

- Documenta los elementos que ha creado.
- Documentación flexible y comprensible.
- Ingeniería de código directa e inversa.
- Puede generar código fuente C++, Java, C#, Visual Basic, Delphi, entre otros.
- Plugins para vincular EA a Visual Studio.NET o Eclipse.
- Soporta control de versiones de repositorios.
- Software propietario (INGENIA, 2006) (MasterMagazine, 2004).

1.5.3- Visual Paradigm for UML

Visual Paradigm for UML es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste**.** (Systems, 2000)

Está disponible en varias ediciones, cada una destinada a determinadas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal.

Características:

- Licencia: Gratuita y Comercial.
- Producto de calidad.
- Soporta aplicaciones Web.
- Varios idiomas.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Entorno gráfico amigable para el usuario.
- Disponibilidad en múltiples plataformas (Windows/Linux/Mac OS X).

Algunas de las funcionalidades de Visual Paradigm for UML son:

- Soporta un conjunto de lenguajes en la generación de código e ingeniería inversa, tales como: Java, C++, PHP, XML Schema, entre otros.
- Soporta un conjunto de estándares entre los que se encuentra UML, SysML, BPMN, XML y XMI.
- Soporte de Modelado UML, modelado de procesos de negocios y un generador de mapeo de objetos relacionales para los lenguajes de programación Java, .NET y PHP.
- Permite la integración con Eclipse, NetBeans, Sun ONE, JBuilder, IntelliJ IDEA, Microsoft Visual Studio y otros.
- Modelado colaborativo con Subversion.
- Diseño de prototipo de Interfaz de Usuario. Permite insertar información adicional a los diagramas mediante notas y comentarios para describir sus elementos lo que facilita la revisión de los prototipos así como el trabajo en equipo.
- Diagramas de flujo de datos Soporte ORM - Generación de objetos Java desde la base de datos.
- Modelado de base de datos. Proporciona una mayor documentación de la base de datos y diagramas de mapeo de relación de objetos.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Esquema de la exportación como imagen (JPG, PNG, entre otros)
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Patrones de Diseño de apoyo.
- Importa los dibujos de Visio en Visual Paradigm.
- Elemento basado en el historial de revisiones.
- Reducción del consumo de recursos.
- Interoperabilidad e integración. Permite la integración con un conjunto de herramientas (Visio drawing, Rational Rose, ERwin Data Modeler Project, Microsoft Excel, Microsoft Word document), intercambiando diagramas UML y modelos, usando representaciones industriales comunes.

- Ofrece herramientas para la generación de reportes en formatos html, pdf y doc (Systems, 2000) (International, 2008).

Luego de haberse realizado un estudio de las características de las herramientas de desarrollo de software expuestas anteriormente, se tomó la decisión de utilizar Visual Paradigm for UML como herramienta para el modelado del sistema del módulo Reportes del proyecto SINAPSIS, por su nivel de integración con varios entorno de desarrollo, entre ellos Eclipse, que será usado para el desarrollo del módulo antes mencionado, por la posibilidad de trabajar ambos sobre la plataforma libre GNU/Linux, sistema operativo sobre el cual se desarrolla el proyecto antes mencionado, por ser multiplataforma, por tener el equipo de desarrollo del proyecto experiencia en su uso, además de que la universidad posee la licencia para el empleo de la herramienta en sus proyectos y una versión nativa para Linux. Características con las cuales no cuenta ninguna de las restantes herramientas estudiadas.

1.6- Herramientas de modelado de Prototipos de interfaz de usuario.

Los prototipos de interfaz de usuario son de gran utilidad en el desarrollo de software, ya que con estos se pueden lograr simulaciones del sistema a construir con un alto grado de detalle, permiten entender los requisitos que no han quedado bien claros en la captura, además de que brindan la posibilidad de validar los requisitos que se hayan propuesto del producto antes de que se inicie su desarrollo. Permite fijar presentaciones en las que se puede solicitar la aprobación de la maqueta del producto a los interlocutores del cliente o del equipo de desarrollo, así como: especificar el contenido, planificar las fases del desarrollo o detectar errores de diseño antes de iniciar el proceso de producción.

Algunas de las herramientas que permiten realizar el modelado de prototipos son: Microsoft Office Visio, Axure RP y Visual Paradigm for UML, entre otras.

1.6.1- Microsoft Office Visio.

Microsoft Office Visio facilita a los profesionales empresariales y de TI la visualización, análisis y comunicación de información, sistemas y procesos complejos. Mediante los diagramas de aspecto profesional de Microsoft Office Visio, se puede mejorar la comprensión de sistemas y procesos, entender mejor la información compleja y utilizar esos conocimientos para tomar mejores decisiones para la empresa. Además de lo anteriormente dicho, la herramienta brinda la posibilidad de confeccionar prototipos de interfaz de usuario de forma fácil y que pueden ser muy agradables para el cliente.

Algunas características de la herramienta son:

- Proporciona una plantilla “Interfaz de usuario de Windows XP” que permite la creación de prototipos con formas diseñadas según las directrices de Microsoft Windows XP.
- Incluye 2 paletas que son las que básicamente permiten la confección de prototipos de interfaz de usuario: “Windows and Dialogs (US units)” y “Common Controls (US units)”, las cuales están compuestas por varios componentes, entre ellos paneles, cajas de texto y botones, que facilitan la modelación de los prototipos.
- Sistemas operativos apropiados: Microsoft Windows XP y Windows Server 2003.
- Permite el acceso rápido a las plantillas que se utilizan con frecuencia.
- Permite crear diagramas más inteligentes vinculándolos a datos para proporcionar una imagen más completa de un proyecto, proceso o sistema, permitiendo la integración con otras aplicaciones de Microsoft office como Microsoft Office Excel 2007 y Microsoft Office Access 2007.
- Permite guardar diagramas en formato de archivo PDF o XPS para facilitar su portabilidad y llegar a un mayor número de destinatarios (Microsoft Corporation, 2010).

1.6.2- Axure RP.

Axure RP es una aplicación eficiente en la creación de prototipos y especificaciones muy precisas para páginas web. Esta herramienta está especializada en la tarea, así que cuenta con todo lo que se puede necesitar para crear los prototipos de forma más eficiente. Esta permite componer la página web visualmente, añadiendo, quitando y modificando los elementos con suma facilidad.

Donde Axure RP demuestra su grado de especialización es en las anotaciones. En este punto, permite especificar el estado de cada elemento (Propuesto, Aceptado, Incorporado), el beneficio esperado (Crítico, Importante, Útil), el riesgo, la estabilidad, a quién va dirigido y a quién se le asignará la tarea.
Para el uso de Axure RP se necesita de los sistemas operativos:
Win98/98SE/Me/2000/XP (Gómez, 2010).

1.6.3- Visual Paradigm for UML.

La herramienta además de que permite llevar a cabo el ciclo completo del desarrollo de software, brinda la posibilidad de confeccionar prototipos de interfaz de usuario, los cuales pueden ser muy agradables para

el cliente y se pueden realizar fácilmente con el apoyo de todos los componentes que contiene. Cuenta con todo lo que se puede necesitar para crear los prototipos de forma eficiente. Permite insertar información adicional a los diagramas mediante notas y comentarios para describir sus elementos lo que facilita la revisión de los prototipos así como el trabajo en equipo. Es importante resaltar su disponibilidad en múltiples plataformas como son: Windows, Linux y Mac OS X. (International, 2008)

Para el modelado de los prototipos de interfaz de usuario del módulo Reportes del proyecto SINAPSIS se hará uso de la herramienta Visual Paradigm for UML, lo cual posibilita que se trabaje con una sola herramienta tanto para el modelado del sistema como para el modelado de prototipos de interfaz de usuario, además esta es multiplataforma, lo cual es una ventaja si se tiene en cuenta la necesidad que existe en estos momentos en la universidad de migrar a software libre.

1.7- Patrones de Casos de Uso.

Un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, estandariza buenos principios y da sugerencias sobre la manera de usarlo en nuevas situaciones, sobre todo en la asignación de responsabilidades. Es decir, es una solución común a un problema común de un determinado contexto. (Jacobson, et al., 2000)

Los patrones de casos de uso modelan reglas que hacen más comprensibles y correctos los modelos. Algunos de estos patrones son los que se muestran a continuación.

Concordancia (Commonality).

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

- Reutilización**.**

Este patrón está constituido por tres 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.

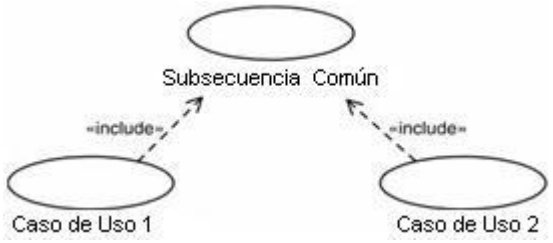


Figura 2. Reutilización

- 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.

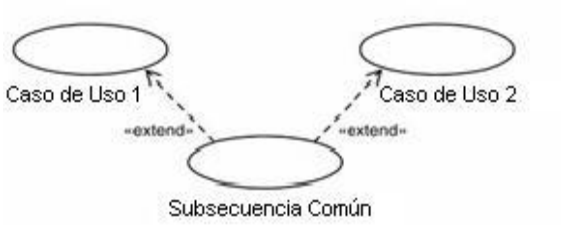


Figura 3. Adición

- Especialización.

Este es 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.

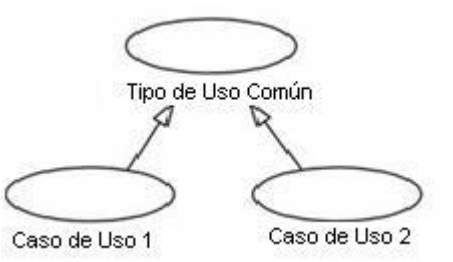


Figura 4. Especialización

CRUD (Creating, Reading, Updating, Deleting).

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

- Completo.

Este patrón consta de un caso de uso, llamado Información CRUD o 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.

