Universidad de las Ciencias Informáticas
Facultad 3

Título: Modelado del Sistema

de Reportes para el Ministerio del Poder Popular

para la Energía y Petróleo de la

República Bolivariana de Venezuela.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Yamilé Mesa Leandro

Maylen Machado Acosta

Tutor:

Ing. Luis Ernesto Saballo López

Ciudad de la Habana

Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yamilé Mesa Leandro

_____

Firma del Autor

Maylen Machado Acosta

_____

Firma del Autor

Ing. Luis E. Saballo López

_____

Firma del Tutor

"La función de un buen software es hacer que lo complejo aparente ser simple"

Grady Booch

Agradecimientos

Esta ha sido la parte más difícil de la tesis, pues son muchas las personas a las que le tengo que agradecer y no quisiera dejar de nombrar, pero como se que va a ser imposible les pido disculpas con antelación y les doy mil gracias a todos los que he conocido en mi paso por la vida, que aunque sus nombres no estén reflejados en este pequeño espacio, los tengo presente siempre. Son muchas las personas que tienen un lugar en mi corazón, que han formado parte de este camino lleno de felicidad, inseguridad, alegrías, de amor. Son muchas las personas que de una forma u otra han ayudado a que este sueño, no solo mío, se haga realidad.

A mi mamita linda por ser la persona más especial del mundo, por confiar siempre en mí ya que el amor de una madre no contempla lo imposible, por ser más que mi madre mi amiga.

A ti papi que aún cuando piensas que no te quiero lo suficiente, eres muy importante en mi vida. No me cabe concebir ninguna necesidad tan importante durante la infancia de una persona que la necesidad de sentirse protegido por un padre, y puedes estar seguro que siempre me sentí protegida por ti. Gracias, muchas gracias por confiar en mi y por formar parte de este sueño, nuestro sueño.

A ti Yuniel, y por qué no? Mi pochi, por regalarme los mejores años de mi vida, por brindarme tanto amor, por estar a mi lado cuando te necesité… Cada paso que doy hacia delante, es una mirada atrás buscando tu recuerdo.

A mi familia, en especial a mi abuelita Clarita, que aunque ya no se encuentra a mi lado, estoy segura de que se sentiría muy orgullosa de mi…Perdóname cuando no tuve mucha paciencia pero te quise, te quiero y te querré mientras viva.

Mimi, mi viejita querida, que me guías, aconsejas, apoyas, gracias muchas gracias por tu amor, eres mi otra abuelita porque la sangre no es lo más importante sino los sentimientos, eso lo sabemos de sobra!!!.

A Mariluz, eres como una madre para mí, porque veo en ti confianza, cariño…

A mis hermanitas Lesly, Aurora, Nana, Yeni, por estar siempre conmigo, por su amor, su comprensión, por ser siempre amigas… Me gustaría agradecértelo de todo corazón, pero para ti, mi querida amiga, mi corazón no tiene fondo.

Grisel, eres una amiga incomparable, de la infancia y de todos los tiempos, se que siempre podré contar contigo.

A mis mulatitas y amigas queridas, Mari, Suce y Yami, por ser mi hombro de apoyo, por decirme “todo estará bien” cuando necesité escuchar esas palabras, por darme siempre fuerzas para seguir, por todos los momentos buenos que pasamos juntas...las quiero mucho de todo corazón.

A mi compañera de tesis Maylen, pero más que eso mi amiga, por su cariño, por escucharme cuando más lo necesitaba y sé que fueron muchas las veces, por soportar también mis impulsos, sabes que eres importante para mi.

A ti Lidy por formar parte de mis amistades especiales, por tus consejos que aunque no lo creas han sido muy útiles para mí, por enseñarme que uno puede caerse pero que tiene que levantarse con la frente en alto. Muchas gracias por ser mi amiga, nunca te voy a olvidar...

A Pascual o mejor dicho Bartolo, por ser buen profesor, pero sobretodo por ser como un padre durante todos estos años, por estar ahí, por los momentos de alegría, diversión…

Mi grupo inmenso de amistades, como olvidarlos: el grupo 3112 con los cuales comencé a vivir esta nueva experiencia, a Julito mi amigo que por cosas de la vida quizás piense que no es así, a Ana, Aray, Mili, Dania, Susana, Jasmín, Ernestina, Alejandro, Reinier, a Yasmany que no estás aquí ahora pero se que disfrutarás este momento al igual que nosotros, a Ramón por tener confianza en mi, a Baby, Dayanis, Yusely, Lisandra, Yonelbys, Reynolds, a Carlitos (o vasito) por los ratos de risa jajaja, no teníamos para cuando parar.

Las amistades de la vocacional, y algunas de ellas de mucho antes, los tengo presente siempre: Nero, Eliza, Yanet Bacallao, Sury, Yoilén, Daimí, Lisandra, Cepero, Janier, Yandy, Ale, Carmen, Zuanly, Lianet, Yadira, Javier, Maikel, Luis mi viejo amigo, Annelys Gato.

A Roger por hacerme reír en ocasiones donde parecía imposible lograrlo.

A mi padrastro Luis, por ser como eres con mi mamá, por toda tu ayuda.

A todos los profesores que han contribuido en mi formación como profesional, en especial a Lily, Manolo, más que profesores fueron un ejemplo a seguir.

A mi tutor, que no por último es menos importante, por su apoyo, por los momentos de risas aún cuando estábamos tensas por la tesis, por darnos siempre muchas fuerzas y esperanzas, además de la seguridad de que todo saldría bien.

Me gustaría terminar con una frase que considero única para un momento como este:

"No te pongas triste ante una despedida. Una despedida es necesaria para volver a reencontrarse. Y un reencuentro, después de un momento o después de toda una vida, es algo inevitable si somos amigos de verdad".

Yami

Agradecimientos

A mi mamá y mi papá por ser los mejores padres del mundo, por todo su amor, cariño, apoyo y comprensión.

A Mile y Mito, que más que hermanos, han sido padres para mí.

A mis sobrinos Bryan, Abney y Osmany, por toda su ternura y amor.

A mi abuelos Gerardo y Antonio y mi abuela Edesa que aunque ya no están me ayudaron siempre y se que estarían orgullosos de mí.

A mi abuela Yaya, mi tía Titica y primos Liobel, Luisander y Sergio por todo su cariño y apoyo.

A mi amor por haber llenado mi vida de felicidad cuando menos pensaba que pudiese ocurrir, gracias por haberme apoyado, comprendido y amado como lo has hecho durante todo este tiempo.

A mi tutor Luis Ernesto, por todo su aliento, apoyo, constancia y dedicación.

A Isanny, Hellen, Yilianny, Dainerys y Yuliet, por ser más que amigas, hermanas.

A mi compañera de tesis Yami, que además de eso ha sido mi amiga desde el comienzo de esta carrera, y por ende ha aguantado mis malcriadeces desde entonces.

A Yanela, Zaira, Raidanys, Anabel, Dania, Susana, Yeilis, María A, Yamilka, Sucell, Jasmín, Maylen F, Yaimara, Lisandra, Damila, y muchas más que no menciono por problemas de espacio, pero que de una u otra forma me han ayudado durante este largo camino.

A Jose, Pichi, Coppen, Oliver, Osain, Damián, Reinier Daniel, Alejandro, Ramón, Gallego, Abe, Yosbel, por todo su cariño y amistad.

A mis grupos 3112 y 3406, por todos los momentos buenos que pase junto a ustedes.

A todos los profesores que me han apoyado en mi formación como profesional.

Gracias a todos por contribuir a realizar mi sueño.

Los quiero…

MY

Dedicatoria

A mi mamá:

Tus brazos siempre se abren cuando necesito un abrazo. Tu corazón sabe comprender cuándo necesito una amiga. Tus ojos sensibles se endurecen cuando necesito una lección. Tu fuerza y tu amor me han dirigido por la vida y me han dado las alas que necesitaba para volar.

A mi papá:

Siempre has estado a mi lado, brindándome seguridad, amor.

Para Uds. mis seres más queridos que siempre esperaron que este momento llegara.

Los quiero

Yami

A mi mamá y mi papá por su apoyo constante, por sus enseñanzas y valores, por su paciencia, confianza y amor en todo momento.

A mis hermanos por toda la confianza que siempre han depositado en mí, y sobre todo por su cariño.

A todos mis seres queridos por ser la luz inspiradora de mis sueños.

MY

Resumen

La Oficina de Tecnología e Informática (OTI) del Ministerio del Poder Popular para la Energía y Petróleo (MENPET) de la República Bolivariana de Venezuela necesita la implementación de un Sistema de Reportes, el cual será desarrollado por un proyecto productivo de la Universidad de las Ciencias Informáticas.

La Ingeniería de Requerimientos (IR) es la disciplina de Ingeniería del Software que permite desarrollar una especificación completa, consistente y no ambigua, en dónde se describen las funciones que realizará el sistema. Además sirve como base para acuerdos comunes entre todas las partes involucradas, de ahí su importancia para la obtención de aplicaciones que cubran las expectativas de los usuarios.

El presente trabajo tiene como propósito principal desarrollar las actividades de la IR, para lograr que los clientes y desarrolladores tengan un criterio unificado respecto a los requisitos de software que debe cumplir el Sistema de Reportes para el MENPET, y de esa manera satisfacer las necesidades de los usuarios. Por esta razón se generan los artefactos necesarios para obtener una buena documentación que garantice dicho entendimiento y la futura implementación del sistema, definiéndose RUP como metodología de desarrollo de software, UML como lenguaje de modelado y Visual Paradigm Enterprise Edition como herramienta CASE que permitió la modelación visual del sistema. Finalmente se evaluó el modelo propuesto mediante el uso de métricas dirigidas a garantizar la calidad de la especificación de requisitos y del modelo de casos de uso del Sistema, obteniéndose resultados positivos.

Palabras Clave

Proceso de desarrollo, Metodología de desarrollo, Ingeniería de Requerimientos, Sistema de Reportes, Métricas.

Tabla de Contenidos

Introducción

El Ministerio del Poder Popular para la Energía y Petróleo (MENPET) de la República Bolivariana de Venezuela, cuenta con un departamento de informática u Oficina de Tecnología e Informática (OTI), que es la encargada de dictar las políticas informáticas en este centro y de dar soporte informático al resto del personal y sus entes adscritos al desarrollar aplicaciones, las cuales entre las funcionalidades que brindan a sus usuarios finales, está la de permitir que estos generen reportes.

Dicho Ministerio necesita la implementación de un sistema que permita generar y gestionar reportes de información, el cual será desarrollado por un proyecto productivo de la Universidad de las Ciencias Informáticas.

La República Bolivariana de Venezuela en la actualidad se encuentra en un proceso de migración a software libre por todas las ventajas que este proporciona. Debido a esta situación el nuevo sistema que requiere el MENPET para generar y gestionar reportes será desarrollado bajo software libre.

La **situación problémica** es la siguiente: En el Ministerio del Poder Popular para la Energía y Petróleo de Venezuela se generan muy a menudo peticiones de información, que deben formar parte de reportes en varias aplicaciones, y esto provoca inconvenientes a la Oficina de Tecnología e Informática. Dentro de los inconvenientes está la modificación de las aplicaciones con el objetivo de incluir los nuevos reportes de información. Esta dificultad está presente a la hora de generar reportes pues las aplicaciones que desarrollan generalmente cuentan con reportes predeterminados, por lo que si aparecen nuevas necesidades de información al salir estos sistemas a producción, que a su vez ameritan incorporar nuevos tipos de reportes a las aplicaciones, los desarrolladores del software (personal de la OTI) tienen que modificar el código fuente de sus sistemas, lo cual atrasa y limita la inclusión de los nuevos reportes. Por todo lo antes descrito se plantea el **problema científico**: ¿Cómo lograr un entendimiento entre clientes y desarrolladores respecto a los requisitos del software, que permita de manera eficiente el desarrollo del Sistema de Reportes para el MENPET?

Dicho problema se enmarca dentro del **objeto de estudio**: Proceso de Desarrollo de Software, y el **campo de acción**: Ingeniería de Requerimientos.

Para dar cumplimiento al problema científico planteado anteriormente, el presente trabajo tiene como **objetivo general**: Desarrollar las actividades de la Ingeniería de Requerimientos, que garanticen el entendimiento entre clientes y desarrolladores, para el eficiente desarrollo del Sistema de Reportes para el MENPET.

Para poder orientarse dentro de esta investigación se plantea la siguiente **hipótesis:** Si se logran desarrollar las actividades de la Ingeniería de Requerimientos que permitan un entendimiento entre clientes y desarrolladores, de los requisitos del software, se facilitará un eficiente desarrollo del Sistema de Reportes para el MENPET.

Con el propósito de alcanzar el objetivo general de esta investigación se llevarán a cabo las siguientes **tareas**:

- Estudio de las metodologías de desarrollo de software, de las principales Herramientas CASE y lenguajes de modelado.
- Estudio acerca de la importancia del analista de sistemas en el proceso de desarrollo de software.
- Estudio del estado del arte referente a los sistemas de reportes.
- Desarrollo de un modelo del dominio.
- Identificación y descripción de los Requisitos Funcionales y No Funcionales que modelan el sistema informático.
- Definición y descripción de los casos de uso del sistema.
- Modelado de los casos de uso del sistema.
- Definición de Patrones de Caso de Uso. Fundamentación y aplicación de estos patrones.
- Definición de prototipo de interfaz de usuario.
- Aplicación de métricas para medir la calidad de los resultados obtenidos.

Con el objetivo de desarrollar las tareas propuestas, se utilizaron los siguientes **métodos científicos de la investigación**:

Métodos Teóricos:

Analítico – sintético: con el objetivo de buscar la esencia de los fenómenos, los rasgos que lo caracterizan y los distinguen, analizar teorías y documentos, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Histórico-lógico: analiza la trayectoria completa de los fenómenos, su condicionamiento a los diferentes períodos de la historia, poniendo de manifiesto la lógica interna de su desarrollo, de su

teoría y haya el conocimiento más profundo de su esencia. Este método expresa en forma teórica la esencia del objeto y explica la historia de su desarrollo.

Modelación: método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. El modelo como sustituto del objeto de investigación es semejante a él, existiendo una correspondencia objetiva entre el modelo y el objeto, siendo el investigador quien elabora dicho modelo. El modelo es el eslabón entre el sujeto y el objeto intermedio. (Hernández León, y otros, 2002)

Métodos Empíricos:

Entrevista: como técnica fundamental para obtener información necesaria y valiosa en el desarrollo de la investigación mediante conversaciones planificadas con los clientes.

La entrevista puede ser estructurada o no estructurada.

Para la realización de esta investigación se utilizó la **entrevista no estructurada** que es más abierta que la estructurada, prevé el tema pero no lleva un cuestionario rígido y puede variar de una persona a otra, es más flexible. Se aplica a especialistas en el tema, es una forma de obtener criterios de expertos. (Hernández León, y otros, 2002)

Los **Resultados Esperados** de este trabajo son:

- Modelo del Dominio.
- Requisitos Funcionales y No Funcionales.
- Modelo del Sistema.
- Análisis de los Resultados.

Capítulo 1

Fundamentación Teórica

1.1 Introducción.

El impacto del software en nuestra sociedad y en la cultura continúa siendo profundo. Al mismo tiempo que crece su importancia, la comunidad del software trata continuamente de desarrollar tecnologías que hagan más sencillo, rápido y menos costosa la construcción de programas de computadora de alta calidad.

Durante el desarrollo de este capítulo se realizará un estudio del estado del arte de los principales sistemas de reportes, de las principales metodologías, herramientas de desarrollo de software conocidas como herramientas CASE y de los lenguajes de modelado. Además, se profundizará en el papel del analista de sistemas en el proceso de desarrollo de software.

Para desarrollar un software que satisfaga las necesidades y expectativas de los clientes o usuarios finales es necesario aplicar correctamente la Ingeniería de Requerimientos, por lo que se explicará en qué consiste la misma, sus actividades y los beneficios que aporta.

1.2 Sistemas de Reportes.

Siempre que se piense en elaborar reportes, se tienen que distinguir claramente algunos aspectos:

1) Definición del reporte, momento en que el autor del reporte define los datos y la manera de presentación de estos. En esta etapa normalmente hay que definir conexiones a los distintos orígenes de datos para ver de donde obtener los resultados que debe reflejar el reporte.

2) Administración del reporte, está referido al hecho que en las organizaciones actuales se tienen distintas categorías de usuarios como por ejemplo los gerentes, los usuarios de servicio al cliente, etc. Por lo tanto, es importante definir quiénes serán los usuarios del reporte, para ello hay que publicar los reportes.

3) Entrega del reporte, es muy común en las organizaciones que muchos reportes sean requeridos de manera periódica, por lo que se podrían aprovechar distintos servicios como el de mensajería para que estos reportes lleguen a los usuarios requeridos.

Las tres acciones, mencionadas anteriormente, conforman lo que se denomina **"El Ciclo de Vida de un reporte".** (Hidalgo)

1.2.1 Principales sistemas existentes.

SQL Server Reporting Services.

Es una plataforma de reportes basada en servidores, la misma que puede ser empleada para crear y administrar reportes tabulares, de matrices, gráficos y de libre formato, la información de estos reportes pueden provenir de diferentes orígenes de datos. Los reportes que se definen pueden ser administrados a través de una conexión basada en Web.

Este servicio es una extensión a las capacidades Business Intelligence que nos provee de herramientas para almacenar información (Report Server), herramientas para crear reportes (Report Designer) y herramientas para administrar reportes (Report Manager).

Reporting Services provee servicios, herramientas e interfaces de programación (API), aunque no es necesario ser desarrollador para usarlo.

Ventajas de Reporting Services

✓ Cuenta con un lenguaje de especificación estándar denominado Report Definition Language o simplemente RDL, el cual es un lenguaje de formato XML, que se encarga de definir el reporte.

✓ Con SQL Server Reporting Services, se pueden conectar a cualquier repositorio de datos, a través de un .NET Data Provider, un proveedor OLE DB provider o uno de tipo ODBC.

✓ Para la distribución, los usuarios pueden acceder a los reportes en base a la infraestructura existente. Es decir, que los usuarios pueden acceder a los reportes a través de una barra de herramientas en el browser. Los reportes son accesados desde un repositorio centralizado, presentado como un folder en orden jerárquico.

✓ Otra de las grandes características de Reporting Services, es que puede distribuir el reporte en distintos formatos, como hojas de excel, documentos pdf, texto, XML, etc.

✓ La arquitectura de Reporting Services, permite a los desarrolladores preparar aplicaciones personalizadas que accedan a los reportes a través de una API que está expuesta como un Web service. (Hidalgo)

1.3 Proceso de Desarrollo de Software.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.

Se puede decir que un proceso define *quién* está haciendo *qué*, *cuándo*, y *cómo* alcanzar un determinado objetivo. En la ingeniería del software el objetivo es construir un producto software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible. (Ver Anexo #1)

Un proceso de desarrollo de software debe ser adaptable y configurable para cumplir con las necesidades reales de un proyecto y/o organización concreta. (Jacobson, y otros, 2004)

El conjunto de actividades necesarias para transformar los requisitos de usuario en un producto, se conoce en el Proceso Unificado de Desarrollo del Software como flujos de trabajo. Las principales actividades son:

- Análisis del Negocio.
- Ingeniería de Requerimientos.
- Análisis del Sistema.
- Diseño del Sistema.
- Implementación del software.
- Prueba o Validación del software.
- Planificación y Evolución.

Para dar cumplimiento al objetivo de este trabajo se desarrollarán los dos primeros flujos.

1.3.1 Papel del Analista de Sistemas.

El analista de sistemas surge de la necesidad de analizar, identificar y separar en procesos toda la información referente al software que se desee construir o mejorar, es el encargado de proponer soluciones y seleccionar la idea más idónea para el problema en cuestión. (Ver Anexo #2) (Kendall, y otros, 1997)

Jacobson, Booch y Rumbaugh, definen al Analista de Sistema como:

"el responsable del conjunto de requisitos que están modelados en los casos de uso, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específicos. El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente. Para la consistencia, el analista de sistemas puede utilizar un glosario para conseguir un acuerdo en los términos comunes, nociones y conceptos durante la captura de los requisitos.

Aunque el analista es responsable del modelo de casos de uso y de los actores que contiene, no es responsable de cada caso de uso en particular. Esto es una responsabilidad aparte, que pertenece al trabajador especificador de casos de uso. El analista de sistemas es también el que dirige el modelado y el que coordina la captura de requisitos."

Hay un analista de sistemas para cada sistema. No obstante, en la práctica, este trabajador está respaldado por un equipo (en talleres o eventos similares) que incluye otras personas que también trabajan como analistas. (Jacobson, y otros, 2004)

1.4 Ingeniería de Requerimientos.

La Ingeniería de Requerimientos (IR) cumple un papel primordial en el proceso de producción de software, ya que se enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de los usuarios o clientes; de esta manera, se pretende minimizar los problemas relacionados por la mala gestión de los requerimientos en el desarrollo de sistemas. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

1.4.1 Requerimientos: conceptos y características.

¿Qué son los requerimientos?

Se presenta a continuación la definición existente en el glosario de la IEEE de lo que es un requerimiento:

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.

2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

3. Una representación documentada de una condición o capacidad como en (1) o (2).

También, Ian Sommerville presenta una definición acerca de lo que es un requerimiento:

"Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste". (Sommerville, 2005)

Tipos de Requerimientos.

Los requerimientos de software pueden dividirse en 2 categorías: requerimientos funcionales y requerimientos no funcionales.

Los requerimientos funcionales son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones. Estos requerimientos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema.

Por otra parte los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

Características de un Requerimiento.

Es importante no perder de vista que un requerimiento debe ser:

- Especificado por escrito: Como todo contrato o acuerdo entre dos partes.
- Posible de probar o verificar. Si un requerimiento no se puede comprobar, entonces ¿cómo se sabe si se cumplió con él o no?
- Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.

- No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

Dificultades para definir los requerimientos.

Durante la etapa de especificación de requerimientos se pueden presentar muchos inconvenientes los cuales son importantes de identificar y prevenir, a continuación se presenta un listado con los problemas más comunes en este proceso:

- Los requerimientos no son obvios y vienen de muchas fuentes.
- Son difíciles de expresar en palabras (el lenguaje es ambiguo).
- La cantidad de requerimientos en un proyecto puede ser difícil de manejar.
- Un requerimiento puede cambiar a lo largo del ciclo de desarrollo.
- El usuario no puede explicar lo que hace.
- Tiende a recordar lo excepcional y olvidar lo rutinario.
- Hablan de lo que no funciona.
- Los usuarios tienen distinto vocabulario que los desarrolladores.
- Usan el mismo término con distinto significado. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

1.4.2 ¿Qué es la Ingeniería de Requerimientos?

El proceso de recopilar, analizar y verificar las necesidades del cliente o usuario para un sistema es llamado Ingeniería de Requerimientos. La meta de la Ingeniería de Requerimientos (IR) es entregar una especificación de requisitos de software correcta y completa. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

Según Rational Software "Ingeniería de requerimientos es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos, entre los clientes y el equipo del proyecto". (Ver Anexo #3)

La ingeniería de requisitos 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. (Pressman, 2005.)

1.4.3 Actividades de la Ingeniería de Requerimientos.

El proceso de ingeniería de requisitos puede ser descrito en los siguientes pasos distintos:

- Identificación de Requisitos.
- Análisis y Negociación de Requisitos.
- Especificación de Requisitos.
- Modelado del sistema.
- Validación de Requisitos.
- Gestión de Requisitos. (Pressman, 2005.)

Como toda división de tareas, no es una estricta representación de la realidad, sino que se hace con el fin de sistematizar la realización de la IR. En general la delimitación entre una actividad y la otra no es tan clara, ya que están sumamente interrelacionadas, existiendo un alto grado de iteración y retroalimentación entre una y otra. (Davyt Dávila, 2001)

Identificación de requisitos.

En esta fase se realizan las actividades involucradas en el descubrimiento de los requerimientos del sistema. Los analistas deben trabajar junto al cliente para descubrir el problema que el sistema debe resolver, los diferentes servicios que el sistema debe prestar, las restricciones que se pueden presentar, etc.

Esto no suele ser tarea fácil: muchas veces los clientes/usuarios no tienen una idea clara de sus necesidades reales, diversas personas dentro de la organización tienen necesidades encontradas, pueden existir limitaciones técnicas o tecnológicas para cumplir con algunos requerimientos, etc. Pero, en definitiva, descubrir los requerimientos del sistema no sólo implica preguntar a las personas qué quieren: es un proceso delicado que involucra comprender el domino de aplicación, es decir, obtener un conocimiento del área general de aplicación del sistema; comprender el problema en sí, lo que implica que se debe extender y especializar el conocimiento sobre el dominio general para que se

aplique al cliente en particular; comprender el negocio, por tanto, se debe entender en profundidad cómo es que este sistema interactuará, afectará a las partes del negocio que estarán involucradas y cómo puede contribuir a lograr las metas de la empresa; finalmente, comprender las necesidades y restricciones de los usuarios del sistema, en particular, se deben entender los procesos de trabajo que se supone que el sistema apoyará y el rol de cualquier otro sistema que actualmente se involucre en dichos procesos.

Es importante, entonces, que la identificación de requisitos sea efectiva, ya que la aceptación del sistema dependerá de cuán bien éste satisfaga las necesidades del cliente y de cuán bien asista a la automatización del trabajo. (Davyt Dávila, 2001)

Análisis y Negociación de Requisitos.

En esta etapa se estudian los requerimientos extraídos en la etapa previa a los efectos de poder detectar, entre otros, la presencia de áreas no especificadas, requisitos contradictorios y peticiones que aparecen como vagas e irrelevantes. (Griselda Báez, y otros, 2001)

Usualmente se hace un análisis luego de haber producido un bosquejo inicial del documento de requerimientos; en esta etapa se leen los requerimientos, se conceptúan, se investigan, se intercambian ideas con el resto del equipo, se resaltan los problemas, se buscan alternativas y soluciones, y luego se van fijando reuniones con el cliente para discutir los requerimientos. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

Especificación de Requisitos.

En esta fase se documentan los requerimientos acordados con el cliente, en un nivel apropiado de detalle.

En la práctica, esta etapa se va realizando conjuntamente con el análisis, pero se podría decir que la Especificación es el "pasar en limpio" el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML. (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software, 2006)

Modelado del Sistema.

Esta fase consiste en modelar el sistema propuesto con el objetivo de evaluar los componentes del sistema y sus relaciones entre sí, determinar cómo están reflejados los requisitos, y valorar como se ha concebido la <<estética>> en el sistema. (Pressman, 2005.)

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. (Pressman, 2005.)

En fin, la validación de especificaciones significa asegurarse de que el documento de requerimientos represente una descripción clara del sistema, y es una verificación final de que los requerimientos cubren las necesidades de los usuarios.

Esta etapa puede confundirse con la de análisis, pero la diferencia es clara: mientras que en el análisis se trabaja sobre el boceto del documento de requerimientos, en la validación se utiliza el documento final, lo que equivale a decir, los requerimientos "depurados". (Davyt Dávila, 2001)

Gestión de Requisitos.

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. (Pressman, 2005.)

1.4.4 Importancia de la Ingeniería de Requerimientos.

Los principales beneficios que se obtienen de la Ingeniería de Requerimientos son:

- Permite gestionar las necesidades del proyecto en forma estructurada: Cada actividad de la IR consiste de una serie de pasos organizados y bien definidos.
- Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados: La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- Disminuye los costos y retrasos del proyecto: es sabido que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro; especialmente aquellas decisiones tomadas durante la IR, ya que es una de las etapas de mayor importancia en el ciclo de desarrollo de software y de las primeras en llevarse a cabo.
- Mejora la calidad del software: La calidad en el software tiene que ver con cumplir un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).

- Mejora la comunicación entre equipos: La especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.

- Evita rechazos de usuarios finales: La ingeniería de requerimientos obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto. (Herrera J, 2003)

1.5 Metodologías de Desarrollo de Software.

Según Piattini (1996), no hay un consenso entre los autores sobre el concepto de metodología, y por lo tanto no existe una definición universalmente aceptada. Sí hay un acuerdo en considerar a la metodología como *"un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software".*

Maddison (1983) define metodología como un *"conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información".*

Por lo tanto, una metodología es un conjunto de componentes que especifican:

- Cómo se debe dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué salidas se producen y cuándo se deben producir.
- Qué restricciones se aplican.
- Qué herramientas se van a utilizar.
- Cómo se gestiona y controla un proyecto.

Generalizando, Piattini llega a la definición de metodología de desarrollo como *"un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software".* Normalmente consistirá en fases o etapas descompuestas en subfases, módulos, etapas, pasos, etc. Esta descomposición ayuda a los desarrolladores en la elección de las técnicas a utilizar en cada estado del proyecto, facilitando la planificación, gestión, control y evaluación de los proyectos.

Sintetizando lo anterior, el autor dice que: *"una metodología representa el camino para desarrollar software de una manera sistemática".* (Ver Anexo #4) (Cataldi, 2000)

La necesidad de una metodología de desarrollo

Para desarrollar un proyecto de software es necesario establecer un enfoque disciplinado y sistemático. Las metodologías de desarrollo influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida.

Las metodologías persiguen tres necesidades principales:

- Mejores aplicaciones, tendientes a una mejor calidad, aunque a veces no es suficiente.
- Un proceso de desarrollo controlado, que asegure uso de recursos apropiados y costo adecuado.
- Un proceso estándar en la organización, que no sienta los cambios del personal.

Las metodologías a veces tienen diferentes objetivos, pero los más representativos pueden ser:

- Brindar un método sistemático, de modo de controlar el progreso del desarrollo.
- Especificar los requerimientos de un software en forma apropiada.
- Construir productos bien documentados y de fácil mantenimiento.
- Ayudar a identificar las necesidades de cambio lo más pronto posible.
- Proporcionar un sistema ágil que satisfaga a todas las personas involucradas.

Los procesos se descomponen hasta el nivel de tareas o actividades elementales, donde cada tarea está identificada por un procedimiento que define la forma de llevarla a cabo. Para aplicar un procedimiento se pueden usar una o más técnicas. Estas pueden ser gráficas con apoyos textuales, formales y determinan el formato de los productos resultantes en la tarea.

Para llevar a cabo las tareas se pueden usar herramientas software que automatizan la aplicación en determinado grado. (Cataldi, 2000)

Diferencias entre Metodología, Ciclo de Vida y Método.

- Una Metodología puede seguir uno o varios modelos de Ciclo de Vida.
- Un Ciclo de Vida indica qué obtener, pero no cómo.

- Una Metodología es un concepto más amplio que Método.
  - Se puede considerar como un conjunto de métodos.
  - Una metodología puede englobar un conjunto de métodos (de análisis, diseño, programación, etc.) para abarcar el ciclo de vida completo. (García Rubio, y otros)

Checkland establece la siguiente diferencia entre método y metodología:"la esencia de una metodología -en forma opuesta a lo que ocurre en un método o técnica- es que ofrece un conjunto de pautas o principios que en cualquier instancia específica pueden ser ajustadas tanto a las características de la situación en la cual debe ser aplicada como a las personas que usan el enfoque. Es tal la variedad de situaciones problemáticas humanas que no habrá ningún enfoque para solución de problemas que pueda ser reducido a una fórmula estándar y manejar aún toda la riqueza de las situaciones en particular". (Checkland, 1989)

Características Deseables de una Metodología.