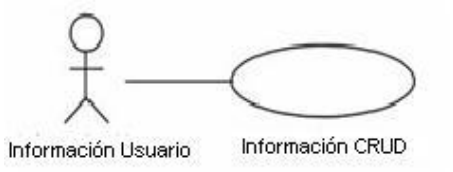


Figura 5. CRUD Completo

- Parcial

Este patrón alternativo 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.

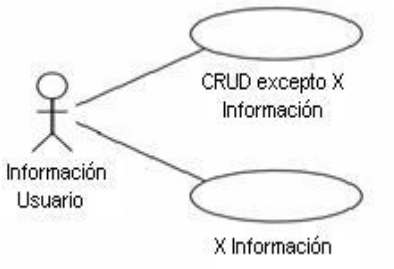


Figura 6. CRUD Parcial

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.

- 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.

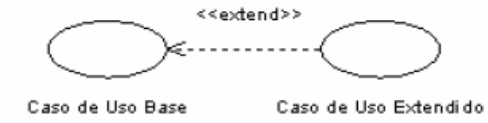


Figura 7. Extensión

- Inclusión.

Existe una relación de inclusión del caso de uso base con el caso de uso incluido. Este último puede ser instanciado como él 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.

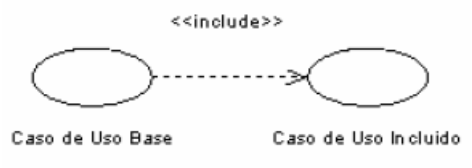


Figura 8. Inclusión

Múltiples actores.

- Roles diferentes:

Captura la concordancia entre actores manteniendo roles separados. Consta 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 9. 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.

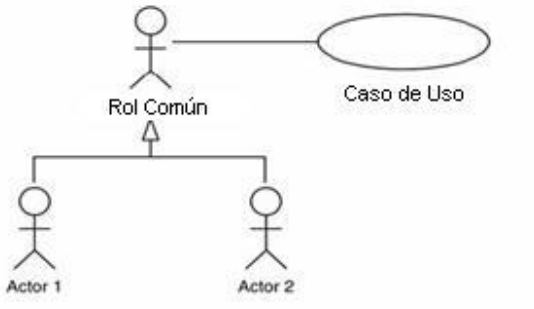


Figura 10. Roles Comunes

(ProfesoresISW, 2008-2009.)

Algunos de los patrones de casos de uso antes mencionados, serán aplicados como buena práctica al diagrama de casos de uso del sistema del módulo Reportes del proyecto SINAPSIS, para hacerlo más comprensible y correcto, en caso de que sea necesario.

1.8- Métricas para la especificación de requisitos.

Un elemento clave de cualquier proceso de ingeniería es la medición. Las medidas se emplean para entender mejor los atributos de los modelos que se crean. Pero, fundamentalmente, se emplean para valorar la calidad de los productos de ingeniería o de los sistemas que se construyen (Pressman, 2002).

Según el estándar IEEE 830, 1998 se considera que una especificación es de calidad si cumple con las siguientes características:

1. Especificación correcta: Un conjunto de requisitos es correcto sólo si todos los requisitos contenidos representan algo que es requerido para la construcción del sistema y no hay errores que afecten el diseño.

2. Especificación no ambigua: Un requisito es no ambiguo si y sólo si puede estar sujeto a una única interpretación.
3. Especificación completa: Si y sólo si, describe todos los requisitos relevantes para el usuario, incluyendo requisitos asociados con funcionalidad, desempeño, restricciones de diseño, atributos o interfaces externas.

4. Especificación consistente: si y sólo si no hay ningún subconjunto de requisitos descrito dentro de ella que está en conflicto con cualquier otro.

5. Especificación verificable: Se considera que una especificación es verificable si lo son cada uno de los requisitos constituyentes. Un requisito individual es verificable si existe un proceso acotado (en plazo y presupuesto) que permita determinar que el sistema construido satisface lo descrito en el propio requisito. La descripción detallada y prueba de los requisitos una vez implementados ayudan considerablemente en su verificación (IEEE, 1998).

Para medir la calidad de la Especificación de requisitos de software, se propone una lista de características, algunas de estas características son: especificidad (ausencia de ambigüedad), compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa.

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

$$\mathbf{Q_1} = \frac{\mathbf{R_{ii}}}{\mathbf{R_t}}$$

Donde $\mathbf{R_{ii}}$ es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas y $\mathbf{R_t}$ el total de requisitos. Los valores de la métrica pueden oscilar entre 0 y 1, pero como la ausencia de ambigüedad es un factor crítico para la calidad de la especificación se recomienda que el resultado al aplicar esta métrica alcance el valor máximo, es decir, 1.

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

$$\mathbf{Q_2} = \frac{\mathbf{n_A}}{\mathbf{n_A} + \mathbf{n_B}}$$

Donde $\mathbf{n_A}$ es el número de requisitos funcionales completos y $\mathbf{n_B}$ es el número de requisitos funcionales pobremente especificados. Como la compleción es difícil de medir, los valores obtenidos una vez aplicada esta métrica deben oscilar entre 0.7 y 1.

Una especificación se considera correcta cuando cada requisito contenido en ella represente una característica que el sistema debe poseer. La ***corrección*** de los requisitos se define usando la siguiente ecuación:

$$\mathbf{Q_3} = \frac{\mathbf{R_c}}{\mathbf{R_c} + \mathbf{R_{nv}}}$$

Donde $\mathbf{R_c}$ es el número de requisitos que se han validado como correctos y $\mathbf{R_{nv}}$ el número de requisitos que no se han validado como correctos todavía. El resultado de esta métrica está siempre entre 0 y 1. La corrección es un factor crítico para la calidad de la especificación por lo que se recomienda que el resultado al aplicar esta métrica alcance el valor máximo, es decir, 1.

La ***consistencia interna*** se calcula de la siguiente manera:

$$\mathbf{Q_4} = \frac{\mathbf{n_u} - \mathbf{n_n}}{\mathbf{n_u}}$$

Donde $\mathbf{n_u}$ es el número de requisitos especificados y $\mathbf{n_n}$ el número de requisitos en conflicto con otros requisitos en la especificación. Los valores de la métrica pueden oscilar entre 0 y 1, pero se recomienda que el resultado al aplicar esta métrica alcance el valor máximo, es decir, 1, por ser este un factor crítico dentro de la especificación.

La ***consistencia externa*** se determina como se muestra a continuación**.**

$$\mathbf{Q_5} = \frac{\mathbf{n_{ec}}}{\mathbf{R_t}}$$

Donde $\mathbf{n_{ec}}$ es el número de requisitos que son consistentes con otros documentos y $\mathbf{R_t}$ el total de requisitos. Los valores de la métrica pueden oscilar entre 0 y 1, pero se recomienda que el resultado al aplicar esta métrica alcance el valor máximo, es decir, 1, por ser este un factor crítico dentro de la especificación.

La ***comprensión*** de los requisitos se determina a partir de la relación que se muestra a continuación.

$$\mathbf{Q_6} = \frac{\mathbf{R_{bc}}}{\mathbf{R_t}}$$

Donde $\mathrm{R_{bc}}$ es el número de requisitos que todos los revisores entienden. Los valores de la métrica pueden oscilar entre 0 y 1, pero se recomienda que el resultado al aplicar esta métrica alcance el valor máximo, es decir, 1, por ser este un factor crítico dentro de la especificación. .

La ***capacidad de verificación*** de los requisitos se determina a partir de la relación que se muestra a continuación.

Donde $\sum_i \mathbf{c}(\mathbf{r_i})$ es el costo para verificar la presencia del requisito $\mathbf{i}$ y $\sum_i \mathbf{t}(\mathbf{r_i})$ el tiempo para verificar la presencia del requisito $\mathbf{i}$**.**

$$Q_7 = \frac{n_r}{n_r + \sum_i c(r_i) + \sum_i t(r_i)}$$

El resultado de esta métrica está siempre entre 0 y 1. Como la capacidad de verificación es difícil de medir, los valores obtenidos una vez aplicada esta métrica deben oscilar entre 0.7 y 1. (Davis, et al., 1993).

Las métricas de especificidad, compleción, corrección, comprensión, capacidad de verificación, consistencia interna y consistencia externa anteriormente explicadas, serán aplicadas en la validación de la Especificación de requisitos de software del módulo Reportes del proyecto SINAPSIS, para con ello lograr una mayor calidad de la misma.

1.9- Conclusiones.

Con el estudio realizado se pudo arribar a las siguientes conclusiones:

- Como guía para el desarrollo se empleará la metodología RUP por ser la más apropiada y adaptable a las características del módulo Reportes del proyecto SINAPSIS.

- Para el modelado del sistema se empleará el estándar UML, por ser de fácil comprensión para su aplicación por parte del equipo de desarrollo y por tener características adaptables a las necesidades de estos.

- Se utilizará la herramienta Visual Paradigm for UML 6.4 Enterprise Edition para el modelado del sistema y en la confección de prototipos de interfaz.

- Se estudiaron patrones de casos de uso, ya que constituyen buenas prácticas para hacer más comprensibles y correctos los diagramas de casos de uso.

- Se estudiaron métricas para la especificación de requisitos que pueden ser empleadas en la validación de esta.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

2.1- Introducción.

En el presente capítulo se muestra la solución que se propone para llevar a cabo el proceso de Ingeniería de Requisitos del módulo Reportes del proyecto SINAPSIS. Mediante el empleo de las etapas Elicitación, Análisis y Especificación de la IR se obtuvieron requisitos de software completos y consistentes, que son reflejados de manera clara, sencilla y sin ambigüedades. Para ello se aplicaron algunas técnicas propias de estas etapas. También se reflejan la especificación de requisitos, los actores del sistema, el diagrama de casos de uso del sistema, y la descripción de los casos de uso con sus respectivos prototipos de interfaz de usuario no funcional. Todo el proceso se realizó utilizando la metodología RUP como guía para el desarrollo.

2.2- Problemas de costo de la elicitación de requisitos. Soluciones.

En la etapa de elicitación de requisitos existen varios problemas que ayudan a comprender por qué ésta muchas veces se hace costosa. Con la identificación de los requisitos del módulo Reportes del proyecto SINAPSIS se evidenciaron problemas de comprensión, ya que, los clientes no estaban completamente seguros de lo que necesitaban, tenían una pobre compresión de las capacidades y limitaciones de su entorno de computación, no existía un total entendimiento del dominio del problema, tenían dificultades para comunicar las necesidades a los analistas y omitieron información por considerar que era obvia.