- ✓ Reglas predefinidas.
- ✓ Determinación de los pasos del ciclo de vida.
- ✓ Verificaciones en cada etapa.
- ✓ Planificación y control.
- ✓ Comunicación efectiva entre desarrolladores y usuarios.
- ✓ Flexibilidad: aplicación en un amplio espectro de casos.
- ✓ De fácil comprensión.
- ✓ Soporte de herramientas automatizadas.
- ✓ Que permita definir mediciones que indiquen mejoras.
- ✓ Que permita modificaciones.
- ✓ Que soporte reusabilidad del software. (Cataldi, 2000)

Ventajas de tener una metodología.

Mejora de los procesos de desarrollo.

- Todas las personas del proyecto trabajan bajo un marco común.

- Estandarización de conceptos, actividades y nomenclaturas.
- Actividades de desarrollo apoyadas por procedimientos y guías.
- Resultados del desarrollo predecibles.
- Uso de herramientas de ingeniería software.
- Planificación de actividades en base a un conjunto de tareas definidas y a la experiencia en otros proyectos.
- Recopilación de mejores prácticas para proyectos futuros.

Mejora de los productos.

- Se asegura que los productos cumplen con los objetivos de calidad propuestos.
- Detección temprana de errores.
- Se garantiza la trazabilidad de los productos a lo largo del desarrollo.

Mejora de las relaciones con el cliente.

- El cliente percibe el orden en nuestros procesos.
- Facilita al cliente el seguimiento de la evolución del proyecto.
- Se establecen mecanismos para asegurar que los productos desarrollados cumplen con las expectativas del cliente. (López Barrio, 2005)

1.5.1 Metodologías Tradicionales.

Las metodologías tradicionales se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado. Sin embargo, es posible por medio de la recolección y estudio de métricas de desarrollo lograr realizar estimaciones acertadas en contextos de desarrollo repetibles. (Arboleda Jiménez, 2005)

Metodologías tradicionales basadas en ciclos de vida.

- En cascada: su característica principal es que no comienza una fase hasta que no ha terminado la anterior.
- Prototipado rápido: consiste en iterar en la fase de análisis tantas veces como sea necesario, mostrando prototipos al usuario para que pueda indicar de forma más eficiente los requisitos del sistema.
- Evolutivo: se asume que los requisitos pueden cambiar en cualquier momento del ciclo de vida y no solo en la etapa de análisis. Se repite todo el ciclo para desarrollar nuevas versiones de todo el sistema.
- Incremental: se desarrolla un subsistema para satisfacer un subconjunto de los requisitos especificados y en posteriores versiones se añaden nuevas funcionalidades que satisfagan más requisitos.
- En espiral: toma las ventajas del modelo de desarrollo en cascada y el de prototipos, incorporando el concepto de análisis de riesgo en cada iteración.

Metodologías orientadas a procesos (basadas en mejores prácticas).

- Rational Unified Process (RUP): define un ciclo de vida iterativo priorizando el uso de lenguajes de modelado, casos de uso y centrado en la arquitectura.
- Microsoft Solution Framework (MSF): metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso. Se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. (Ver Anexo #5) (López Barrio, 2005)

Rational Unified Process (RUP).

Es un proceso de Ingeniería del Software que proporciona una visión disciplinada para la asignación de tareas y responsabilidades en las organizaciones de desarrollo de software.

RUP integra un conjunto de "**buenas prácticas"** para el desarrollo de software en un marco de procesos válido para un rango amplio de tipos de proyectos y organizaciones. Las principales buenas prácticas cubiertas son:

· Desarrollo iterativo.

· Gestión de requisitos.

· Uso de arquitecturas basadas en componentes.

· Uso de técnicas de modelado visual.

· Verificación continua de la calidad.

· Gestión y control de cambios. (Palacio, 2005)

Desarrollar software iterativamente.

– Los malos entendidos se detectan al inicio.

– Facilita el feedback para elicitar requerimientos.

– El equipo se concentra en lo esencial.

– Evaluaciones continuas dan un estado más exacto del proyecto.

– Las inconsistencias entre análisis, diseño e implementación se detectan tempranamente.

– Permite una mejor gerencia de riesgos.

– Se aplican las lecciones aprendidas.

– El cliente ve resultados a corto plazo.

Gerenciar los requerimientos.

– Las comunicaciones se basan en requerimientos definidos.

– Los requerimientos se priorizan y filtran.

– Se hace posible una evaluación de la funcionalidad deseada.

– Las inconsistencias se detectan tempranamente.

– Se puede contar con un repositorio de requerimientos.

Usar arquitecturas basadas en componentes.

– Conlleva a la modularidad.

– Contribuye con el control de cambios.

– Existen herramientas que soportan la construcción basada en componentes.

– Arquitecturas libres de errores.

Modelar software visualmente.

– Disminuye la ambigüedad.

– Los detalles no necesarios se ocultan.

– Se identifican arquitecturas no modulares e inflexibles.

– Un gráfico dice más que mil palabras.

Verificación continua de la calidad.

– Se hace una evaluación objetiva del estatus del proyecto.

– Se detecta inconsistencias entre análisis, diseño e implementación.

– Las pruebas se concentran en los aspectos de mayor riesgo.

– Los defectos se identifican claramente, se reducen los costos de su depuración.

Gerenciar los cambios.

– Las solicitudes de cambios se logran con buena comunicación.

– Las tasas de cambios arrojan información sobre el desempeño del proceso.

– La propagación del cambio es controlada.

– Se mantiene una arquitectura robusta.

El proceso propuesto por RUP posee dos dimensiones:

– la primera, representa el aspecto dinámico del proceso, y está expresado en términos de ciclos, fases, iteraciones e hitos;

– la segunda, representa el aspecto estático, que se describe en términos de componentes, actividades, flujos de trabajo, artefactos, y actores. (Mendoza)

Como se mencionó anteriormente en su visión estática, el modelo RUP está compuesto por:

- Roles: analista de sistemas, diseñador, diseñador de pruebas, roles de gestión y roles de administración.
- Actividades: RUP determina el trabajo de cada rol a través de actividades. Cada actividad del proyecto debe tener un propósito claro, y se asigna a un rol específico. Las actividades pueden tener duración de horas o de algunos días; y son elementos base de planificación y progreso.

- Artefactos: Son los elementos de entrada y salida de las actividades. Son productos tangibles del proyecto. Las cosas que el proyecto produce o usa para componer el producto final (modelos, documentos, código, ejecutables…)

- Disciplinas: son "contenedores" empleados para organizar las actividades del proceso. RUP comprende 6 disciplinas técnicas y 3 de soporte:

Técnicas: modelado del negocio, requisitos, análisis y diseño, implementación, pruebas y desarrollo.

Soporte: gestión de proyecto, gestión de configuración y cambio, y entorno.

- Flujos de trabajo: son el "pegamento" de los roles, actividades, artefactos y disciplinas, y constituyen la secuencia de actividades que producen resultados visibles.

En su visión dinámica, la visión de la estructura del ciclo de vida RUP se basa en un desarrollo iterativo, jalonado por hitos para revisar el avance y planear la continuidad o los posibles cambios de rumbo.

RUP divide el proceso de desarrollo en ciclos, teniendo una versión del producto al final de cada ciclo.

Cuatro son las fases que dividen el ciclo de vida de un proyecto RUP, las cuales concluyen con un hito bien definido donde deben tomarse ciertas decisiones:

1.- Inicio. Es la fase de la idea, de la visión inicial de producto, su alcance. El esbozo de una arquitectura posible y las primeras estimaciones. Concluye con el

"hito de objetivo".

2.- Elaboración. Comprende la planificación de las actividades y del equipo necesario. La especificación de las necesidades y el diseño de la arquitectura.

Termina con el "hito de Arquitectura".

3.- Construcción. Desarrollo del producto hasta que se encuentra disponible para su entrega a los usuarios. Termina con el "hito del inicio de la capacidad operativa".

4.- Transición. Traspaso del producto a los usuarios. Incluye: manufactura, envío, formación, asistencia y el mantenimiento hasta lograr la satisfacción de los usuarios. Termina con el "hito de entrega del producto".

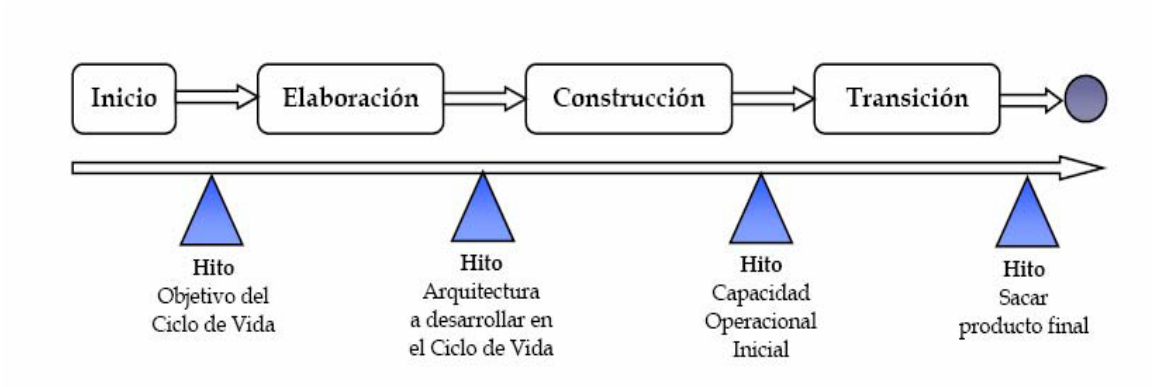


Figura 1: Fases e Hitos de RUP.

Cada fase se subdivide a su vez en iteraciones, siendo el número de iteraciones en cada una de ellas variable. (Palacio, 2005)

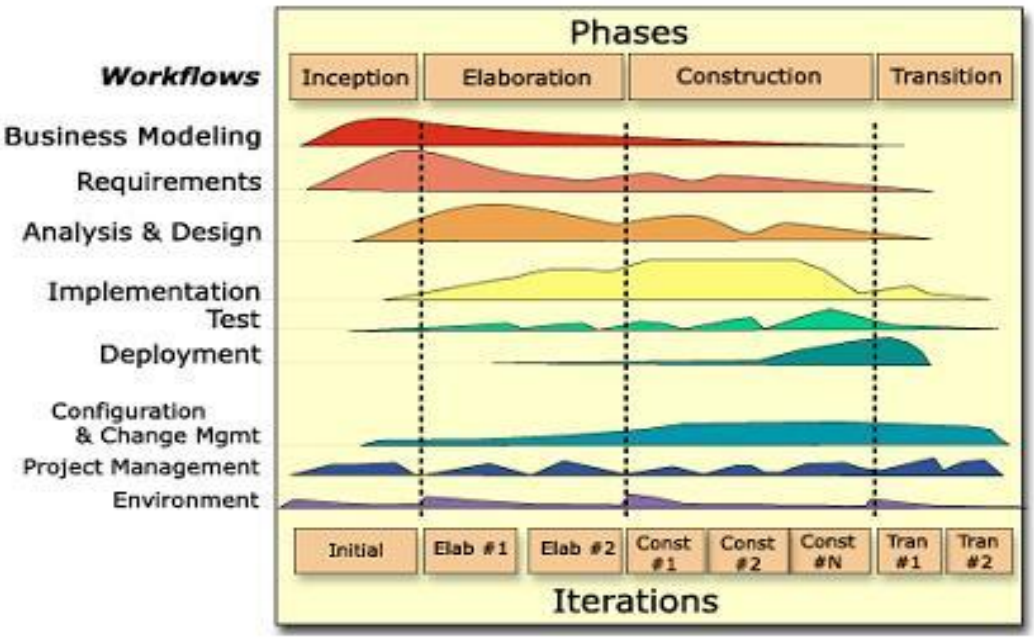


Figura 2: Fases e Iteraciones de la Metodología RUP.

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

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (Mendoza Sanchez, 2004)

El Proceso Unificado está dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Un proceso dirigido por casos de uso.

Los casos de uso han sido adoptados casi universalmente para la captura de requisitos de sistemas software en general, y de sistemas basados en componentes en particular, pero los casos de uso son mucho más que una herramienta para capturar requisitos. Dirigen el proceso de desarrollo en su totalidad.

¿Por qué casos de uso?

Existen varios motivos por los cuales los casos de uso son buenos, se han hecho populares y se han adoptado universalmente. Las razones fundamentales son:

- Proporcionan un medio sistemático e intuitivo de capturar requisitos funcionales centrándose en el valor añadido para el usuario.

- Dirigen todo el proceso de desarrollo debido a que la mayoría de las actividades como el análisis, diseño y prueba se llevan a cabo partiendo de los casos de uso. El diseño y la prueba pueden también planificarse y coordinarse en términos de casos de uso. Esta característica es aun más evidente cuando la arquitectura se ha estabilizado en el proyecto, después del primer conjunto de iteraciones.

Un proceso centrado en la arquitectura.

Los casos de uso solamente no son suficientes. Se necesitan más cosas para conseguir un sistema de trabajo. Esas “cosas” son la arquitectura. Se puede pensar que la arquitectura de un sistema es la visión común en la que todos los empleados (desarrolladores y otros usuarios) deben estar de acuerdo, o como poco, deben aceptar. La arquitectura da una clara perspectiva del sistema completo, necesaria para controlar el desarrollo.

¿Por qué es necesaria la arquitectura?

Un sistema software grande y complejo requiere de una arquitectura para que los desarrolladores puedan progresar hasta tener una visión común.

Se necesita una arquitectura para:

- Comprender el sistema.
- Organizar el desarrollo.
- Fomentar la reutilización.
- Hacer evolucionar el sistema.

Un proceso iterativo e incremental.

Como se indicó anteriormente, dos de las tres claves del Proceso Unificado son que el proceso de desarrollo software debería estar dirigido por los casos de uso y centrado en la arquitectura. Estos aspectos tienen un impacto técnico evidente sobre el producto del proceso. Conseguir el equilibrio correcto entre los casos de uso y la arquitectura es algo muy parecido al equilibrado de la forma y la función en el desarrollo de cualquier producto. Se consigue con el tiempo. Durante el más corto proceso de desarrollo de software, los desarrolladores elaboran conscientemente este equilibrio (entre casos de uso y arquitectura) a lo largo de una serie de iteraciones. Por tanto, la técnica de desarrollo iterativo e incremental constituye el tercer aspecto clave del Proceso Unificado.

La tercera clave proporciona la estrategia para desarrollar un producto software en pasos pequeños manejables:

- Planificar un poco.
- Especificar, diseñar, e implementar un poco.
- Integrar, probar, y ejecutar un poco cada iteración.

¿Por qué un desarrollo iterativo e incremental?

En dos palabras: para obtener un software mejor. Dicho con unas cuantas palabras más, para cumplir los hitos principales y secundarios con los cuales se controla el desarrollo. Y dicho con algunas palabras más:

- ✓ Para tomar las riendas de los riesgos críticos y significativos desde el principio.
- ✓ Para poner en marcha una arquitectura que guíe el desarrollo del software.

✓ Para proporcionar un marco de trabajo que gestione de mejor forma los inevitables cambios en los requisitos y en otros aspectos.

✓ Para construir el sistema a lo largo del tiempo en lugar de hacerlo de una sola vez cerca del final, cuando el cambiar algo se ha vuelto costoso.

✓ Para proporcionar un proceso de desarrollo a través del cual el personal puede trabajar de manera más eficaz. (Jacobson, y otros, 2004)

Es importante destacar que la Metodología RUP es más adaptable para proyectos de largo plazo. (Mendoza Sanchez, 2004)

En resumen, la meta de RUP es: (Kruchten, 1999)

• asegurar la producción de un software de alta calidad que reúna las necesidades de los usuarios finales dentro de un plan y un presupuesto predecible;

• proveer un enfoque disciplinado para asignar tareas y responsabilidades dentro del desarrollo del sistema;

• proveer un camino metódico, sistemático para desarrollar, diseñar y validar una arquitectura;

• reducir en gran medida el riesgo que representa la construcción de sistemas complejos, porque evoluciona de forma incremental partiendo de sistemas más pequeños en los que ya se tiene confianza. (Mendoza)

1.5.2 Metodologías Ágiles.

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó La Alianza Ágil (The Agile Alliance), una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las

organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía "ágil". (Ver Anexo #6)

Principales Metodologías Ágiles.

- XP (Extreme Programming): Utilizada para proyectos de corto plazo. Consiste en desarrollos rápidos e iterativos, cuya particularidad es tener como parte del equipo al usuario final. (Ver Anexo #7) (López Barrio, 2005)

- SCRUM. Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

- Crystal Methodologies. Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

- Dynamic Systems Development Method (DSDM). Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

- Adaptive Software Development (ASD). Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y

aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

- Feature -Driven Development (FDD). Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.

- Lean Development (LD). Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios. (Canós, y otros, 2003)

1.5.3 Diferencias entre Metodologías Ágiles y Tradicionales.

Las metodologías tradicionales se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Muy efectivas y necesarias en proyectos grandes.

Las metodologías ágiles, dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Con cortos documentos centrados en lo esencial. (Marcela, 2007)

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios

Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1: Comparación entre las Metodologías Ágiles y las Metodologías Tradicionales.

En fin las metodologías tradicionales hacen énfasis en la planeación y las metodologías agiles en la adaptabilidad del proceso. (Arboleda Jiménez, 2005)

1.6 Herramientas de Desarrollo de Software.

La industria de computadoras ha desarrollado un soporte automatizado para el desarrollo y mantenimiento de software. Este es llamado Computer Aided Software Engineering (CASE). La mejor razón para la creación de estas herramientas fue el incremento en la velocidad de desarrollo de los sistemas.

La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las Herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software.

1.6.1 ¿Qué son las Herramientas CASE?

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

CASE se define también como:

- Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.
- La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas.
- Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales. (Instituto, 1999)

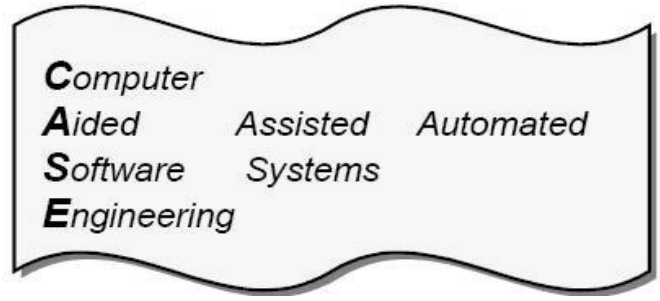


Figura 3: Variaciones en el significado de CASE.

1.6.2 Clasificación de las Herramientas CASE.

No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase determinada. Podrían clasificarse atendiendo a:

- Las plataformas que soportan.

- Las fases del ciclo de vida del desarrollo de sistemas que cubren.

- La arquitectura de las aplicaciones que producen.

- Su funcionalidad.

CASE es una combinación de herramientas software (aplicaciones) y de metodologías de desarrollo:

1. Las herramientas permiten automatizar el proceso de desarrollo del software.

2. Las metodologías definen los procesos automatizar.

Una primera clasificación del CASE es considerando su amplitud:

TOOLKIT: es una colección de herramientas integradas que permiten automatizar un conjunto de tareas de algunas de las fases del ciclo de vida del sistema informático: Planificación estratégica, Análisis, Diseño, Generación de programas.

WORKBENCH: Son conjuntos integrados de herramientas que dan soporte a la automatización del proceso completo de desarrollo del sistema informático. Permiten cubrir el ciclo de vida completo. El producto final aportado por ellas es un sistema en código ejecutable y su documentación.

Una segunda clasificación es teniendo en cuenta las fases (y/o tareas) del ciclo de vida que automatizan:

UPPER CASE: Planificación estratégica, Requerimientos de Desarrollo Funcional de Planes Corporativos.

MIDDLE CASE: Análisis y Diseño.

LOWER CASE: Generación de código, test e implantación (Politécnica, 2005)

1.6.3 Ventajas que aportan las Herramientas CASE.

Las herramientas CASE permiten a las compañías desarrollar sistemas sin encarar el problema de tener cambios en las necesidades del negocio, antes de finalizar el proceso de desarrollo; además de competir más efectivamente usando estos sistemas desarrollados nuevamente para compararlos con sus necesidades de negocio actuales. En un mercado altamente competitivo, esto puede hacer la diferencia entre el éxito y el fracaso.

Las herramientas CASE también permiten a los analistas tener más tiempo para el análisis y diseño y minimizar el tiempo para codificar y probar.

La introducción de CASE integradas está comenzando a tener un impacto significativo en los negocios y sistemas de información de las organizaciones.

Con un CASE integrado, las organizaciones pueden desarrollar rápidamente sistemas de mejor calidad para soportar procesos críticos del negocio y asistir en el desarrollo y promoción intensiva de la información de productos y servicios.

Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software, algunas de ellas son:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la Base de Datos.
- Generar estructuras de código.

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad. Para conseguir estos dos objetivos es conveniente contar con una organización y una metodología de trabajo, además de la propia herramienta.

La mejora de calidad se consigue reduciendo sustancialmente muchos de los problemas de análisis y diseño, inherentes a los proyectos de mediano y gran tamaño (lógica del diseño, coherencia, consolidación, etc.). La mejora de productividad se consigue a través de la automatización de determinadas tareas, como la generación de código y la reutilización de objetos o módulos. (Instituto, 1999)

Las cuatro razones para la adopción de herramientas CASE son:

- El incremento de la productividad del analista
- La mejora de la comunicación entre analistas y usuarios
- La integración de actividades del ciclo de vida y el análisis.
- La valoración del impacto de los cambios por mantenimiento.

1.6.4 Principales Herramientas CASE.

Dentro de las herramientas de desarrollo de software se encuentran: Rational Rose, Poseidón, Visual Paradigm, Umbrello UML Modeller, ArgoUML, BOUML, Magic Draw, entre otras. Cada una de ellas

tiene ventajas y desventajas a la hora de representar gráficamente un sistema software. (Ver Anexo #8)

Visual Paradigm

Soporta hasta la fecha UML 2.0 completo, BPMN, SysML, DFD y ERD. Es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o Pdf, y permite control de versiones. Está soportada para software libre, no da problema en la instalación de sus módulos.

Tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community (que es gratuita). Facilita licencias especiales para fines académicos.

Licencia: Gratuita y Comercial.

Características:

- Producto de calidad.
- Soporta aplicaciones web.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones. (Giraldo, y otros, 2005)

1.7 Lenguajes de Modelado.

Para realizar la modelación visual de un sistema software se pueden utilizar diferentes lenguajes, entre ellos se encuentran: Lenguaje Unificado de Modelado (UML), Lenguaje de Modelado de Procesos de Negocio (BPML), etc. (Ver Anexo #9)

Lenguaje Unificado de Modelado (UML).

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e

incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema.

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes. (Jacobson, y otros, 2000)

El lenguaje UML estandariza los artefactos y la notación, pero no define un proceso oficial de desarrollo. He aquí algunas de las razones que explican esto:

1. Aumentar las probabilidades de una aceptación generalizada de la notación estándar del modelado, sin la obligación de adoptar un proceso oficial.

2. La esencia de un proceso apropiado admite mucha variación y depende de las habilidades del personal, de la razón investigación-desarrollo, de la naturaleza del problema, de las herramientas y de muchos otros factores.

Diagramas UML.

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. A continuación se nombran los mismos:

• Diagramas de Casos de Uso para modelar los procesos del negocio.

• Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.

• Diagramas de Secuencia para modelar el paso de mensajes entre objetos.

• Diagramas de Colaboración para modelar interacciones entre objetos.

• Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.

• Diagramas de Clases para modelar la estructura estática de las clases en el sistema. (Larman, 1999)

1.8 Conclusiones.

La Ingeniería de Requerimientos se centra en un aspecto fundamental: cómo definir lo que se desea producir y así minimizar los problemas relacionados por la mala gestión de los requerimientos en el desarrollo de sistemas. Para ello la IR propone una conjunto de actividades como: la identificación, el análisis y negociación de requisitos, la especificación de los mismos, el modelado de sistema, así como la validación y gestión de requisitos. Cada una de estas actividades serán realizadas durante el transcurso de este trabajo para desarrollar el modelado del Sistema de Reportes para el MENPET.

En este capítulo se ha realizado un estudio de las principales metodologías, herramientas y lenguajes existentes para el desarrollo de software, con el objetivo de seleccionar la tecnología informática adecuada y necesaria en el modelado de un sistema de reportes. Dicha tecnología será utilizada durante el desarrollo de las actividades de la IR antes mencionadas.

RUP es la metodología propuesta para desarrollar el modelado del sistema, ya que posee un alto nivel organizacional, permite la asignación de roles y responsabilidades de forma disciplinada y cada una de sus fases está bien documentada dentro del desarrollo de software. Este proceso de Ingeniería del Software aumenta la productividad de los desarrolladores mediante un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Se centra en la producción y mantenimiento de modelos del sistema más que en producir documentos. Además RUP es más adaptable para proyectos de largo plazo, dentro de sus características principales se encuentran que está dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental, las mismas son puntos claves en el desarrollo de software. Es una guía de cómo utilizar de manera efectiva *UML*, un lenguaje que permite:

- Modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).Puede conectarse con lenguajes de programación (ingeniería directa e inversa).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Cubre las cuestiones relacionadas con el tamaño, propias de los sistemas complejos y críticos.

- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

Como herramienta CASE se propone la utilización de Visual Paradigm pues ofrece: unas características gráficas muy cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML, un diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad, el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación, capacidades de ingeniería directa (versión profesional) e inversa, modelo y código que permanece sincronizado en todo el ciclo de desarrollo, disponibilidad de múltiples versiones para cada necesidad, disponibilidad de integrarse en los principales IDEs, disponibilidad en múltiples plataformas. Visual Paradigm permite también la generación de código PHP, que es el lenguaje de programación con el que se desarrollará el sistema de reportes para el MENPET.

Capítulo 2

Modelado del Sistema de Reportes para el MENPET.

2.1 Introducción.

Es necesario antes de desarrollar el Sistema de Reportes para el Ministerio del Poder Popular para la Energía y Petróleo de la República Bolivariana de Venezuela, comprender su estructura y dinámica, comprender sus problemas actuales e identificar las mejoras potenciales, analizar el estado del negocio y determinar si los procesos están claramente definidos para así realizar el modelo del negocio o dominio según corresponda. Una vez en este punto, se hace necesario comenzar a definir que es lo que debe hacer el sistema, siendo obligatorio ir a la captura de los requerimientos que este debe cumplir y después pasar al siguiente paso de crear el modelo del sistema. La validación de los requerimientos es muy importante para evitar resultados inesperados, costos excesivos y pérdida de tiempo por lo que se propone un prototipo del sistema no funcional, lo cual constituye una técnica de validación. Además en este capítulo se mencionará el modelo de proceso de Ingeniería de Requerimientos utilizado en el desarrollo del software para el MENPET, las técnicas de la IR aplicadas en dicho proceso y los patrones de casos de uso usados para estructurar el diagrama de casos de uso del sistema.

2.2 Proceso de Ingeniería de Requerimientos.

El proceso de Ingeniería de Requerimientos es un conjunto estructurado de actividades, mediante las cuales se obtiene, se valida y mantiene el Documento de Especificación de Requerimientos (ESRE).

Cualquier tarea en donde el resultado sea importante, se puede realizar de mejor manera al utilizar algún tipo de proceso ordenado. Para obtener este orden, se diseñan procesos basándose en algún modelo que guíe a la hora de diferenciar y secuenciar las actividades. (Davyt Dávila, 2001)

El modelo de proceso de IR utilizado para el desarrollo del Sistema de Reportes fue el Modelo Espiral pues proporciona el potencial para el desarrollo rápido de versiones incrementales del software, le permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Este modelo mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo integra al marco de trabajo iterativo que refleja de forma más realista el

mundo real, además incorpora el concepto de análisis de riesgo en cada iteración y permite comenzar el proyecto con un alto grado de incertidumbre. (Ver Anexo #10)

2.2.1 Técnicas de la Ingeniería de Requerimientos.

Existen distintas técnicas que se utilizan para llevar a cabo cada una de las actividades del proceso de Ingeniería de Requerimientos. (Ver Anexo #11)

A continuación se muestra una tabla con las técnicas aplicadas durante el proceso de desarrollo del Sistema de Reportes:

Técnicas	Identificación de Requisitos	Análisis y Negociación de Requisitos	Especificación de Requisitos	Modelado del Sistema	Validación de Requisitos	Gestión de Requisitos
Entrevistas	x	x				
Sistemas existentes	x	x				
Tormenta de Ideas	x					
Glosario	x	x	x	x	x	x
ESRE	x	x	x	x	x	x
Caso de Uso			x	x	x	x
Casa de Calidad o QFD					x	
Prototipos					x	

Tabla 2: Técnicas de la IR aplicadas durante el proceso de desarrollo.

2.3 La comprensión del contexto del sistema mediante un modelo del dominio.

2.3.1 ¿Por qué un modelo del dominio?

En el Ministerio del Poder Popular para la Energía y Petróleo de la República Bolivariana de Venezuela no se logra determinar el proceso del negocio con fronteras bien establecidas donde se logren ver claramente, quienes son las personas que lo inician, quienes son los beneficiados con cada uno de estos procesos, pero además quienes son las personas que desarrollan las actividades en cada uno de estos procesos. (Ver Anexo #12)

2.3.2 Modelo del dominio del MENPET.

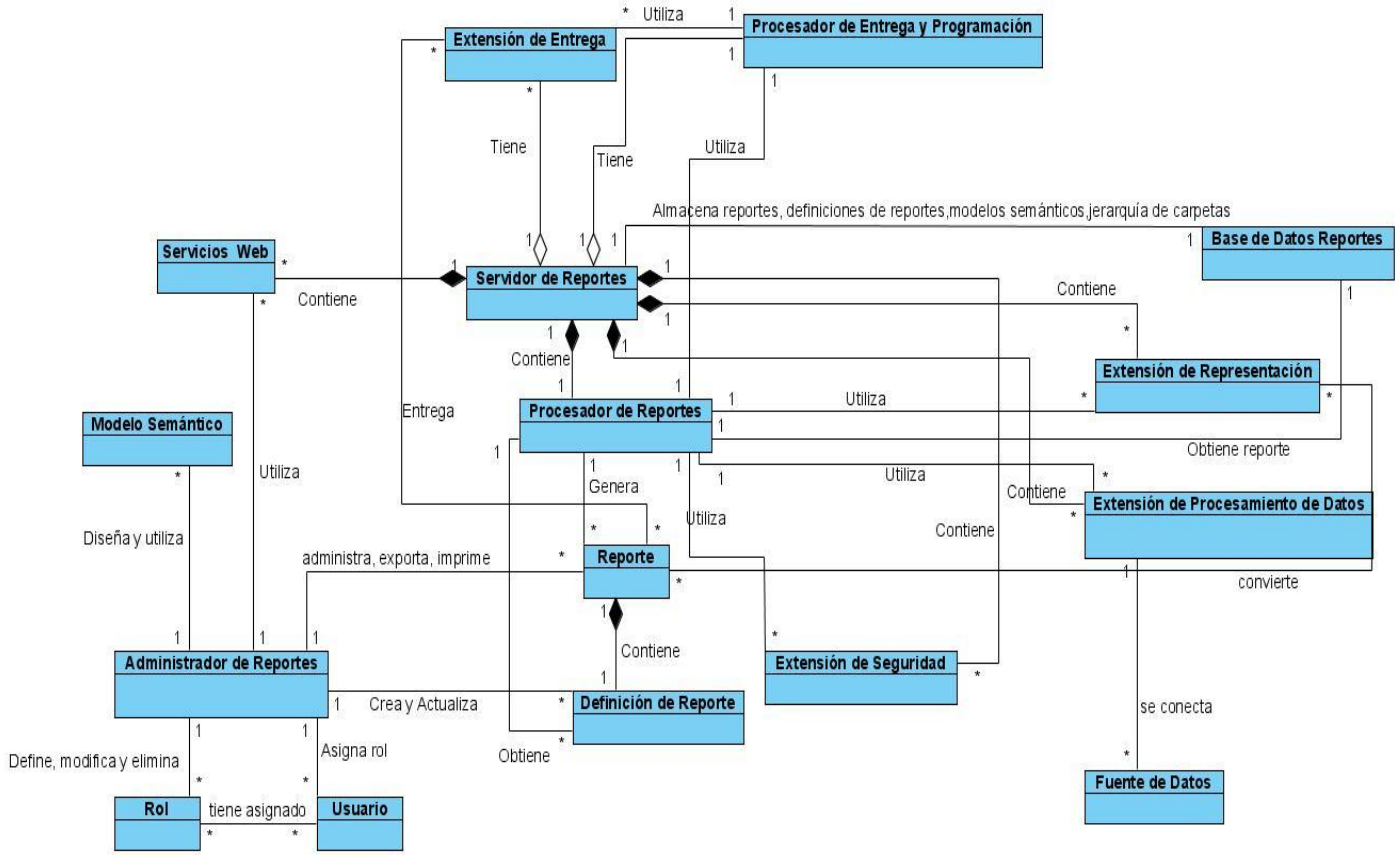


Figura 4: Modelo del dominio.