Para darle solución a estos problemas se realizaron un conjunto de actuaciones propuestas por la IR. Se valoró el impacto en el negocio y la viabilidad técnica del sistema a desarrollar. Se identificaron las personas que ayudarían a especificar los requisitos del módulo Reportes. Se definió el entorno técnico del sistema. Se definieron varios métodos de elicitación de requisitos, tales como: entrevistas, tormentas de ideas y prototipos. Se definieron los requisitos desde diferentes puntos de vista, ya que se solicitó la participación de muchas personas para ello. Además se identificó lo fundamental de cada requisito registrado.

2.3- Técnicas de Ingeniería de Requisitos utilizadas.

El proceso de Ingeniería de Requisitos es fundamental dentro del proceso de desarrollo de un producto de software, ya que a partir de los resultados obtenidos en este se realizará el sistema informático.

Para hacer posible la aplicación de este proceso en el módulo Reportes del proyecto SINAPSIS, inicialmente se realizó un estudio con el objetivo de conocer a grandes rasgos el entorno de negocio en el que queda enmarcado el proyecto a desarrollar, identificar las dudas que se pudieran tener en este sentido y preparar un conjunto de preguntas que pudieran aclarar algunos temas generales.

Luego se desarrollaron entrevistas con el cliente, como una de las formas más efectivas de obtención de información. Esta técnica fue realizada con el objetivo de identificar los requisitos y objetivos del sistema a construir. Seguido a esto se realizó el modelado de los prototipos de interfaz con el objetivo de obtener un mejor entendimiento del sistema que se va a desarrollar, sobre todo de los requisitos que estaban menos claros. Además se efectuaron talleres de requisitos haciendo uso de técnicas como la "Tormenta de ideas", donde se generaron discusiones que permitieron aclarar los requisitos del cliente e hicieron posible la confección de los artefactos.

Todos los requisitos fueron analizados con el objetivo de eliminar las inconsistencias y ambigüedades que estos pudieran tener, esto se realizó según la lista de comprobación que propone Pressman (Pressman, 2002). Además se clasificaron en funcionales y no funcionales. Estas actividades se llevaron a cabo con carácter iterativo e incremental, hasta quedar bien definidos los requisitos, los casos de uso y sus descripciones.

2.4- Requisitos de software.

Los requisitos del módulo Reportes del proyecto SINAPSIS se obtuvieron del cliente, mediante un conjunto de técnicas ya explicadas anteriormente (*Ver epígrafe 2.2*), cada uno fue analizado y agrupado en funcionales y no funcionales. Estos últimos a su vez se clasificaron según las categorías de usabilidad, eficiencia, seguridad, portabilidad, reusabilidad y capacidad (Consumo de recursos). Los requisitos funcionales constituyen una condición o capacidad que debe cumplir el sistema, es decir, el comportamiento interno del software; y los no funcionales son las propiedades o cualidades que este debe tener.

A los requisitos funcionales se le asignó una prioridad, alta o baja, en dependencia de las necesidades de los clientes o usuarios. Los de alta prioridad serán desarrollados primeramente debido a la importancia que tienen. Los requisitos fueron especificados en una plantilla estándar definida en la UCI[3], denominada Especificación de requisitos de software.

[3] Universidad de las Ciencias Informáticas.

2.4.1- Requisitos Funcionales.

Como resultado de la aplicación de las técnicas de elicitación y análisis de requisitos se obtuvieron las condiciones o capacidades que el sistema debe ser capaz de cumplir. Estos fueron clasificados según su prioridad en Alta (Esencial), Media (Deseado) o Baja (Opcional), en función de las necesidades de los clientes, y según su costo de desarrollo en Alto, Medio o Bajo. A continuación se presentan dos de estas condiciones con sus descripciones, para ver las restantes remitirse al documento "Especificación de requisitos de software".

1. **RF_R.30** *Generar reporte Cantidad de Proyectos por Nivel:* El sistema permitirá generar el reporte Cantidad de Proyectos por Nivel con la siguiente información:

   - Organismo.
   - Número de proyectos.
   - Porciento de número de proyectos sobre total.
   - Porciento de número de proyectos sobre total del organismo superior.
   - Monto planificado en el año (Bs.F).
   - Porciento de monto planificado sobre monto total.
   - Porciento de monto planificado sobre monto total del organismo superior.
   - Monto total planificado de los proyectos (Bs.F).

   A partir de los siguientes filtros de búsqueda:

   - Organismos.
   - Estado del proyecto.
   - Sector.
   - Lineamiento.
   - Año Ley de Presupuesto.
   - Rango de fechas de ejecución del proyecto.
   - Directriz.
   - Objetivo.
   - Estrategia.
   - Política.
   - Ámbito de localización.
     - Internacional.
     - Regional.

- Sin expresión territorial.
- Nacional.
  - Estado.
  - Municipio.
  - Parroquia.
  - Centro poblado.

- Rango de porciento de ejecución financiera.
- Rango de porciento de ejecución física.

Prioridad: Alta.

Costo de desarrollo: Bajo.

2. **RF_R.44** *Generar reporte de Trazas:* El sistema permitirá generar el reporte de Trazas con la siguiente información:

   - Usuario.
   - Nombre y apellido de la persona.
   - Organismo.
   - Acción realizada.
   - Fecha.
   - Hora.

A partir de los siguientes filtros de búsqueda:

- Organismo.
- Usuario.
- Código de proyecto.
- Rango de fechas.

Prioridad: Alta.

Costo de desarrollo: Bajo.

2.4.2- Requisitos No Funcionales.

Los requisitos no funcionales son las cualidades o propiedades que debe cumplir el sistema. Se obtuvieron del cliente, al igual que los requisitos funcionales. Para brindar una organización de estos,

en el presente trabajo se han agrupado por las categorías: usabilidad, eficiencia, seguridad, portabilidad, reusabilidad y capacidad.

Usabilidad.

- Cumplir con las pautas de diseño de las interfaces.
- Utilizar campos de selección en la interfaz en los casos que sea posible.
- Contar con una ayuda del módulo en línea para guiar en el uso de la interfaz.
- Permitir uso del teclado para realizar operaciones sobre el sistema (Permitir acceso rápido al sistema usando el teclado).
- Mostrar los valores de los campos numéricos utilizando separadores, según estándares.

Eficiencia.

- Responder en tiempos aceptables las peticiones que se realicen en el sistema.
  El sistema debe ser capaz de dar respuestas a las peticiones en un tiempo de no más de 30 segundos. Teniendo en cuenta el nivel de concurrencia que pueda existir, debe ser capaz de prestar servicio sin que se deterioren los tiempos de respuestas.

Seguridad.

- Registrar cada una de las operaciones llevadas a cabo por un usuario en el sistema.
- Definir niveles de acceso a la información en el sistema.

Portabilidad.

- El sistema debe ser una aplicación Web.
- Garantizar compatibilidad con navegadores de uso común.
  El sistema deberá ser compatible con los navegadores:
  - Microsoft Internet Explorer 6.0 o superior.
  - Mozilla Firefox 2.0 o superior.
- Garantizar que las actualizaciones del sistema sean a nivel central (Servidor).
- Emplear el protocolo ssl (Secure Socket Layer). para el aseguramiento del canal.

Reusabilidad.

- Garantizar que los formatos de los archivos de salida del sistema sean compatibles con los programas más comunes.

Los ficheros que genere el sistema deben utilizar formatos estándares como (rtf, pdf, xls, ods) de manera que sean compatibles con las siguientes herramientas:

- Microsoft Office 2003 o superior
- Acrobat Reader 6.0 o superior
- Open Office 2.3 o superior

Capacidad.

- Considerar características técnicas mínimas para la ejecución en clientes.

Para que un cliente de la aplicación pueda ejecutar procesos, en línea, considerados en el sistema el punto de acceso deberá cumplir con los siguientes requisitos mínimos.

- Procesador 2.0 GHz
- Memoria 512 MB.
- Disco duro 20 GB.
- Sistema Operativo Windows 98, 2000, XP o para Servidor o Linux.
- Navegador internet Explorer 6.0 o posterior, Mozilla Firefox 2.X
- Conexión a Internet. mínimo 56Kbps

Para que un cliente de la aplicación pueda observar y analizar resultados de los procesos consignados en documentos electrónicos, el punto de acceso deberá cumplir con los siguientes requisitos mínimos.

- Herramienta Microsoft Office 2003 o superior
- Herramienta Acrobat Reader 6.0 o superior
- Herramienta Open Office 2.3 o superior

2.5- Modelo de casos de uso del sistema.

El Modelo de casos de uso representa los requisitos funcionales del sistema. Se realizó con el objetivo de ayudar a los clientes, usuarios y desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen diversos tipos de usuarios, que son representados mediante los actores. Los cuales a su vez, utilizan el sistema al interactuar con los casos de uso, que no son más que fragmento de funcionalidad del sistema. Todos los actores y casos de uso del sistema, con sus respectivas descripciones forman un Modelo de casos de uso del sistema (Jacobson, et al., 2000).

Para la realización del Modelo de casos de uso del sistema del módulo Reportes del proyecto SINAPSIS, se utilizó una plantilla estándar definida en la UCI, denominada Modelo de sistema.

2.5.1- Actores del sistema.

Los actores no representan solamente a personas. Estos también pueden ser otros sistemas o hardware externos que interactúan con el sistema. Un usuario físico puede actuar como uno o varios actores. Asimismo, varios usuarios concretos pueden actuar como diferentes ocurrencias del mismo actor (Jacobson, et al., 2000). El modelo de casos de uso del módulo Reportes del proyecto SINAPSIS incluye dos actores, el actor Usuario y el actor Administrador. Estos se describen a continuación.

Actor	Descripción
Usuario	El actor Usuario es el encargado de consultar la información referente a los proyectos, sus montos, avance en la ejecución de los mismos, así como acceder a reportes consolidados que le permita ver de manera general el comportamiento de los proyectos enfocado a distintos aspectos como la distribución presupuestaria, las directrices, sectores, organismos, entre otros.
Administrador	El actor Administrador es el encargado de revisar los reportes referentes a las trazas de las acciones realizadas por los usuarios del sistema.