2.3.3 Descripción de los conceptos del modelo del dominio.

Servicios Web

El objeto Servicios Web es el encargado de proveer completo acceso a las funcionalidades del Servidor de Reportes. Este objeto actúa como una interfaz de comunicación entre el cliente y el Servidor de Reportes.

Procesador de Reportes

El objeto Procesador de Reportes es el encargado de procesar reportes y definiciones de reporte. El mismo genera los reportes del sistema, para ello obtiene la definición del reporte, recibe de la Extensión de Procesamiento de Datos la información de la fuente de datos, y organiza la estructura del reporte la cual es enviada posteriormente al objeto Extensión de Representación para que sea visualizado al usuario.

Extensión de Seguridad

El objeto Extensión de Seguridad es el encargado como su nombre lo indica de la seguridad de sistema. Mediante el objeto se puede autenticar y autorizar a usuarios o grupos a utilizar los diferentes reportes publicados en el Servidor de Reportes.

Extensión de Representación

El objeto Extensión de Representación es el facultado para visualizar el reporte generado al usuario. Para realizar este proceso se convierten los datos y la información de diseño del Procesador de Reportes al formato especificado por el usuario y después se visualiza. Entre los formatos a visualizar se pueden encontrar archivos HTML, XLS (Excel), PDF, XML, entre otros.

Extensión de Procesamiento de Datos

El objeto Extensión de Procesamiento de Datos es el encargado de la capa de acceso a datos del sistema. Este objeto se utiliza para comunicarse a las diferentes fuentes de datos y obtener la información necesaria para conformar los reportes. Posee funcionalidades para conectarse a la fuente de datos, realizar consultas parametrizadas o no, verificar conexiones a una fuente de datos específica, entre otras.

Procesador de Entrega y Programación

El objeto Procesador de Entrega y Programación es el encargado de ejecutar operaciones programadas anteriormente por el usuario trabajando en conjunto con la Extensión de Entrega. Este se ejecuta en el Servidor de Reportes y utiliza un temporizador para generar eventos programados.

Extensión de Entrega

El objeto Extensión de Entrega es el encomendado para realizar la entrega de los diferentes reportes generados en los destinos predefinidos por el cliente. Los diferentes destinos pueden ser a través de correo o una carpeta compartida en la red. Este objeto es utilizado por el Procesador de Entrega y Programación después que haya ejecutado una operación programada.

Definición de Reporte

El objeto Definición de Reporte es la entidad encargada de almacenar la estructura, semántica e interpretación de un reporte.

Reporte

El objeto Reporte es la entidad encargada de almacenar el reporte generado de forma completa. Este se puede encontrar en diferentes formatos, ya sea PDF, HTML, XLS, entre otros. El Reporte está asociado a una definición, la cual tiene que validar y cumplir con sus estructuras.

Modelo Semántico

El objeto Modelo Semántico se utiliza para almacenar la estructura semántica por la que está compuesta una base de datos. Este objeto es sumamente importante para ganar en rendimiento y seguridad, ya que no sería necesario realizar consultas a la base de datos que comprometan la información almacenada en esta.

Usuario

El objeto Usuario es la entidad de trabajar con los usuarios del sistema. En ella se recogen los datos necesarios para un correcto funcionamiento del sistema y que no se vea comprometida su seguridad.

Rol

El objeto Rol es la entidad encargada de agrupar los diferentes usuarios en roles para lograr una mejor administración del sistema. El objeto Rol puede contener un grupo de Usuarios.

Administrador de Reportes

El objeto Administrador de Reportes es utilizado para gestionar los reportes, las carpetas, los modelos semánticos, configurar la seguridad basada en roles, crear la programación y entrega de reportes.

Base de Datos Reportes

El objeto Base de Datos Reportes se utiliza para almacenar las definiciones de reportes, los reportes, los modelos semánticos, así como la jerarquía de carpetas.

Fuente de Datos

El objeto Fuente de Datos se utiliza para almacenar una conexión a una Base de datos, la cual contiene los detalles necesarios para comunicarse con la Base de Datos, como el driver o la localización física.

Servidor de Reportes

El objeto Servidor de Reportes es el responsable de almacenar todas las propiedades, los objetos y los metadatos en la Base de Datos Reportes. Los datos almacenados incluyen reportes publicados, definiciones de reporte y la jerarquía de carpetas que proporciona el direccionamiento de todos los elementos que administra el servidor de reportes.

2.4 Modelo del sistema.

2.4.1 Requisitos Funcionales.

R1 Gestionar Carpetas.

R1.1 Crear carpetas.

R1.2 Modificar carpetas.

R1.3 Eliminar carpetas.

R2 Gestionar rol a nivel de elementos.

R2.1 Crear rol a nivel de elementos.

R2.2 Modificar rol a nivel de elementos.

R2.3 Eliminar rol a nivel de elementos.

R3 Gestionar rol a nivel de sistema.

R3.1 Crear rol a nivel de sistema.

R3.2 Modificar rol a nivel de sistema.

R3.3 Eliminar rol a nivel de sistema.

R4 Gestionar Suscripciones Individuales.

R4.1 Crear suscripción Individual.

R4.2 Modificar suscripción Individual.

R4.3 Eliminar suscripción Individual.

R5 Actualizar y Mantener activa la cola de eventos y notificaciones.

R6 Entregar reporte.

R6.1 Entregar los reportes por e-mail.

R6.2 Entregar los reportes en recursos compartidos en la red.

R7 Gestionar Suscripciones.

R7.1 Modificar suscripciones.

R7.2 Eliminar suscripciones.

R8 Cambiar Fuente de Datos de reporte.

R8.1 Cambiar a Fuente de Datos Compartida.

R8.2 Cambiar a Fuente de Datos Personalizada.

R9 Gestionar asignación de rol.

R9.1 Crear asignación de rol.

R9.2 Modificar asignación de rol.

R9.3 Eliminar asignación de rol.

R10 Gestionar asignación de rol de sistema.

R10.1 Crear asignación de rol de sistema.

R10.2 Modificar asignación de rol de sistema.

R10.3 Eliminar asignación de rol de sistema.

R11 Gestionar reportes caché.

R11.1 Crear reportes caché.

R11.2 Eliminar reportes caché.

R11.3 Visualizar reportes caché.

R12 Administrar Historial.

R13 Descargar definición de reporte.

R14 Actualizar definición de reporte.

R15 Mover reporte.

R16 Gestionar reporte.

R16.1 Modificar reporte.

R16.2 Visualizar reporte.

R16.2.1 Generar un reporte.

R 16.2.1.1 Recuperar la Definición del Reporte.

R16.2.1.2 Ejecutar una consulta.

R16.2.1.3 Obtener los datos para conformar el reporte.

R16.2.1.4 Iterar en un conjunto de filas y recuperar datos.

R16.2.1.5 Extraer metadatos.

R16.2.1.6 Crear el reporte.

R16.2.2 Devolver un reporte creado que se encuentra en caché.

R16.3 Eliminar reporte.

R17 Organizar o agrupar los datos de un reporte visualizado.

R18 Crear reporte enlace.

R19 Autenticar usuario.

R19.1 Permitir configurar el sistema para hacer uso de los sistemas de autenticación existentes en el MENPET (mediante tablas o formularios).

R20 Imprimir reporte.

R20.1 Configurar el reporte para imprimir.

R21 Gestionar fuentes de datos.

R21.1 Crear fuente de datos.

R21.2 Modificar fuente de datos.

R21.3 Eliminar fuente de datos.

R22 Exportar reporte.

R22.1 Exportar a Excel (.xls).

R22.2 Exportar a Word (.doc).

R22.3 Exportar a PDF (.pdf).

R22.4 Exportar a texto plano.

R22.5 Exportar a HTML.

R22.6 Exportar a XML.

R22.7 Exportar el reporte como Imagen (.jpg, .png).

R23 Definir Reporte.

R 23.1 Definir Datos.

R23.1.1 Analizar una consulta y devolver una lista de nombres de campo.

R23.1.2 Pasar parámetros a una consulta.

R23.1.3 Pasar parámetros con varios valores a una consulta.

R23.1.4 Obtener los parámetros que necesitan ser pasados a una consulta.

R23.1.5 Obtener los campos utilizados para la agrupación de los datos.

R23.1.6 Obtener los campos utilizados para la ordenación de los datos.

R23.1.7 Seleccionar la fuente de datos.

R23.1.8 Conectar con la fuente de datos.

R23.2 Diseñar Reporte.

R23.2.1 Incluir gráficas en los reportes.

R23.3 Mostrar Vista Previa

R24 Gestionar Modelos Semánticos

R24.1 Crear modelo Semántico.

R24.2 Eliminar Modelo Semántico.

R25 Subir archivo.

R26 Hacer schedules.

2.4.2 Requisitos No Funcionales.

Usabilidad

✓ Las ventanas del sistema contienen claro y bien estructurado los datos, y al mismo tiempo permiten la interpretación correcta e inequívoca de la información.

✓ El diseño de la interfaz de usuario está orientado a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.

✓ El sistema debe ser intuitivo y tener un alto nivel de usabilidad permitiendo que usuarios sin mucho conocimiento informático, en aproximadamente 30 días puedan explotarlo al 100%.

Sistema Intuitivo

A pesar de que la mayoría de los usuarios que interactuarán con el sistema son programadores, informáticos o usuarios con experiencia, este debe ser lo suficientemente intuitivo y fácil de usar como para que cualquier persona con conocimientos mínimos de informática en un mes de entrenamiento y familiarización con la aplicación sea capaz de explotarla al máximo.

✓ El sistema será puesto en explotación en el MENPET, por lo que se necesita utilizar en el diseño del mismo un lenguaje que resulte familiar al personal que interactuará con él.

Rendimiento

✓ La aplicación debe estar concebida para el consumo mínimo de recursos.

✓ El sistema debe ser rápido y ofrecer tiempos de respuesta relativamente bajos para mantener la agilidad en todos los servicios informáticos que se brindan en el MENPET.

✓ El sistema debe permitir la concurrencia.

Concurrencia

Debido a las características del lugar donde será implantado el sistema, no será necesario soportar una alta concurrencia, pero si es necesario que puedan estar usando el sistema decenas de usuarios en la generación de reportes simultáneamente sin que se ralentice el proceso o se caiga el servicio.

Software

✓ El sistema debe correr y brindar el servicio desde un servidor con Sistema Operativo libre. (Debian GNU/Linux, rama Stable).

✓ El sistema debe utilizar PostgreSQL versión 8.2 o superior como SGBD.

✓ El sistema debe brindar servicio a través del servidor Web Apache2.

✓ En las computadoras de los clientes solo se requiere de un navegador (Mozilla Firefox o Internet Explorer).

Apariencia o interfaz externa

✓ Por el uso diario y constante que tendrá el software, la interfaz debe ser agradable, que favorezca el estado de ánimo del cliente y que combine correctamente los colores, tipo de letra y tamaño y que los iconos estén en correspondencia con lo que representan.

✓ Deben utilizarse plantillas con un mismo estilo para que el usuario no se sienta perdido dentro de la aplicación.

✓ La aplicación contendrá distintas interfaces que servirán para el intercambio de información con la misma, ya sea directamente con usuarios o con otros programas o sistemas software.

Interfaces de Usuario

Las interfaces de usuario que deben ser implementadas por el software serán interfaces Web, la aplicación deberá ser accesible remotamente a través de la web utilizando navegadores estándares como Mozilla Firefox o Internet Explorer.

Interfaces de Comunicación

Las principales interfaces de comunicación que brindará el sistema serán servicios web (WebServices), que serían utilizados por otras aplicaciones para instanciar el Sistema de Reportes. El Sistema de Reportes se comunicará con los gestores de bases de datos mediante el protocolo TCP/IP, mientras que los clientes accederán al mismo vía HTTP/HTTPS.

Seguridad

✓ Chequear si el usuario que está accediendo al sistema está autenticado y brindarle servicio de autenticación.

✓ Permitir que cuando se borre cualquier documento o información pueda existir una opción de advertencia antes de realizar la acción.

✓ Se establecerá la existencia de distintos roles que establezcan las acciones que pueden realizar cada usuario, previendo que la información sea manejada solamente por personas autorizadas, garantizando la confidencialidad de la misma.

Confidencialidad

✓ La información manejada por el sistema está protegida de acceso no autorizado y divulgación.

Disponibilidad

✓ El sistema deberá permanecer constantemente online para garantizar el acceso a la información a los usuarios autorizados, excepto cuando sea necesario reiniciarlo o detenerlo por tareas de mantenimiento o cambio de configuración.

Disponibilidad del servicio

Debido a que el sistema mantendrá en todo momento una cola de eventos a ejecutar para la generación y envío automático de reportes, es necesario que este permanezca online constantemente para atender dichas solicitudes o alguna otra desde un usuario directamente. El sistema permanecerá fuera de servicio sólo en caso de que se estén realizando tareas de mantenimiento o de configuración.

Integridad

✓ La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción.

Restricciones de diseño e implementación

✓ El sistema debe ser implementado en lenguaje de programación PHP, versión 5.

✓ Se debe utilizar la herramienta CASE Visual Paradigm Enterprise Edition.

Estándares Aplicables

✓ XML: El sistema hará uso del lenguaje XML, como lenguaje estándar para ficheros de configuración e información guardada en bases de datos como es el caso de las Definiciones de Reportes.

Descripción del sistema propuesto.

2.4.3 Explicación del sistema.

El Sistema de Reportes para el MENPET es un sistema Web basado en la administración de reportes que combine la capacidad de un generador de reportes tradicional con la de un sistema de gestión de los mismos.

El servidor de reportes es el componente principal del Sistema de Reportes. Este se implementa como un servicio del Sistema Operativo (SO) y como un servicio Web. El servidor Web contiene servicios que constituyen un conjunto de interfaces programáticas que las aplicaciones cliente pueden utilizar para obtener acceso a servidores de reportes. El servicio del SO proporciona parámetros de inicialización, así como mantenimiento del servidor de reportes. Los servicios funcionan conjuntamente y constituyen una única instancia del servidor de reportes.

El servidor de reportes almacena todas las propiedades, los objetos y los metadatos en una base de datos de PostgreSQL. Los datos almacenados incluyen reportes publicados, definiciones de reporte y la jerarquía de carpetas que proporciona el direccionamiento de todos los elementos que administra el servidor de reportes.

A través de sus subcomponentes, el servidor de reportes procesa solicitudes de reportes y permite que los reportes estén disponibles para el acceso a petición o la distribución programada. Los subcomponentes del servidor de reportes incluyen procesadores y extensiones. Los procesadores son el núcleo del servidor de reportes. Las extensiones también son procesadores, pero realizan funciones muy específicas.

El servidor de reportes incluye dos procesadores que realizan el procesamiento de reportes previo e intermedio, así como operaciones programadas y de entrega.

El Procesador de Reportes es un componente del servidor de reportes que procesa reportes y definiciones de reporte. El procesamiento se inicia mediante solicitudes de un reporte publicado. Cuando se realiza una solicitud de procesamiento de reportes para un reporte publicado, el Procesador de Reportes obtiene la definición o el modelo de reporte de la base de datos del servidor de reportes,

inicializa parámetros y variables que se encuentran en expresiones y realiza otro procesamiento previo que prepara el reporte para datos. Entonces la extensión de procesamiento de datos se conecta con la fuente de datos y recupera los datos. El Procesador de reportes combina los datos de reporte con el diseño de reporte de la DR y representa el reporte en el formato solicitado utilizando la extensión de representación.

El Procesador de Reportes responde a tres solicitudes:

1. Solicitud de un reporte a petición: El Procesador de Reportes recupera la definición de reporte, envía la solicitud de datos a una extensión de procesamiento de datos, combina la definición de reporte con los datos, la envía a una extensión de representación y el usuario recibe el reporte representado.

2. Solicitud para generar reporte caché: El procesador de reportes recupera la DR y la combina con los datos recuperados por la extensión de procesamiento de datos, y almacena el reporte caché en la base de datos.

3. Solicitud para reporte caché: El procesador de reportes recupera el reporte caché de la caché y pasa a la extensión de presentación, el usuario recibe el reporte representado.

El Procesador de Entrega y Programación procesa reportes desencadenados a partir de una programación y los entrega a diferentes destinos. Este procesador está incluido para permitir operaciones programadas y controlar las extensiones de entrega utilizadas para insertar reportes en buzones de correo electrónico o en recursos compartidos en la red. El Procesador de entrega y programación ofrece las siguientes funcionalidades:

- Mantiene una cola de eventos y notificaciones.
- Llama al Procesador de Reportes para ejecutar reportes, procesar suscripciones o borrar reportes de la memoria caché. Todo procesamiento de reportes que es consecuencia de un evento de programación se efectúa mediante el servicio del SO del Servidor de reportes y no con el servicio Web del Servidor de Reportes.
- Llama a la extensión de entrega especificada en una suscripción para que el reporte se pueda entregar.

Otros componentes y servicios que funcionan con el Procesador de Entrega y Programación controlan otros aspectos de la operación de programación y entrega. En concreto, el Procesador de Entrega y

Programación se ejecuta en el servicio del SO del Servidor de Reportes y utiliza un temporizador como “cron” o “anacron”, que son demonios del SO Linux, para generar eventos programados. En la siguiente lista se describe el funcionamiento de las operaciones programadas en una implementación del Servidor de Reportes:

1. Las operaciones programadas se definen cuando un usuario crea una programación. En la programación se definen la fecha y hora en que se ejecutará un reporte.

2. El servidor de reportes guarda la información de programación en su base de datos.

3. El temporizador ejecuta el trabajo en la fecha y hora especificada en la programación. El trabajo crea un evento que se agrega a una cola que mantiene el servidor de reportes.

4. El evento hace que se produzca un proceso de reporte o suscripción. Los eventos se procesan cuando son detectados en la cola; el reporte se procesa o entrega en consecuencia.

El Servidor de Reportes admite extensiones de autenticación personalizadas, extensiones de procesamiento de datos, extensiones de representación y extensiones de entrega. Un servidor de reportes requiere al menos una extensión de seguridad, una extensión de procesamiento de datos y una extensión de representación. Las extensiones de entregas son opcionales, pero necesarias si desea admitir controles personalizados o de distribución de reportes.

Las extensiones de seguridad se utilizan para autenticar y autorizar usuarios y grupos para un servidor de reportes. La extensión de seguridad predeterminada se basa en la autenticación del sistema en el que sea insertado el Sistema de Reportes.

Las extensiones de procesamiento de datos se utilizan para consultar una fuente de datos. Y cuando esto sucede, devuelven un conjunto de filas planas. El Sistema de Reportes podría utilizar diferentes extensiones para interactuar con distintos tipos de fuentes de datos.

Las extensiones de representación convierten los datos y la información de diseño del Procesador de reportes en el formato específico de un dispositivo, ejemplo HTML, Excel, XML, imagen, PDF.

Las extensiones de entrega son utilizadas por el Procesador de Entrega y Programación para entregar los reportes en diversas ubicaciones. El Sistema de Reportes contiene una extensión de entrega por correo electrónico y una extensión de entrega a recursos compartidos de archivos. La extensión de entrega por correo electrónico envía un mensaje de correo electrónico mediante el Protocolo simple de transferencia de correo (SMTP) que contenga el reporte o un vínculo de dirección URL al mismo. La

extensión de entrega a recursos compartidos de archivos guarda reportes en una carpeta compartida en la red. Se puede especificar la ubicación, el formato de representación, el nombre de archivo y las opciones de sobrescritura del archivo que se crea. Las extensiones de entrega funcionan conjuntamente con las suscripciones. Cuando un usuario crea una suscripción, elige una de las extensiones de entrega disponibles para determinar cómo se entrega el reporte.

Otro componente del Sistema de Reportes es el Administrador de Reportes, el cual es una herramienta administrativa que se utiliza gestionar los reportes vía Web. Se puede usar el Administrador para buscar en el Servidor de Reportes las carpetas y reportes a utilizar. Estas se pueden gestionar dependiendo del rol del usuario. La seguridad basada en roles es uno de los aspectos más importantes de esta herramienta. Dentro de las tareas que gestiona el Administrador se encuentran las siguientes: gestionar reportes, gestionar carpetas, configurar la seguridad basada en roles, crear la programación y entrega de reportes, gestionar modelos semánticos, entre otras.

2.4.4 Patrones de casos de uso.

Un patrón es una descripción de un problema y su solución que recibe un nombre que puede emplearse en otros contextos indicando la manera de utilizarlo en circunstancias diversas. (Larman, 1999)

Patrones aplicados en el Diagrama de Casos de Uso del Sistema.

Extensión Concreta

Este patrón consiste en dos casos de uso y una relación de extensión entre ellos. El caso de uso extendido es concreto, es decir, este puede ser instanciado por sí solo, así como, ser una extensión del caso de uso base. El caso de uso base puede ser concreto o abstracto.

Este patrón es aplicable cuando un flujo de datos puede ser extendido del flujo de datos de otro caso de uso, así como ser ejecutado por sí solo. (Overgaard, et al., 2004)

CRUD: Completo

Este patrón consiste en un caso de uso llamado CRUD que fusiona las diferentes operaciones que pueden ser realizadas como simples casos de uso tales como: crear, leer, actualizar y eliminar segmentos de información dentro de un caso de uso formando una sola unidad conceptual.

Este patrón debe ser utilizado cuando todos los flujos contribuyen al mismo valor de negocio y son todos cortos y simples.

CRUD posee algunas ventajas obvias como son, la reducción del tamaño del modelo agrupando las 4 funciones básicas en una sola haciendo el modelo más claro para el analista. (Overgaard, et al., 2004)

CRUD: Parcial

Este patrón consiste en modelar una de las alternativas de un caso de uso como un caso de uso separado, y debe ser utilizado cuando una de estas alternativas sea más significativa, más larga, o mucho más compleja que las demás. (Overgaard, et al., 2004)

Múltiples actores: Rol común

Es un patrón alternativo, donde hay dos actores que desempeñan el mismo papel hacia el caso de uso. Este papel es representado por otro actor, heredado por los actores que comparten este papel. Es aplicable cuando, desde el punto de vista del caso de uso, hay solamente una entidad externa interactuando con cada instancia del caso de uso. (Overgaard, et al., 2004)

2.4.5 Identificación y justificación de los actores del sistema.

Actor	Descripción
Administrador	Es el encargado de definir reportes, de administrar las funcionalidades del Sistema de Reportes.
Usuario	Es toda persona que interactúe con el sistema.

Tabla 3: Identificación y justificación de los actores del sistema.

2.4.6 Diagrama de casos de uso del sistema.

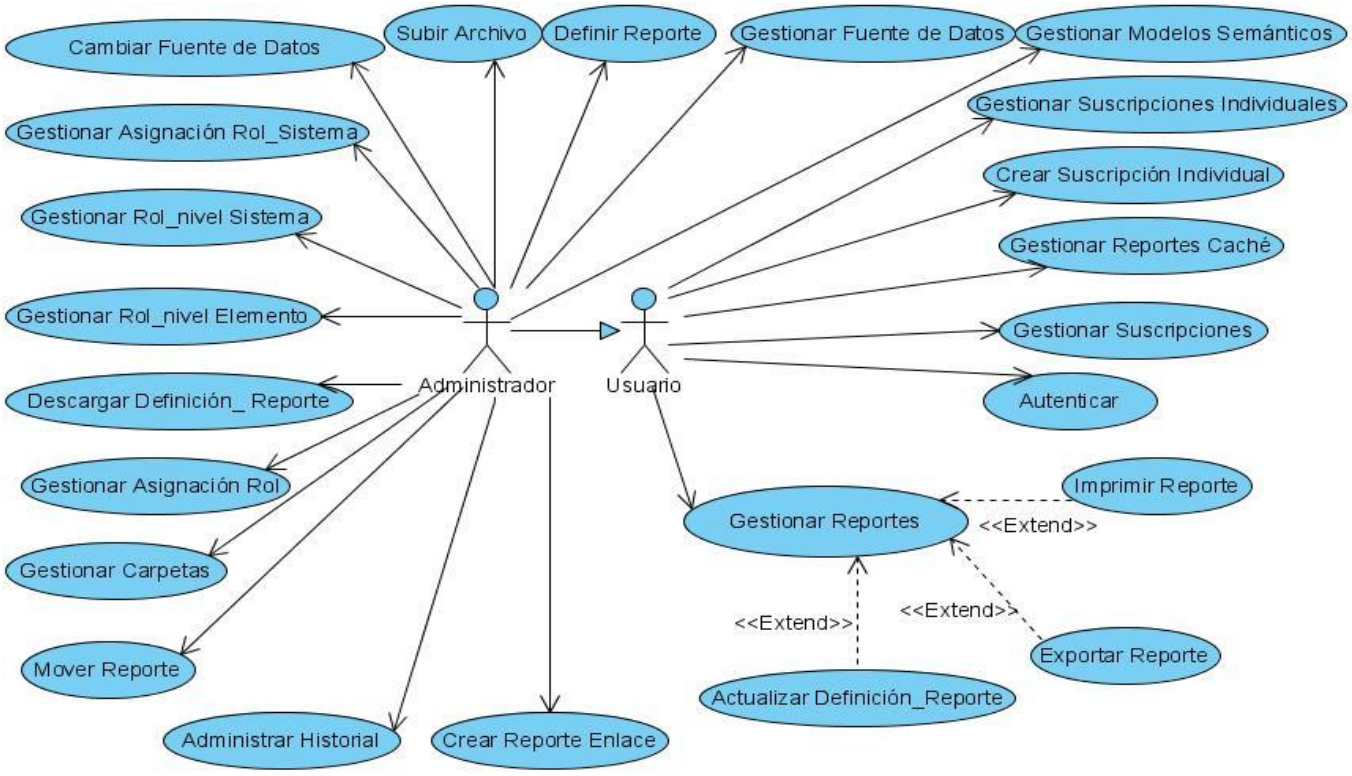


Figura 5: Diagrama de casos de uso del sistema.

2.4.7 Especificación de los casos de uso del sistema.

Las descripciones detalladas de los casos de uso del sistema brindan una información más profunda de las funcionalidades del sistema. (Ver Anexo #13)

Caso de Uso:	**Autenticar**
Actores:	Usuario(inicia)
Resumen:	El caso de uso inicia cuando el usuario entra el nombre de usuario y la contraseña para acceder al sistema, estos datos se validan y se permite la entrada del usuario al sistema con los privilegios de su rol.
Precondiciones:	El usuario debe de estar registrado previamente.
Referencias	R19, 19.1
Prioridad	Crítico
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El usuario introduce nombre y contraseña y selecciona “Aceptar”.	2. El sistema procesa los datos entrados por el usuario. 3. El sistema le da entrada al usuario habilitando los permisos correspondientes a su rol.

Prototipo de Interfaz

Flujos Alternos

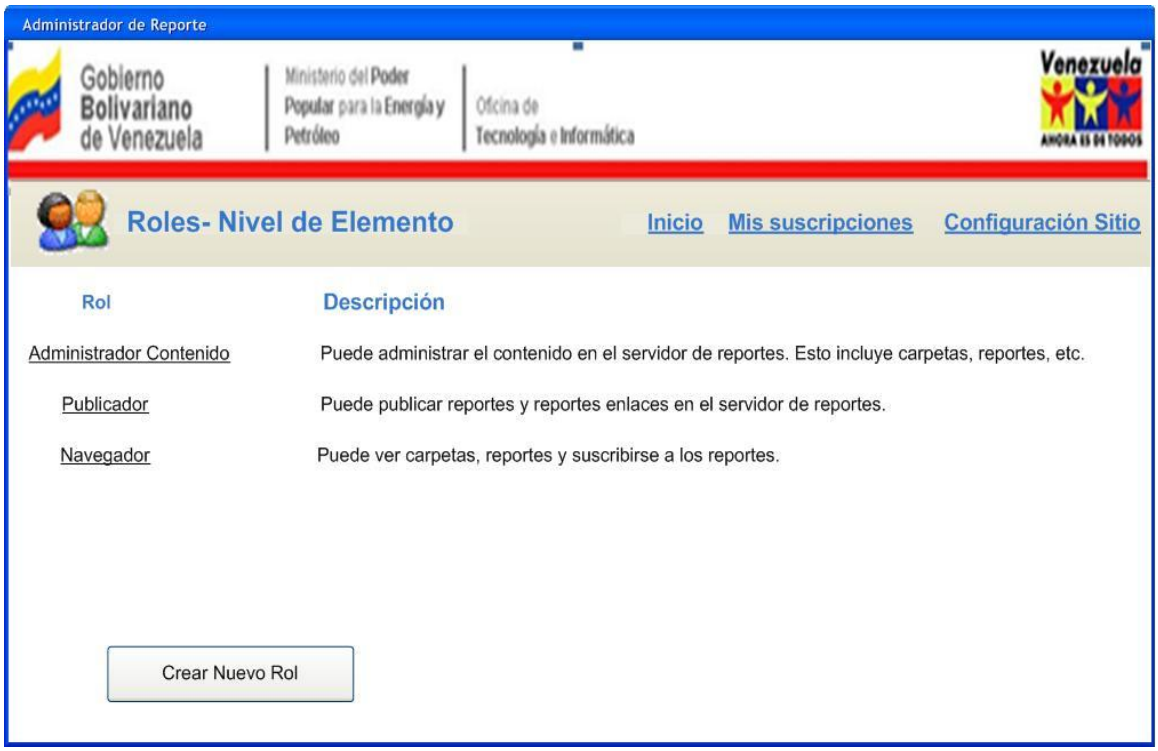
Acción del Actor	Respuesta del Sistema
	2.1 El sistema detecta errores en los datos insertados y los notifica al Usuario, solicitando que sean rectificados.
2.2 El Usuario arregla los errores y solicita aceptar. Regresar a la línea 2 del flujo normal de eventos.	

* El usuario selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones	Se garantiza la entrada del usuario al sistema con los privilegios de su rol.