Tabla 1. Actores del Sistema

2.5.2- Casos de uso del sistema.

Anteriormente se explicó que los casos de uso forman parte del Modelo de casos de uso del sistema (*Ver epígrafe 2.4*). Estos no son más que, ‘’una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema’’, es decir, son un fragmento de funcionalidad del sistema. Estos además de representar funcionalidades del sistema también se utilizan como contenedores de los requisitos no funcionales, que son específicos de un caso de uso (Jacobson, et al., 2000).

Para el módulo Reportes del proyecto SINAPSIS quedaron definidos los siguientes 16 casos de uso:

1. Generar reporte Parametrizado.
2. Generar reporte cantidad de proyectos por Nivel.
3. Generar reporte cantidad de proyectos por Localización.
4. Generar reporte cantidad de proyectos por Sectores.
5. Generar reporte cantidad de proyectos por Directriz.
6. Generar reporte cantidad de población beneficiada por Nivel.

7. Generar reporte cantidad de población beneficiada por Sectores.
8. Generar reporte cantidad de población beneficiada por Directriz.
9. Generar reporte cantidad de empleos generados por Nivel.
10. Generar reporte cantidad de empleos generados por Sectores.
11. Generar reporte cantidad de empleos generados por Directriz.
12. Generar reporte distribución presupuestaria por proyectos.
13. Generar reporte presupuesto de proyectos por año.
14. Generar reporte POAN.
15. Generar reporte desembolso de proyectos por Nivel.
16. Generar reporte Trazas.

2.5.3- Diagrama de casos de uso del sistema.

Un diagrama de casos de uso describe parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par actor/casos de uso que interactúan, es decir, se utilizan para ilustrar los requisitos del sistema al mostrar cómo reacciona una respuesta a eventos que se producen en el mismo (Jacobson, et al., 2000).

El diagrama de casos de uso que se realizó, basado en el estudio de las funcionalidades del módulo Reportes del proyecto SINAPSIS, lo componen los dos actores y los 16 casos de uso ya definidos, este se muestra a continuación.

Figura 11. Diagrama de Casos de Uso del Sistema del módulo Reportes.

2.5.4- Especificación de casos de uso del sistema.

Las especificaciones de casos de uso se realizaron con el objetivo de lograr una mejor comprensión de los procesos a automatizar. En ellas se describen los diferentes flujos de sucesos entre el actor y el sistema, auxiliados de prototipos de interfaz de usuario no funcionales. A continuación se presentan dos de las especificaciones realizadas para el módulo Reportes del proyecto SINAPSIS, las restantes descripciones se pueden consultar en el documento “Modelo de Sistema”.

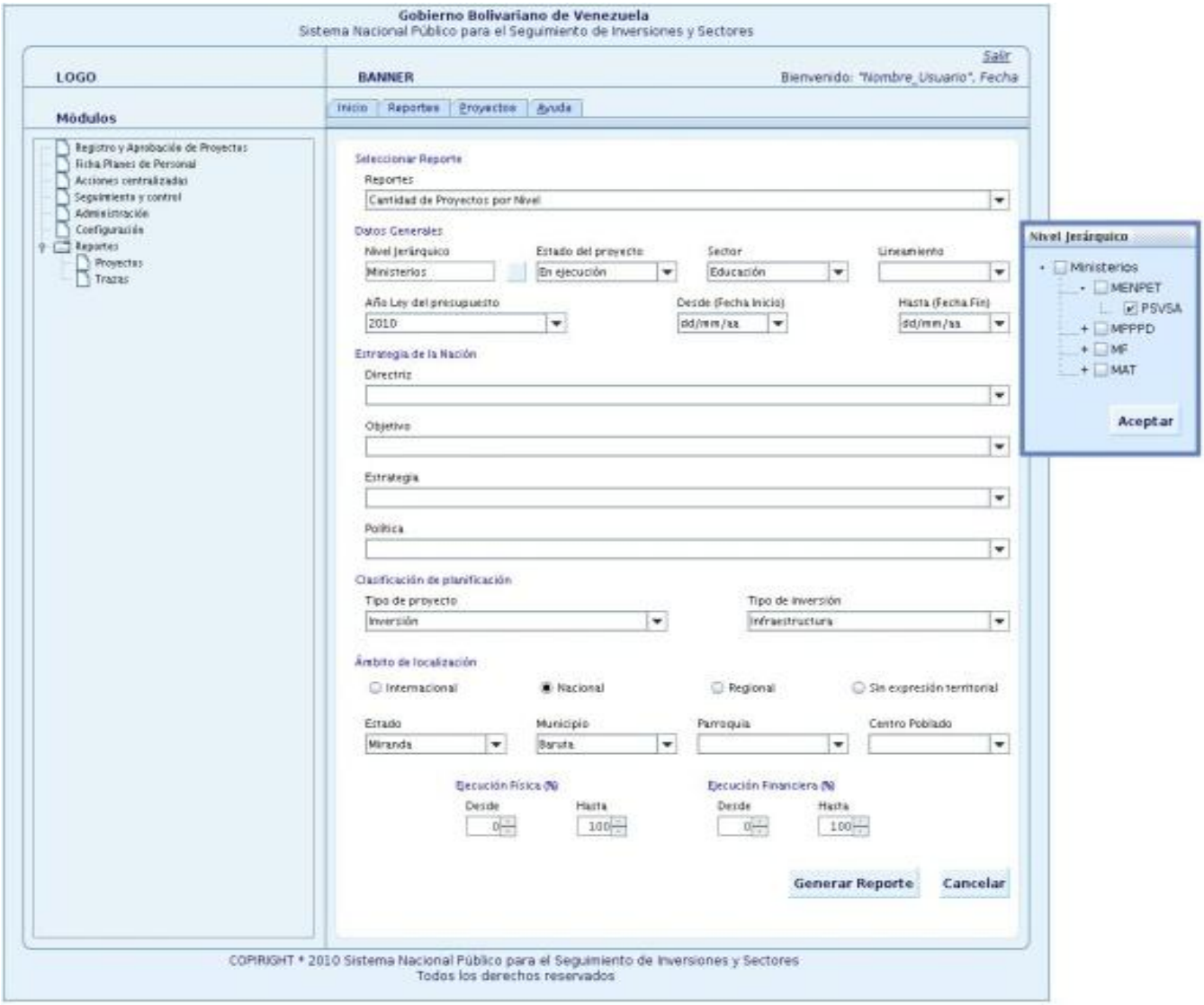
Generar reporte Cantidad de proyectos por nivel

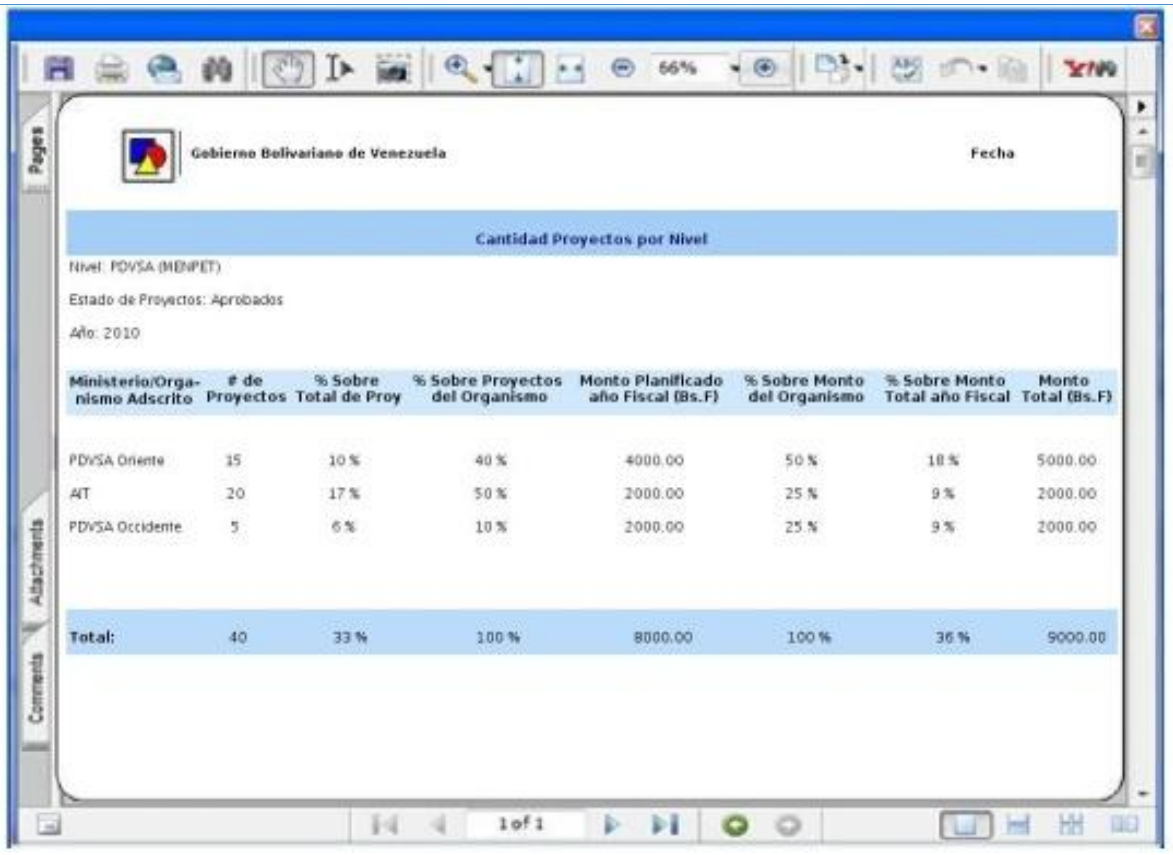
Caso de Uso:	Generar reporte Cantidad de proyectos por nivel
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el Usuario necesita generar un reporte. Consiste en que el Usuario selecciona el tipo de reporte a visualizar e introduce y selecciona los diferentes criterios o filtros por los que desea acotar los resultados del reporte. También tiene la posibilidad de imprimirlo y guardarlo en diferentes formatos. El caso de uso termina con la visualización del reporte.
Precondiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El usuario debe estar autenticado con los permisos necesarios.
Referencias	RF_R.1, RF_R.2, RF_R.3, RF_R.4, RF_R.5, RF_R.6, RF_R.7, RF_R.8, RF_R.9, RF_R.24, RF_R.25, RF_R.30
Prioridad	Crítico
Complejidad	Media
Nivel del caso de uso	Usuario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el actor Usuario selecciona la opción “Reportes” y dentro de esta la opción “Proyectos”.	2. El sistema solicita el tipo de reporte que se desea visualizar.
3. El actor Usuario selecciona el tipo de reporte “Cantidad de Proyectos por Nivel”.	4. El sistema muestra los filtros de búsqueda correspondientes. *Ver RF_R.30.*
5. El actor Usuario selecciona e introduce los datos correspondientes y selecciona la opción “Generar reporte”.	6. El sistema valida que los datos introducidos son correctos y que no hay campos obligatorios vacíos.
	7. El sistema muestra los datos seleccionados en los filtros de búsqueda y el listado de organismos adscritos

	según el nivel jerárquico seleccionado. *Ver RF_R.30.*
8. El actor Usuario selecciona una de las siguientes opciones: ➢ Guardar (Se invoca al CU Guardar reporte) ➢ Imprimir (Se invoca al CU Imprimir)	
9. El actor Usuario selecciona la opción Cerrar el reporte.	10. El sistema cierra el reporte, terminando así el caso de uso.