Caso de Uso:	**Gestionar Rol _ nivel Elemento**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere crear, modificar o eliminar algún rol a nivel de elemento.
Precondiciones:	Para modificar un rol, deben existir roles creados.
Referencias	R2, 2.1, 2.2, 2.3
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección “Gestionar Rol _ nivel Elemento”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador solicita la interfaz correspondiente a “Configurar Sitio”.	2. El sistema muestra una interfaz donde se le permite al administrador: a) “Configurar Definición de rol a nivel de elemento”. b) “Configurar Definición de rol a nivel de sistema”.
3. El administrador selecciona la opción de “Configurar Definición de rol a nivel de elemento”.	4. El sistema muestra una interfaz donde aparecen los roles existentes y le permite al administrador: • Crear rol • Modificar rol

	• Eliminar rol

Prototipo de Interfaz

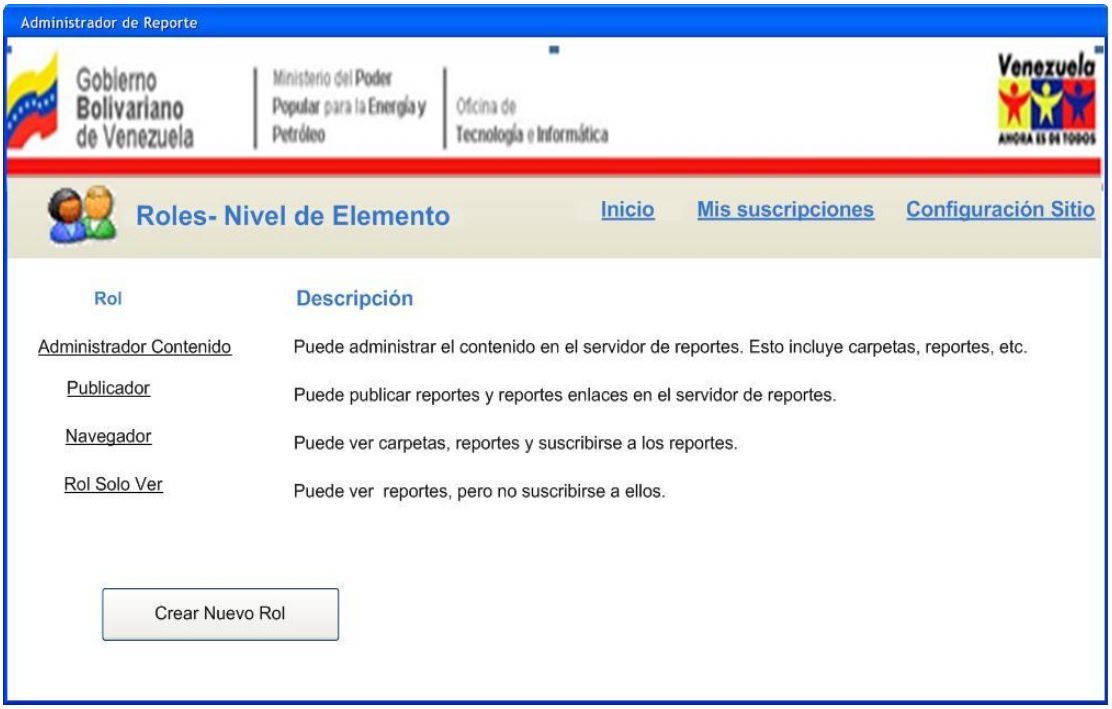


Flujo Normal de Eventos

Sección "Crear Rol_nivel Elemento"

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción "Crear Nuevo Rol". 3. El administrador entra la información solicitada, asigna las tareas correspondientes al rol y selecciona "Aceptar".	2. El sistema muestra la interfaz donde aparecen todas las tareas que permite desarrollar a nivel de elemento y se le pueden asignar a un rol, además de la solicitud del nombre que recibirá el rol y una pequeña descripción del mismo. 4. El sistema muestra la interfaz donde aparecen todos los roles existentes, incluyendo el rol creado.

Prototipo de Interfaz



Flujo Normal de Eventos	
Sección "Modificar Rol_nivel Elemento"	
Acción del Actor	**Respuesta del Sistema**

1. El administrador selecciona el rol que quiere modificar. 3. El administrador actualiza la lista de tareas asignadas al rol y selecciona “Aceptar”.	2. El sistema muestra todas las tareas que permite desarrollar a nivel de elemento, incluyendo las asignadas al rol. 4. El sistema modifica el rol y muestra un mensaje de que la acción se realizó satisfactoriamente.
Flujo Normal de Eventos	
Sección “Eliminar Rol_nivel Elemento”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona el rol que quiere eliminar. 3. El administrador selecciona” Eliminar”. 5. El administrador selecciona “Aceptar”	2. El sistema muestra todas las tareas que permite desarrollar a nivel de elemento, incluyendo las asignadas al rol. 4. El sistema muestra un mensaje de confirmación. 6. El sistema elimina el rol.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El administrador selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Crear Rol_nivel Elemento” y Sección “Modificar Rol_nivel Elemento” se obtiene el rol deseado. Sección “Eliminar Rol_nivel Elemento” se elimina el rol.

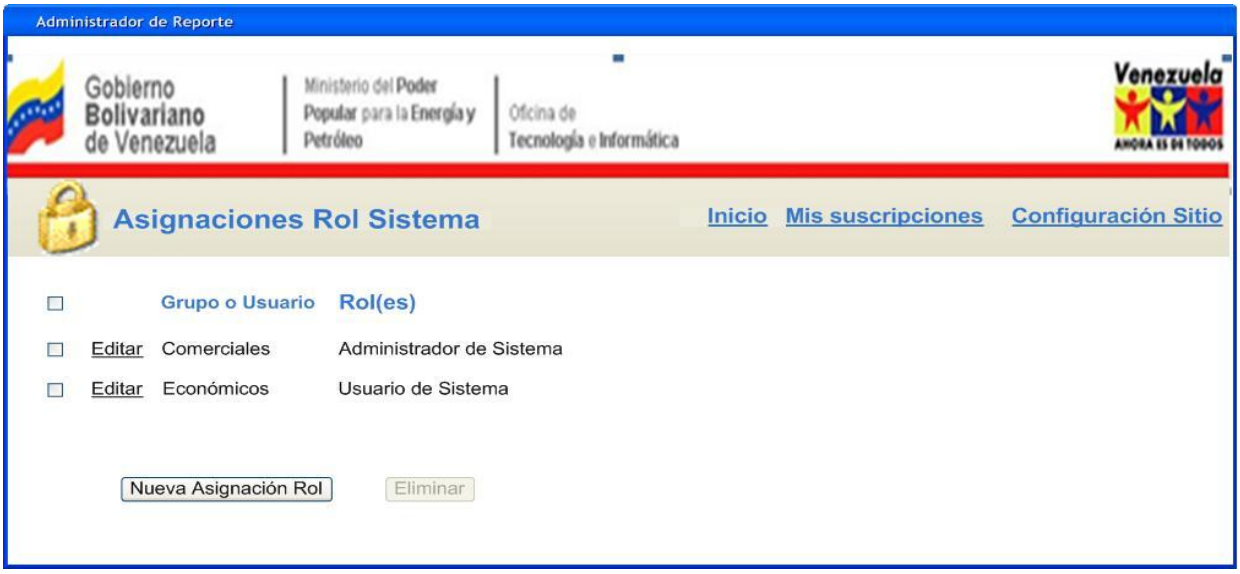
Caso de Uso:	**Gestionar Asignación Rol _Sistema**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere crear, modificar o eliminar una asignación de rol de sistema a un usuario o grupo.
Precondiciones:	Para modificar una asignación de rol de sistema a un usuario o grupo, deben

<table>
<tr><td></td><td colspan="2">tener roles asignados.</td></tr>
<tr><td>Referencias</td><td colspan="2">R10, 10.1, 10.2, 10.3</td></tr>
<tr><td>Prioridad:</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Gestionar Asignación Rol _ Sistema”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El administrador solicita la interfaz correspondiente a “Configurar Sitio”.

3.El administrador selecciona la opción de “Configurar ancho de seguridad del sitio”</td><td>2. El sistema muestra una interfaz donde le permite configurar la Seguridad del sitio.

4. El sistema muestra todos los usuarios o grupos existentes con roles asignados y le permite al administrador:
• Crear Asignación Rol
• Modificar Asignación Rol
• Eliminar Asignación Rol</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>

Flujo Normal de Eventos	
Sección "Crear Asignación Rol Sistema"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción "Nueva Asignación Rol". 3. El administrador entra la información solicitada, asigna uno o más roles al usuario o grupo y selecciona "Aceptar".	2. El sistema muestra la interfaz donde aparecen todos los roles existentes a nivel del mismo con sus correspondientes descripciones y que pueden ser asignados a un usuario o grupo, además de la solicitud del nombre de usuario o grupo al que le asignará el rol. 4. El sistema muestra la interfaz donde aparecen todos los usuarios o grupos existentes con roles asignados, incluyendo la nueva asignación.
Prototipo de Interfaz	



Flujo Normal de Eventos	
Sección "Modificar Asignación Rol Sistema"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción "Editar" del	2. El sistema muestra la interfaz de "Nueva

usuario o grupo al cual le desea modificar la asignación de rol. 3. El administrador actualiza la lista de roles o el rol asignado al usuario o grupo y selecciona “Aceptar”.	Asignación Rol” donde aparecen todos los roles existentes incluyendo el o los asignados al usuario o grupo. 4. El sistema modifica la asignación de rol de este usuario o grupo y muestra un mensaje de que la acción se realizó satisfactoriamente.
Flujo Normal de Eventos	
Sección “Eliminar Asignación Rol Sistema”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona el usuario o grupo al cual le desea eliminar la asignación de rol y selecciona “Eliminar”. 3. El administrador selecciona “Aceptar”.	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina la asignación de rol correspondiente al usuario o grupo seleccionado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El administrador selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Crear Asignación Rol Sistema” y Sección “Modificar Asignación Rol Sistema” se le asigna el rol deseado a un usuario o grupo. Sección “Eliminar Asignación Rol Sistema” se elimina la asignación de rol.

Caso de Uso:	**Gestionar Asignación Rol**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere crear, modificar o eliminar una asignación de rol a un usuario o grupo.
Precondiciones:	Para modificar una asignación de rol a un usuario o grupo, deben tener roles

<table>
<tr><td></td><td colspan="2">asignados.</td></tr>
<tr><td>Referencias</td><td colspan="2">R9, 9.1, 9.2, 9.3</td></tr>
<tr><td>Prioridad:</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Gestionar Asignación Rol ”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. El administrador selecciona la carpeta que desea asegurar para evitar el acceso no autorizado de un usuario o grupo, además de la modificación de su contenido, de sus propiedades.

3. El administrador selecciona la pestaña “Propiedades” y después la opción “Seguridad”.

5. El administrador selecciona la opción “Editar Seguridad”.</td><td colspan="2">2. El sistema muestra una interfaz donde aparecen las pestañas “Contenido” y “Propiedades".

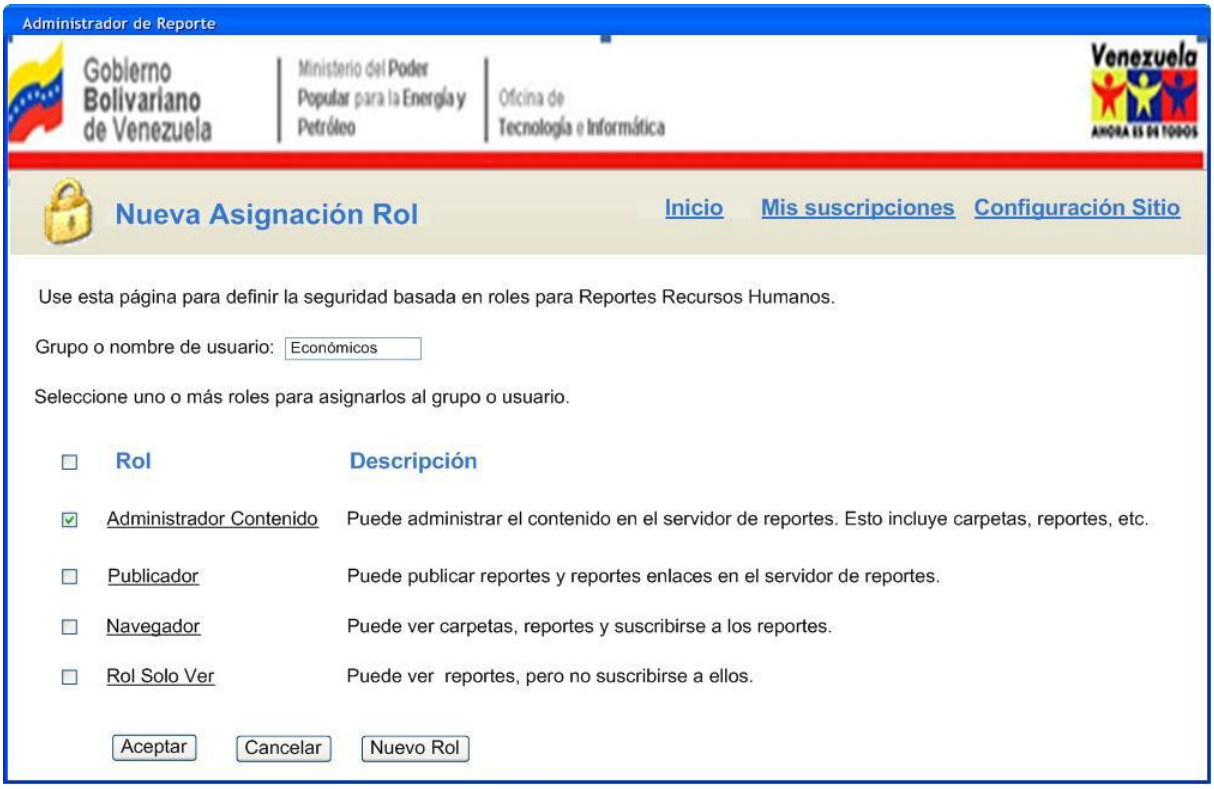
4. El sistema muestra todos los usuarios o grupos existentes con roles asignados, además de permitir editar la seguridad de la carpeta.

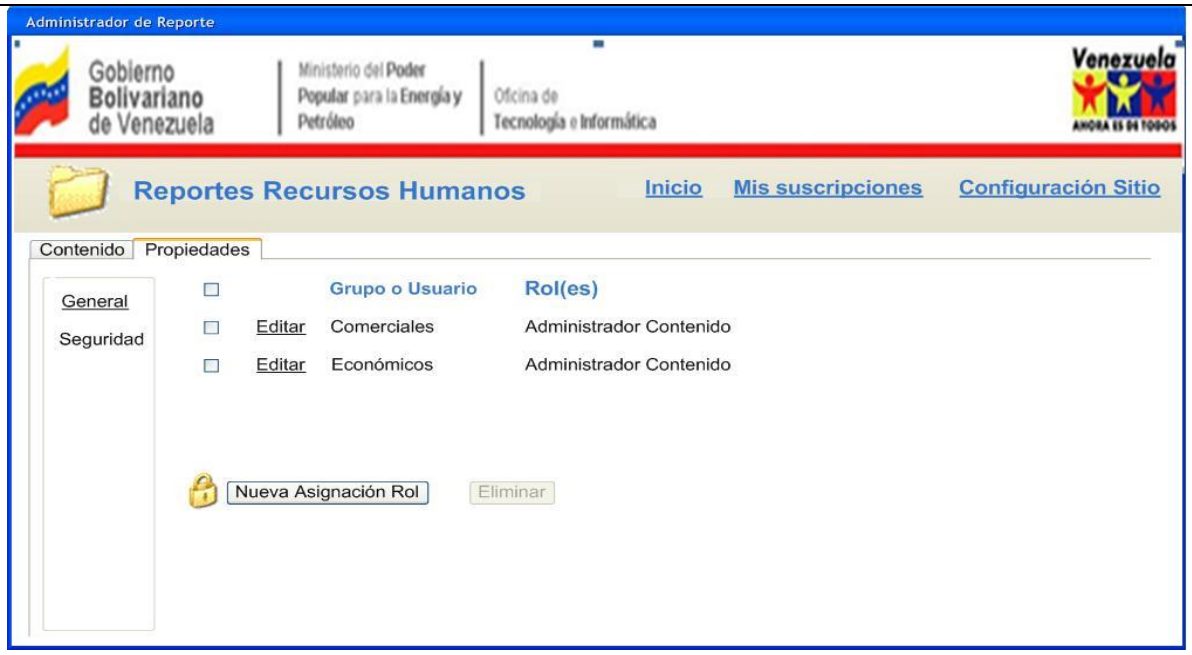
6. El sistema muestra una interfaz donde le permite al administrador:
• Crear Asignación Rol
• Modificar Asignación Rol
• Eliminar Asignación Rol</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>

Flujo Normal de Eventos	
Sección "Crear Asignación Rol"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción "Nueva Asignación Rol".	2. El sistema muestra la interfaz donde aparecen todos los roles a nivel de elementos

3. El administrador entra la información solicitada, asigna uno o más roles al usuario o grupo y selecciona “Aceptar”.	existentes con sus correspondientes descripciones y que se le pueden asignar a un usuario o grupo, además de la solicitud del nombre de usuario o grupo al que le asignará el rol. 4. El sistema muestra la interfaz donde aparecen todos los usuarios o grupos existentes con roles asignados, incluyendo la nueva asignación.