Prototipo de Interfaz





Flujo alterno al paso 5 “Operación cancelada”	
Acción del Actor	**Respuesta del Sistema**
5.a El actor Usuario selecciona la opción “Cancelar”.	5.b El sistema cancela la operación y regresa al paso 2 del flujo normal de eventos.
Flujo alterno al paso 6 “Datos incorrectos y/o campos vacíos”	
Acción del Actor	**Respuesta del Sistema**
	6.a El sistema valida que los datos introducidos no son correctos y/o que hay campos obligatorios vacíos.
	6.b El sistema muestra símbolos de error resaltando los campos obligatorios vacíos y/o donde se introdujeron datos incorrectos. Regresa al paso 4 del flujo normal de eventos.

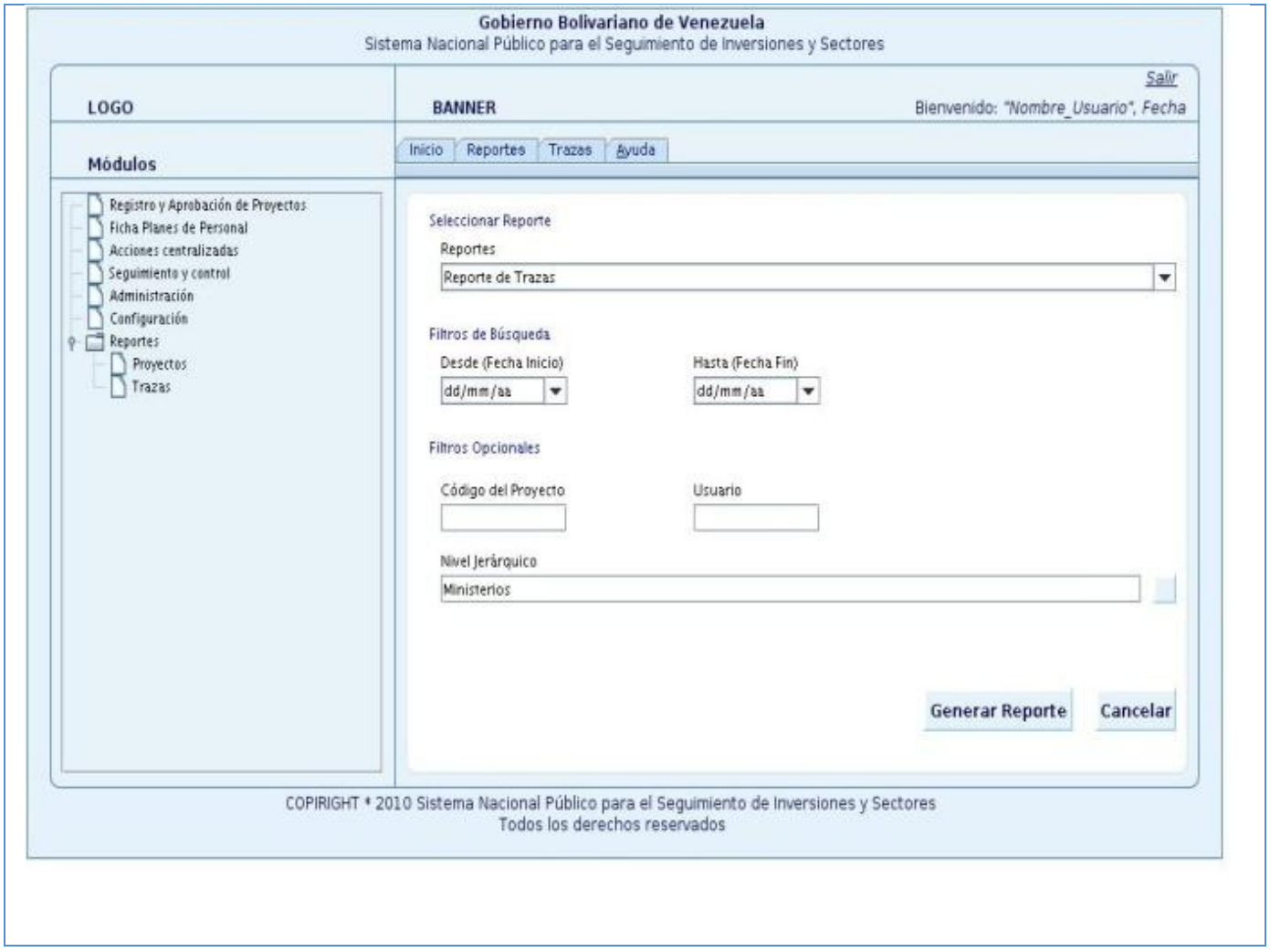
Pos-condiciones	El sistema queda en el mismo estado.

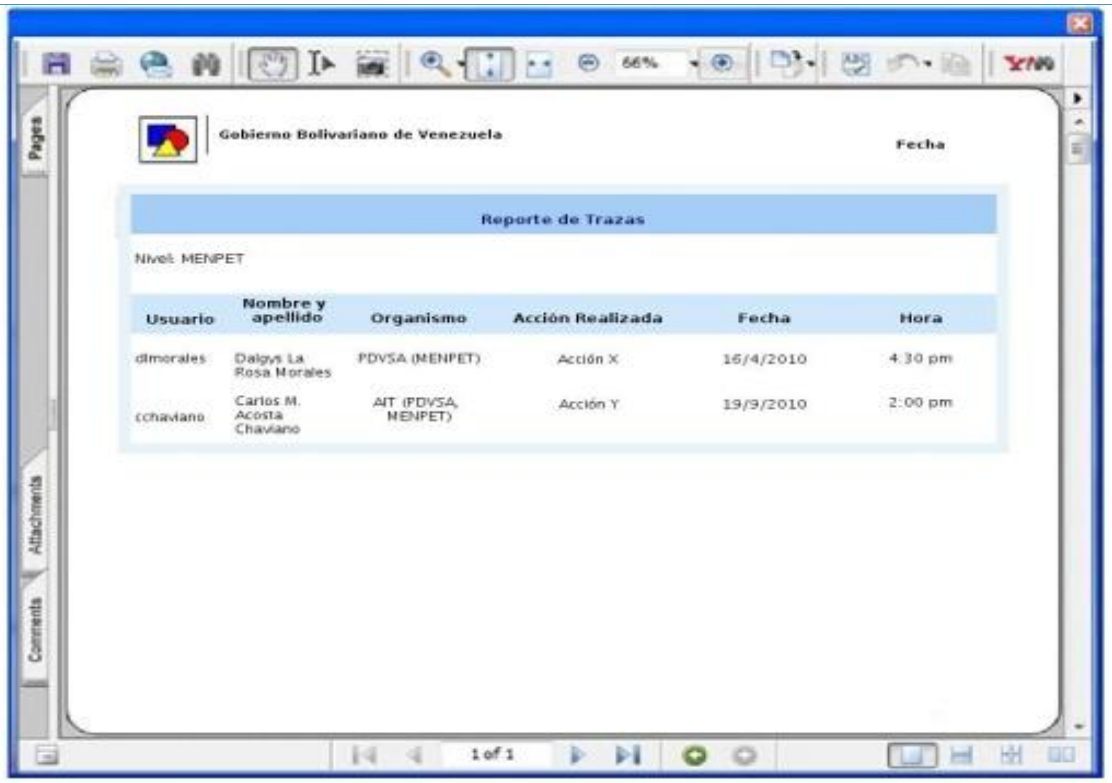
Generar reporte Trazas

Caso de Uso:	Reporte Trazas
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el Administrador necesita generar un reporte. Consiste en que el Administrador selecciona el tipo de reporte a visualizar e introduce y selecciona los diferentes criterios o filtros por los que desea acotar los resultados del reporte. También tiene la posibilidad de imprimirlo y guardarlo en diferentes formatos. El caso de uso termina con la visualización del reporte.
Precondiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El usuario debe estar autenticado con los permisos necesarios.
Referencias	RF_R.1, RF_R.2, RF_R.3, RF_R.4, RF_R.5, RF_R.6, RF_R.7, RF_R.8, RF_R.9, RF_R.44
Prioridad	Crítico
Complejidad	Media
Nivel del caso de uso	Usuario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el actor Administrador selecciona la opción “Reportes” y dentro de esta la opción “Trazas”.	2. El sistema solicita el tipo de reporte que se desea visualizar.
3. El actor Administrador selecciona el tipo de reporte “Reporte de Trazas”.	4. El sistema muestra los filtros de búsqueda correspondientes. *Ver RF_R.44.*
5. El actor Administrador selecciona e	6. El sistema valida que los datos introducidos son

introduce los datos correspondientes y selecciona la opción “Generar reporte”.	correctos y que no hay campos obligatorios vacíos.
	➢ 7. El sistema muestra los datos seleccionados en los filtros de búsqueda y el listado de las Trazas. *Ver RF_R.44.*
8. El actor Administrador selecciona una de las siguientes opciones: ➢ Guardar (Se invoca al CU Guardar reporte) ➢ Imprimir (Se invoca al CU Imprimir)	
9. El actor Administrador selecciona la opción Cerrar el reporte.	10. El sistema cierra el reporte, terminando así el caso de uso.
Prototipo de Interfaz	


Gobierno Bolivariano de Venezuela
Sistema Nacional Público para el Seguimiento de Inversiones y Sectores
Salir
LOGO
BANNER
Bienvenido: "Nombre_Usuario", Fecha
Módulos
Inicio
Reportes
Trazas
Ayuda
Registro y Aprobación de Proyectos
Ficha Planes de Personal
Acciones centralizadas
Seguimiento y control
Administración
Configuración
Reportes
Proyectos
Trazas
Seleccionar Reporte
Reportes
Reporte de Trazas
Filtros de Búsqueda
Desde (Fecha Inicio)
dd/mm/aa
Hasta (Fecha Fin)
dd/mm/aa
Filtros Opcionales
Código del Proyecto
Usuario
Nivel Jerárquico
Ministerios
Generar Reporte
Cancelar
COPIRIGHT * 2010 Sistema Nacional Público para el Seguimiento de Inversiones y Sectores
Todos los derechos reservados



Flujo alterno al paso 5 "Operación cancelada"	
Acción del Actor	**Respuesta del Sistema**
5.a El actor Administrador selecciona la opción "Cancelar".	5.b El sistema cancela la operación y regresa al paso 2 del flujo normal de eventos.
Flujo alterno al paso 6 "Datos incorrectos y/o campos vacíos"	
Acción del Actor	**Respuesta del Sistema**
	6.a El sistema valida que los datos introducidos no son correctos y/o que hay campos obligatorios vacíos.
	6.b El sistema muestra símbolos de error resaltando los campos obligatorios vacíos y/o donde se introdujeron datos incorrectos. Regresa al paso 4 del flujo normal de eventos.
Pos-condiciones	El sistema queda en el mismo estado.

2.6- Conclusiones.

En este capítulo se mostró la solución propuesta para el módulo Reportes del proyecto SINAPSIS, arribándose a las siguientes conclusiones:

- El empleo de técnicas de captura de requisitos propició que se obtuvieran las necesidades de los clientes y usuarios.

- Con el desarrollo de las etapas Elicitación, Análisis y Especificación de requisitos de la IR se obtuvieron los artefactos Especificación de requisitos de software y Modelo de Sistema.

- La modelación de los artefactos utilizando la herramienta Visual Paradigm, y UML como lenguaje de modelado, propició un mayor entendimiento entre clientes y desarrolladores.

CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS

3.1- Introducción.

En el presente capítulo se muestran los procedimientos y métodos utilizados para medir la calidad de los artefactos obtenidos, para el módulo Reportes del proyecto SINAPSIS, como parte de la etapa de validación de requisitos del proceso de IR. Para ello se emplearon listas de chequeo y se aplicaron un conjunto de métricas, con el fin de verificar la calidad de los mismos. También se realizó una matriz de trazabilidad para verificar que cada requisito de software estuviera asociado con al menos un caso de uso del sistema.

La obtención de los artefactos con la calidad requerida en cada fase, ayuda a minimizar los errores y contribuye en la construcción de un producto con mayor calidad que cumpla con todas las expectativas del cliente. Es por esto que se le confiere gran importancia al análisis de los resultados obtenidos en cada etapa del proceso de desarrollo de software.

3.2- Validación de la Especificación de requisitos de software.

Para poder tener seguridad en cuanto a la calidad de la "Especificación de los requisitos de software", es necesario realizar la medición de esta. El proceso de medición de la especificación, se llevó a cabo a través de listas de chequeo *(ver documento "Listas de chequeo")*, aplicadas en dos revisiones por parte del proyecto Calidad de la facultad. En la primera se detectaron un promedio de siete no conformidades, que se fueron corrigiendo hasta que la especificación cumplió cada uno de los aspectos de calidad contenidos en la lista. El Acta de liberación de la documentación revisada puede ser consultada en el anexo ***A***.

Con el objetivo de complementar la validación de la especificación mediante la aplicación de las listas de chequeo, se aplicaron además métricas para la calidad de la Especificación de los requisitos de software (*Ver epígrafe 1.8*). A continuación se muestra como se realizó el proceso.

Primeramente fue necesario calcular el número total de requisitos, para así poder aplicar las métricas que hacen uso de este.

Donde R_t representa el total de requisitos, R_f los requisitos funcionales y R_{nf} a los requisitos no funcionales.

$$R_t = R_f + R_{nf}$$

$$R_t = 44 + 14$$

$$R_t = 58$$

Especificidad (*Ver epígrafe 1.8*).

$$Q_1 = \frac{R_{ii}}{R_t}$$

$$Q_1 = \frac{58}{58} = 1$$

Corrección (*Ver epígrafe 1.8*).

$$Q_2 = \frac{R_c}{R_c + R_{nv}}$$

$$Q_2 = \frac{58}{58 + 0} = 1$$

Compleción (*Ver epígrafe 1.8*).

$$Q_3 = \frac{n_A}{n_A + n_B}$$

$$Q_3 = \frac{58}{58 + 0} = 1$$

Comprensión (*Ver epígrafe 1.8*).

$$Q_4 = \frac{R_{bc}}{R_t}$$

$$Q_4 = \frac{58}{58} = 1$$

Capacidad de verificación (*Ver epígrafe 1.8*).

$$Q_5 = \frac{n_r}{n_r + \sum_i c(r_i) + \sum_i t(r_i)}$$

$$Q_5 = \frac{58}{58 + 5.8 + +7.9} = 0.82$$

Consistencia interna (*Ver epígrafe 1.8*).

$$Q_6 = \frac{n_u - n_n}{n_u}$$

$$Q_6 = \frac{58 - 0}{58} = 1$$

Consistencia externa (*Ver epígrafe 1.8*).

$$Q_7 = \frac{n_{ec}}{R_t}$$

$$Q_7 = \frac{58}{58} = 1$$

Para un mejor entendimiento de los resultados obtenidos a partir de la aplicación de las métricas, se muestra la gráfica de la Figura 12.

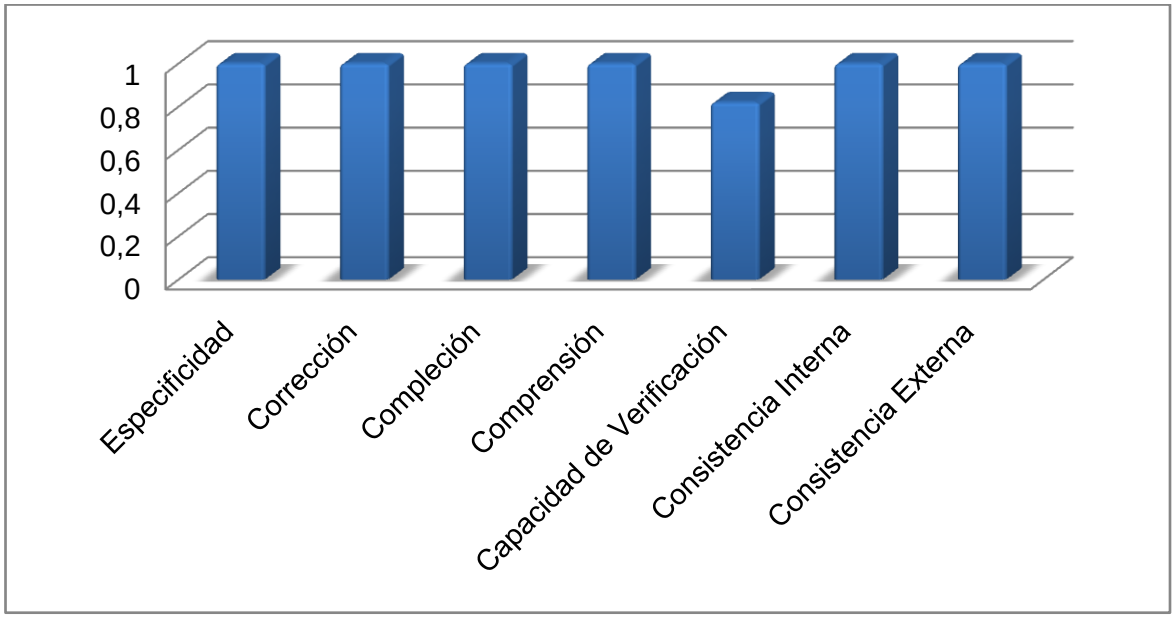


Figura 12. Resultados de las métricas aplicadas a la Especificación de requisitos.

Luego de aplicadas las métricas a la Especificación de requisitos del módulo Reportes del proyecto SINAPSIS, si se observan detenidamente los resultados, se puede apreciar que la especificación cuenta con la calidad requerida, ya que a medida que los resultados de las métricas se acercan a 1 (equivalente al 100%) se alcanza mayor calidad, lo cual contribuye en gran medida a lograr un mejor entendimiento entre clientes y desarrolladores.

Dentro de los atributos que se evaluaron se encuentra la ausencia de ambigüedad, cuyo resultado garantiza que cada uno de los requisitos fue interpretado de la misma forma, ya que los revisores involucrados en el proceso coincidieron en todas las interpretaciones.

Además se puede observar el alto grado de compleción, a pesar de ser un factor difícil de medir, su valor se debe fundamentalmente a que todos los requisitos que el software debe cumplir han sido incluidos y especificados.

Por otra parte se evaluaron la corrección y la comprensión, los cuales obtuvieron un valor óptimo, demostrando que se está en presencia de una especificación que fue bien comprendida por los revisores y que todos los requisitos representan una capacidad o cualidad que debe estar presente en el sistema a construir.

Además de las anteriormente mencionadas, se tuvieron en cuenta la consistencia interna y externa, para las cuales se obtuvo resultados satisfactorios, lo cual significa que ningún requisito está en conflicto con otro.

Por último, y no por ello se considera el atributo menos importante, es válido analizar la capacidad de verificación, para la cual también se obtuvo un valor bastante alto, teniendo en cuenta para este caso que los valores a partir de 0.7 se consideran satisfactorios, demostrando con esto que cada requisito dentro de la especificación puede ser verificado en un proceso acotado, en cuanto a plazo y presupuesto, que permitirá determinar que el sistema construido satisface lo descrito en cada uno de estos.

3.3- Matriz de trazabilidad.

La matriz de trazabilidad es una de las técnicas estudiadas. Se realizó con el objetivo de verificar que cada requisito de software estuviera asociado con al menos un caso de uso del sistema (Escalona, et al., 2002). Seguidamente se muestra la matriz de trazabilidad que corrobora lo antes planteado:

Requisitos	Casos de Uso															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RF_R.1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.2	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.3	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.5	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.6	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.7	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.8	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.9	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RF_R.10	x															
RF_R.11	x															
RF_R.12	x															
RF_R.13	x															
RF_R.14	x															
RF_R.15	x															
RF_R.16	x															
RF_R.17	x															
RF_R.18	x														x	
RF_R.19	x														x	
RF_R.20	x														x	
RF_R.21	x			x	x		x	x		x	x	x	x			
RF_R.22	x			x	x		x	x		x	x	x	x			
RF_R.23	x			x	x		x	x		x	x	x	x			
RF_R.24		x	x	x	x							x				
RF_R.25		x	x	x	x											
RF_R.26						x	x	x								
RF_R.27						x	x	x								
RF_R.28									x	x	x					
RF_R.29	x															
RF_R.30		x														

RF_R.31			x													
RF_R.32				x												
RF_R.33					x											
RF_R.34						x										
RF_R.35							x									
RF_R.36								x								
RF_R.37									x							
RF_R.38										x						
RF_R.39											x					
RF_R.40												x				
RF_R.41													x			
RF_R.42														x		
RF_R.43															x	
RF_R.44																x

Tabla 2. Matriz de trazabilidad

Como se puede apreciar en la matriz de trazabilidad de la Tabla 2, cada requisito está asociado con al menos un caso de uso del sistema.

3.4- Validación del Modelo de sistema.

El “Modelo de sistema” junto a la “Especificación de Requisitos”, los casos de uso, los actores y la descripción de ambos, constituyen los principales artefactos obtenidos durante la aplicación del proceso de IR al módulo Reportes del proyecto SINAPSIS.

Con el objetivo de garantizar la factibilidad del “Modelo de sistema”, el mismo fue sometido a dos revisiones por parte del proyecto Calidad de la facultad, lo cual incluye también la revisión de los actores del sistema, el diagrama de casos de uso del sistema y la especificación de casos de uso. En la primera revisión se detectaron un promedio de siete no conformidades, las cuales fueron solucionadas. Para llevar a cabo estas revisiones se aplicaron listas de chequeo, las cuales pueden ser consultadas en el documento “Listas de chequeo”.

Como constancia de las revisiones realizadas se encuentra el Acta de liberación de la documentación en el anexo ***A***.

3.5- Aceptación del cliente.

Luego de liberados los artefactos anteriormente mencionados (*Ver epígrafe 3.4*), fueron presentados a los clientes del sistema, los cuales procedieron a su revisión, con el objetivo de verificar que las especificaciones mostradas cumplieran con todas las expectativas, necesidades, condiciones y propiedades que se espera obtener con el desarrollo del módulo Reportes del proyecto SINAPSIS. Los artefactos presentados, es decir, la "Especificación de requisitos de software" y el "Modelo de sistema" fueron aceptados, quedando de esta forma legalmente firmadas en el acta de aceptación, convirtiéndose estos artefactos en compromisos de obligatorio cumplimiento por parte del equipo de desarrollo, para con los clientes del sistema.

Como constancia de haber realizado un adecuado proceso de elicitación, análisis y especificación de requisitos se muestra el acta de aceptación del módulo Reportes del proyecto SINAPSIS (*ver anexo **B***) por parte del cliente.

3.6- Conclusiones.

En el capítulo se llevó a cabo el análisis de los resultados obtenidos durante el desarrollo del trabajo de diploma, es decir, se evaluaron la Especificación de requisitos de software y el Modelo de sistema, arribándose a las siguientes conclusiones:

- La aplicación de las métricas para evaluar la calidad de la Especificación de los requisitos de software demostró que la misma cuenta con los aspectos de calidad requeridos.

- El realización de la matriz de trazabilidad permitió verificar que todos los requisitos estuvieran asociados con al menos un caso de uso del sistema.

- La aplicación de las listas de chequeo para evaluar la calidad, tanto de la Especificación de requisitos como del Modelo de sistema, posibilitó que estos cumplieran con los aspectos de calidad requeridos.

CONCLUSIONES

Al finalizar el presente trabajo de diploma se arriba a las siguientes conclusiones:

- La interacción con los clientes del sistema propició que se obtuvieran resultados satisfactorios en la identificación de las condiciones y capacidades que debe cumplir el módulo Reportes del proyecto SINAPSIS.

- Los artefactos obtenidos, entre ellos la Especificación de requisitos de software y el Modelo de sistema, propiciaron un mayor entendimiento y un acuerdo común entre los clientes y los desarrolladores, con respecto a las funcionalidades que el módulo Reportes debe brindar.

- El análisis de los resultados obtenidos, es decir, de la Especificación de requisitos y del Modelo de sistema, permitió medir el grado de factibilidad de estos en función de la calidad, funcionalidad, ambigüedad y consistencia, lo cual demostró que cuentan con la calidad requerida.

RECOMENDACIONES

- Realizar el análisis, diseño e implementación del módulo Reportes a partir del modelado de sistema presentado.

- Continuar perfeccionando el modelado de sistema mediante la actualización de los cambios que sean necesarios durante las etapas de análisis, diseño, implementación y pruebas.

- Hacer un seguimiento de los requisitos de software durante las posteriores fases de desarrollo aplicando la Gestión de requisitos que propone la Ingeniería de Requisitos.

BIBLIOGRAFÍA

1. **Adriano, Natalia Valeria. 2006.** *"Comparación del Proceso de Elicitación de Requerimientos en el desarrollo de Software a Medida y Empaquetado. Propuesta de métricas para la elicitación" , (Maestría en Ingeniería de Software).* Universidad Nacional de La Plata. Facultad de Informática. Córdoba (Argentina) : s.n., 2006.

2. **Báez, Griselda y Barba, Silvia I. 2001.** Departamento de Informática. Instituto Superior Politécnico "José Antonio Echeverría". [En línea] 2001. [Citado el: 12 de febrero de 2010.] http://www.inf.puc-rio.br/wer01/Mod-Req-1.pdf..

3. **Beck, K. 2000..** *"Una explicación de la programación extrema. Aceptar el cambio".* s.l. : Addison Wesley, 2000. ISBN/ 0201616416.

4. **Campderrich, Benet. 2003.** *"Ingeniería del software". [En línea].* s.l. : UOC, 2003. ISBN/ 8484297934.

5. **Christel, M G y K.C, Kang. 1992.** *"Issuesin Requirernents Elicitation".* 1992.

6. **Cockburn, A. A. Cockburn. 2001.** *Writing Effective Use Cases.* s.l. : Adison- Wesley, 2001.

7. **Davis, S. Overmyer, y otros. 1993.** *Identifying and measuring quality in software requirements specification.* California : Los Alamitos, 1993.

8. **Dorfman M. y Thayer, R. 1997.** *"Software Engineering", IEEE Computer Society Press.* s.l. : Los Alamitos, 1997.

9. **Durán, Amador y Bernárdez, Beatriz. 2000.** *"Metodología para la Elicitación de Requisitos de Sistemas Software".* Sevilla : s.n., 2000.

10. **Escalona, María José y Koch, Nora. 2002.** Departamento de Lenguaje y Sistemas Informáticos. Univeridad de Sevilla. [En línea] Diciembre de 2002. [Citado el: 12 de febrero de 2010.]

11. **Figueroa, Roberth G, Solís, Camilo J y Cabrera, Armando A.** *"METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES". .* Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación. : s.n.

12. **Fornaris, Maite Sánchez, Alcantara Rabí, Dayanis Elvia y Hernández Luque, Eylin. 2010.** Vinculando.org. *Vinculando.org.* [En línea] 2010. [Citado el: 23 de febrero de 2010.]

http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistemas_informacion.html.

13. **Giraldo, Gloria Lucía y otros. 2009.** Universidad de los Andes. Facultad de Ingeniería. [En línea] 27 de marzo de 2009. [Citado el: 5 de marzo de 2010.] http://revistaing.uniandes.edu.co/pdf/A2%2029.pdf?ri=e616dad77cd6db29a676ac313920f111.

14. **Gómez, Julián. 2010.** SOFTONIC. [En línea] 2010. http://axure-rp.softonic.com/.

15. **GSInnova. 2007.** Grupo Soluciones Innova. [En línea] 2007. [Citado el: 4 de febrero de 2010.] http://www.rational.com.ar/herramientas/roseenterprise.html.

16. **Hernández Sampier, Roberto y otros. 2003.** *"Metodología de la Investigación".* La Habana : Félix Varela, 2003.

17. **Hernández, L. Rolando A y Coello González, Sayda. 2002.** *"El paradigma cuantitativo de la investigación científica" .* Ciudad de la Habana, Cuba : Eduniv, 2002. ISBN/ 959-16-0343-6.

18. **IEEE. 1998.** *IEEE Std 1233:1998. Guide for Developing System Requirements Specification.* s.l. : Institute of Electrical and Electronics Engineers, 1998. ISBN/ 0-7381-1723-4.

19. **INGENIA. 2006.** INGENIA: Software Libre. *INGENIA: Software Libre.* [En línea] 2006. [Citado el: 11 de febrero de 2010.] https://community.ingenia.es/redmine/wiki/1.

20. **International, Visual Paradigm. 2008.** Visual Paradigm. *Visual Paradigm.* [En línea] Visual Paradigm International, 2008. [Citado el: 11 de febrero de 2010.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml64.jsp.

21. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *"El Proceso Unificado de Desarrollo de Software".* España : Addison Wesley, 2000. ISBN/ 8478290362.

22. **Jeffries, R., Anderson, A. y Hendrickson, C. 2001.** *"Extreme Programming Installed".* s.l. : Addison-Wesley, 2001.

23. **Lilly, S. 1999.** *Use Case–Based Requirements: Review Checklist.* 1999.

24. **López Requena, Martín Luis. 2006.** Málaga.NET. [En línea] 2006. [Citado el: 16 de febrero de 2010.] www.malagadnug.org/ficheros/MSFMartinLuisReq.pdf.

25. **Lutowski, Rick. 2005.** *Software Requirements. Encapsulation, Quality and Reuse.* 2005.

26. **MasterMagazine. 2004.** MasterMagazine. *MasterMagazine.* [En línea] 2004. [Citado el: 11 de febrero de 2010.] http://www.mastermagazine.info/termino/7006.php.

27. **Microsoft Corporation. 2010.** Microsoft Office. [En línea] 2010. [Citado el: 24 de febrero de 2010.] http://office.microsoft.com/es-es/visio/HA101656403082.aspx.

28. **Milestone Consulting.** Milestone Consulting. [En línea] [Citado el: 5 de marzo de 2010.] http://www.milestone.com.mx/articulos/sysml_modelando_sistemas_y_sistemas_de_sistemas.htm.

29. **OMG. 2009.** Object Management Group (OMG). [En línea] 2009. [Citado el: 5 de marzo de 2010.] http://www.omg.org/spec/BPMN/1.2.

30. **ÖVERGAARD, Gunnar y PALMKVIST, Karin. 2004.** *"Use Cases: Patterns and Blueprints".* s.l. : Addison Wesley, 2004.

31. **Paradigm, Visual. 2007.** Descargas de Software. *Descargas de Software.* [En línea] 05 de Marzo de 2007. [Citado el: 11 de febrero de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

32. **Pressman, Roger S. 2002.** *"Ingeniería del Software: Un Enfoque Práctico", Quinta Edición.* México : McGraw Hill, 2002.

33. **Pressman, Roger S. 2002.** *Ingeniería de Software: Un enfoque práctico.* 2002.

34. *Profundización del flujo de trabajo de requerimientos.* **ProfesoresISW. ProfesoresISW, 2008-2009..** Universidad de las Ciencias Informáticas, La Habana, Cuba. : s.n., ProfesoresISW, 2008-2009.

35. **Rational Software Corporation. 2003.** *RUP Help.* 2003.

36. **Sánchez González, Carlos. 2004.** *"ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras", Tesis (Ingeniería Informática).* Universidad de Coruña, Facultad de Informática, Departamento de Tecnologías de la Información y las Comunicaciones : s.n., 2004.

37. **Sanchez, Ing. Informática María A. Mendoza. 2002.** informatizate. *informatizate.* [En línea] 27 de noviembre de 2002. [Citado el: 16 de febrero de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

38. **Sommerville, Ian. 2005.** *“Ingeniería del Software”.* México DF : Editorial Pearson, 2005.

39. **Systems, Sparx.** *Guía del Usuario de Enterprise Architect.* [Soporte Email: soporte@sparxsystems.com.ar] Creswick, Australia : Sparx Systems.

40. **—. 2000.** Sparx Systems. *Sparx Systems.* [En línea] 2000. [Citado el: 11 de febrero de 2010.] http://www.sparxsystems.com.ar/products/ea/index.html.

41. **Tarazona, Ivon y Gomez, Oriana. 2006.** alfa.facyt.uc.edu.ve. [En línea] 2006. [Citado el: 23 de febrero de 2010.] http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/download/exposiciones2005-2006/uml.pdf.

42. **Wiegers, Karl E. 2006..** *"More About Software Requirements: Thorny Issues and Practical Advice".* s.l. : Microsoft Press, 2006. ISBN/ 0735622671.

43. **—. 2003.** *Software Requirements: Practical techniques for gathering and managing requirements.* s.l. : Microsoft Press, 2003.

44. **Young, Ralph R. 2004.** *"The Requirements Engineering Handbook".* Boston : Artech House, 2004. ISBN/ 1580532667.

ANEXOS

A

Acta de liberación de los artefactos.

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Martes, 04 de mayo de 2010.

Luego de haber efectuado 2 iteraciones de revisiones a los artefactos: Especificación de Requisitos No Funcionales, Especificación de Requisitos y Modelo de Sistema del módulo Reportes del proyecto Sinapsis del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 7 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que se considera que los artefactos están correctamente y listos para ser utilizados.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

B

Acta de aceptación de los artefactos.

PDVSA

Caracas, 07 de Diciembre de 2009.

Señor **Juan Carlos Montané Izaguirre**
Jefe del Proyecto **"SISTEMA NACIONAL PÚBLICO PARA EL SEGUIMIENTO DE INVERSIONES Y SECTORES".**

Estimado: **Juan Carlos Montané Izaguirre**

Después de analizado los documentos:

- Especificación de requisitos de software. Módulo de Proyectos.
- Especificación de requisitos de software. Módulo de Acciones Centralizadas.
- Especificación de requisitos de software. Módulo de Seguimiento y Control.
- Especificación de requisitos de software. Módulo de Administración de Usuario.
- Especificación de requisitos de software. Módulo de Configuración.
- Especificación de requisitos de software. Módulo de Reporte.
- Especificación de requisitos de software. Módulo de Migración de Datos.
- Modelo de Sistema. Módulo de Proyectos.
- Modelo de Sistema. Módulo de Acciones Centralizadas.
- Modelo de Sistema. Módulo de Seguimiento y Control.

PETROLEOS DE VENEZUELA
Avenida Libertador, edificio Petróleos de Venezuela, Torre Este, La Campiña. Apartado Postal 61373. Caracas 1060-A. Venezuela
Teléfono: (02) 708 1111 Fax: (02) 708 3267 www.pdvsa.com

PDVSA

- Modelo de Sistema. Módulo de Administración de Usuario.
- Modelo de Sistema. Módulo de Configuración.
- Modelo de Sistema. Módulo de Reporte.
- Modelo de Sistema. Módulo de Migración de Datos.
- Arquitectura de Software.

Visión del Proyecto concretado en el **Anexo 3 al Convenio PDVSA-ALBET**, manifestamos nuestra conformidad.

Atentamente,

Saludos,

07/12/09

Saúl Chirinos Gutiérrez
Gerente del Centro de Servicios Comunes-Occidente.
Jefe de Proyecto Sistema Nacional Público para el
Seguimiento de Inversiones y Sectores.

PETROLEOS DE VENEZUELA
Avenida Libertador, edificio Petróleos de Venezuela, Torre Este, La Campiña. Apartado Postal 61373. Caracas 1060-A. Venezuela
Teléfono: (02) 708 1111 Fax: (02) 708 3257 www.pdvsa.com

GLOSARIO

- **Actores**: Roles pertenecientes a los usuarios, agrupados según sus iteraciones con las funcionalidades del sistema.
- **CASE:** Acrónimo de Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.
- **Caso de uso (CU)**: Representación de la agrupación de funcionalidades comunes. Representan un conjunto de iteraciones entre el sistema y sus actores.
- **Ejecución Financiera:** Realización financiera del proyecto, constituido por los pagos y las transacciones financieras referentes al proyecto.
- **Ejecución Física:** Realización física del proyecto, lo constituye las actividades que se deben realizar en el proyecto.
- **FONDEN:** Fondo de Desarrollo Nacional.
- **Fuentes de financiamientos:** Fuentes por las cuales le llegan los recursos a los entes.
- **IEEE:** Corresponde a las siglas de The Institute of Electrical and Electronics Engineers (el Instituto de Ingenieros Eléctricos y Electrónicos), es la asociación técnica y profesional, sin fines de lucro, más grande del mundo formada por profesionales de todas las disciplinas de la ingeniería. Su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales.
- **OMG:** Object Management Group es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas de la industria del software, como IBM, Apple, Sun Microsystems y HP. Se encarga de la definición y el mantenimiento de estándares para aplicaciones de la industria de la computación, como UML, CORBA y otros.
- **PDF:** acrónimo del inglés Portable Document Format (formato de documento portátil), es un formato de almacenamiento de documentos, de tipo compuesto (imagen vectorial, mapa de bits y texto). Está especialmente ideado para documentos susceptibles de ser impresos, ya que especifica toda la

información necesaria para la presentación final del documento, determinando todos los detalles de cómo va a quedar, no requiriéndose procesos anteriores de ajuste ni de maquetación.

- **Plan Operativo Anual de la Nación (POAN):** Es la expresión de la de Planificación y Asignación de Recursos por Proyectos. Los Proyectos del POAN trascienden las rutinas institucionales, las llamadas operaciones de mantenimiento institucional. Por lo tanto, los Ministerios y sus entes adscritos deben identificar, diseñar y justificar sus propuestas de Proyectos- POAN, en estricto apego a los propósitos del desarrollo nacional y a los objetivos y metas de corto y mediano plazo.

- **Plug-in:** Un complemento (o plug-in en inglés) es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

- **Programación Orientada a Objetos (POO):** Es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (propiedades o datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). Expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

- **Refactorización (Refactoring):** Tipo de reestructuración de código que se aplica en desarrollos orientados a objetos. Tiene como objetivo limpiar el código para lograr una mejora en el diseño del software y ayudar a encontrar errores ocultos.

- **Release:** Producto final preparado para lanzarse como versión definitiva a menos que aparezcan errores que lo impidan. Implementa todas las funciones del diseño y se encuentra libre de cualquier error que suponga un punto muerto en el desarrollo.

- **Reporte**: Se refiere a la información lógica, relevante, y organizada, obtenida a partir de la recuperación de datos incluidos en el sistema.

- **XMI:** XML Metadata Interchange (XML de Intercambio de Metadatos) es una especificación para el Intercambio de Diagramas, escrita para proveer una manera de compartir modelos UML entre diferentes herramientas de modelado.

- **XML:** Acrónimo de Extensible Markup Language («lenguaje de marcas ampliable»), es un metalenguaje extensible de etiquetas, aunque no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.