Prototipo de Interfaz



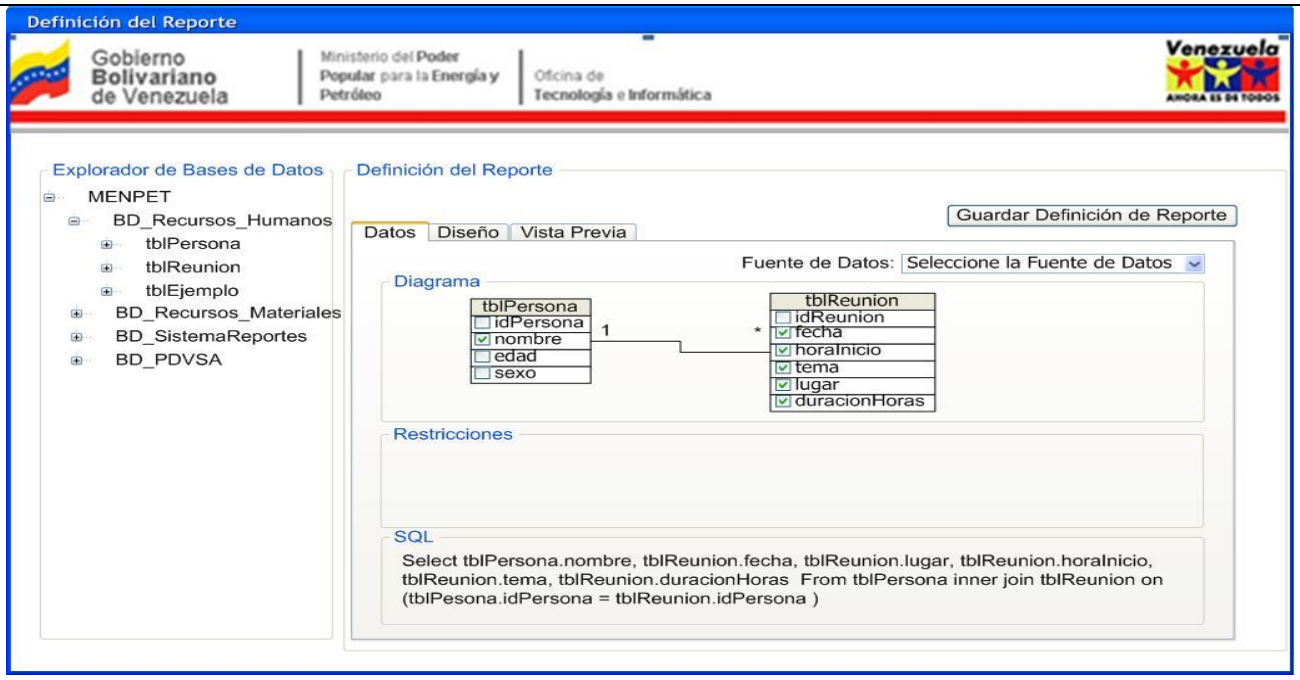


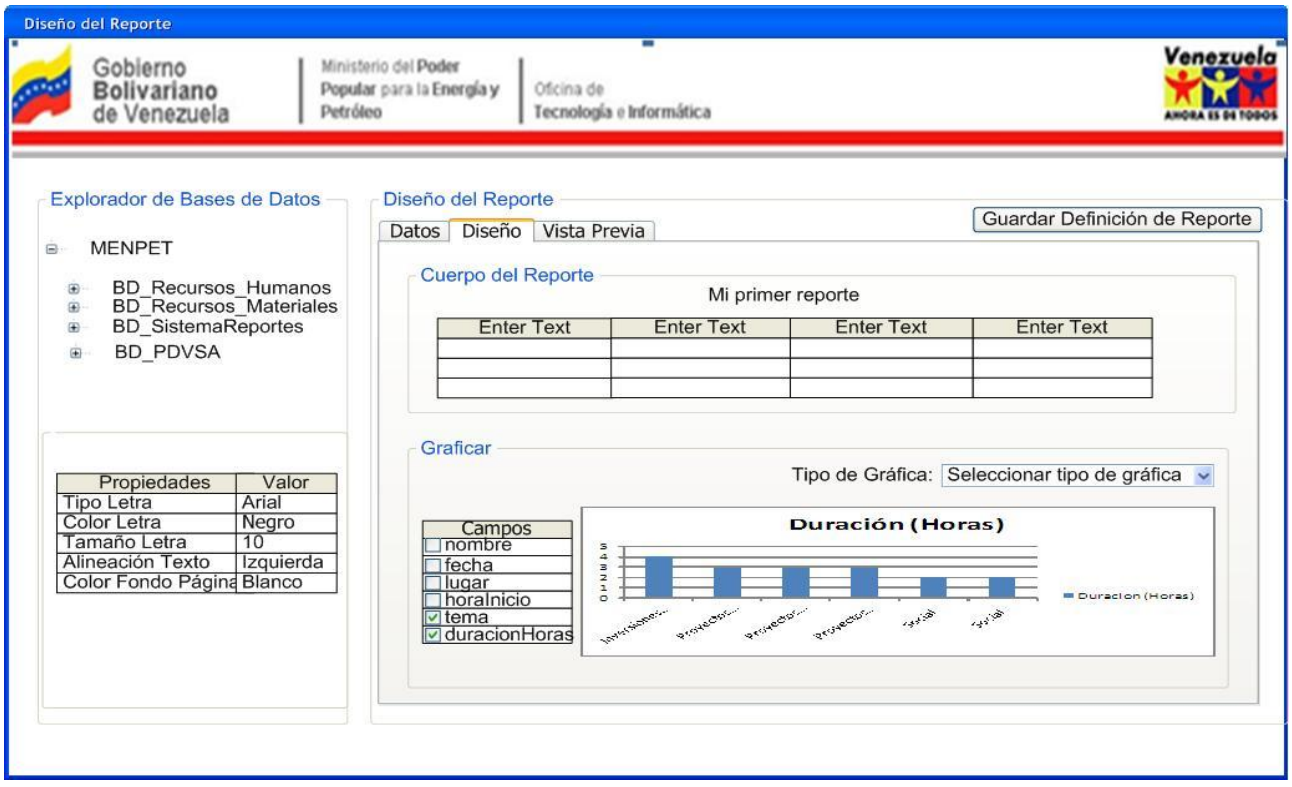
Flujo Normal de Eventos	
Sección "Modificar Asignación Rol "	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción "Editar" del usuario o grupo al cual le desea modificar la asignación de rol. 3. El administrador actualiza la lista de roles o el rol asignado al usuario o grupo y selecciona "Aceptar".	2. El sistema muestra la interfaz de "Nueva Asignación Rol" donde aparecen todos los roles existentes incluyendo el o los asignados al usuario o grupo. 4. El sistema modifica la asignación de rol de este usuario o grupo y muestra un mensaje de que la acción se realizó satisfactoriamente.
Flujo Normal de Eventos	
Sección "Eliminar Asignación Rol"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona el usuario o grupo al cual le desea eliminar la asignación de rol y selecciona "Eliminar".	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina la asignación de rol correspondiente al usuario o grupo

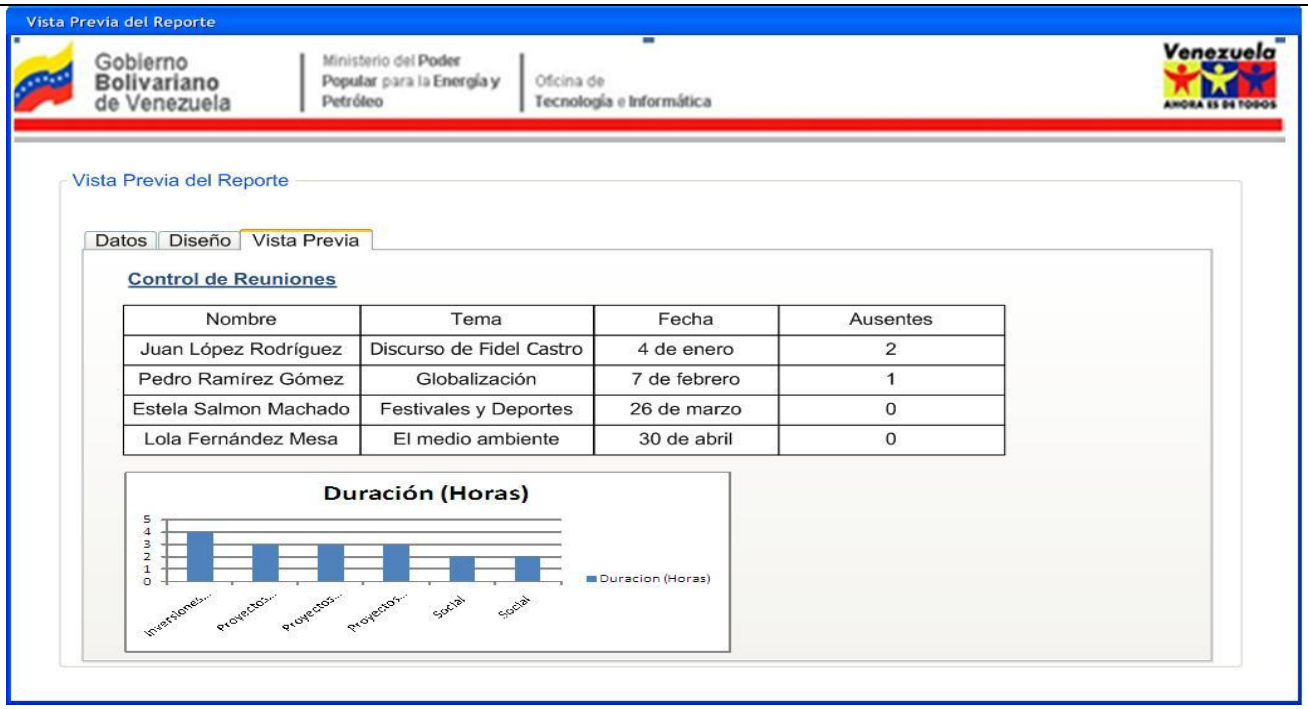
3. El administrador selecciona “Aceptar”.	seleccionado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El administrador selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Crear Asignación Rol” y Sección “Modificar Asignación Rol” se le asigna el rol deseado a un usuario o grupo. Sección “Eliminar Asignación Rol” se elimina la asignación de rol.

Caso de Uso:	**Definir Reporte**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador desea crear una definición de reporte.
Precondiciones:	El administrador debe estar autenticado en el sistema.
Referencias	R23, 23.1, 23.1.1, 23.1.2, 23.1.3, 23.1.4, 23.1.5, 23.1.6, 23.1.7, 23.1.8, 23.2, 23.2.1, 23.3
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción “Definir Reporte”.	2. El sistema brinda una interfaz donde muestra los modelos semánticos de las bases de datos. Además muestra las pestañas Datos, Diseño y Vista Previa.
3. El administrador selecciona la opción “Datos”.	4. El sistema le permite establecer los datos de la definición del reporte que se desea construir.
5. El administrador selecciona de una base de datos las tablas involucradas en la definición del reporte, arrastrando hacia la sección Diagrama la(s) tabla(s)	6. El sistema relaciona la(s) tabla(s) si en el modelo de datos están relacionadas.

que seleccionó. 7. Selecciona los campos deseados de las tablas seleccionadas y puede incluir parámetros condicionales si el reporte que desea construir lo amerita. 9. El administrador selecciona la fuente de datos que necesita para conectarse a la base de datos. 11. El administrador selecciona la pestaña “Diseño”. 13. El administrador diseña el reporte (selecciona el “Tipo de letra”, “Tamaño de letra”, la organización del reporte). Además selecciona el tipo de gráfica si desea incluir alguna. 15. El administrador selecciona los campos y la opción “Guardar Definición“. 17. El administrador selecciona la pestaña “Vista Previa”. 19. Si el administrador ve que el reporte no coincide con todo lo que desea, realiza los cambios necesarios y los guarda.	8. El sistema muestra la consulta a la base de datos, generada a través de los campos seleccionados en la(s) tabla(s) y los parámetros condicionales incluidos. 10. El sistema establece conexión con la base de datos. 12. El sistema muestra una interfaz donde permite personalizar el reporte, además de incluir gráficas. 14. El sistema permite seleccionar los campos que se desean mostrar en la gráfica. 16. El sistema guarda la definición de reporte creada en la base de datos del servidor de reportes. 18. El sistema genera el reporte y lo muestra al administrador. 20. Guarda los cambios realizados en la definición de reporte.
Prototipo de Interfaz	


Definición del Reporte
Gobierno Bolivariano de Venezuela
Ministerio del Poder Popular para la Energía y Petróleo
Oficina de Tecnología e Informática
Venezuela
AHORA ES DE TODOS
Explorador de Bases de Datos
MENPET
BD_Recursos_Humanos
tblPersona
tblReunion
tblEjemplo
BD_Recursos_Materiales
BD_SistemaReportes
BD_PDVSA
Definición del Reporte
Guardar Definición de Reporte
Datos
Diseño
Vista Previa
Fuente de Datos: Seleccione la Fuente de Datos
Diagrama
tblPersona
idPersona
nombre
edad
sexo
1
*
tblReunion
idReunion
fecha
horaInicio
tema
lugar
duracionHoras
Restricciones
SQL
Select tblPersona.nombre, tblReunion.fecha, tblReunion.lugar, tblReunion.horaInicio, tblReunion.tema, tblReunion.duracionHoras From tblPersona inner join tblReunion on (tblPesona.idPersona = tblReunion.idPersona)


Diseño del Reporte
Gobierno Bolivariano de Venezuela
Ministerio del Poder Popular para la Energía y Petróleo
Oficina de Tecnología e Informática
Venezuela
AHORA ES DE TODOS
Explorador de Bases de Datos
MENPET
BD_Recursos_Humanos
BD_Recursos_Materiales
BD_SistemaReportes
BD_PDVSA
Propiedades
Valor
Tipo Letra
Arial
Color Letra
Negro
Tamaño Letra
10
Alineación Texto
Izquierda
Color Fondo Página
Blanco
Diseño del Reporte
Guardar Definición de Reporte
Datos
Diseño
Vista Previa
Cuerpo del Reporte
Mi primer reporte
Enter Text
Enter Text
Enter Text
Enter Text
Graficar
Tipo de Gráfica: Seleccionar tipo de gráfica
Campos
nombre
fecha
lugar
horaInicio
tema
duracionHoras
Duración (Horas)
Duracion (Horas)



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
9. El administrador no selecciona una fuente de datos. 9.2 El administrador selecciona la fuente de datos necesaria para obtener los datos del reporte. Regresar a la línea 10 del flujo normal de eventos.	9.1 El sistema muestra un mensaje solicitando la selección de una fuente de datos.
Poscondiciones:	Se crea la definición de reporte deseada.

Caso de Uso:	**Gestionar Carpetas**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere crear, modificar o eliminar una carpeta.
Precondiciones:	Para modificar una carpeta, debe haber carpetas creadas.

<table>
<tr><td>Referencias</td><td colspan="2">R1, 1.1, 1.2, 1.3</td></tr>
<tr><td>Prioridad:</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Gestionar Carpetas”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El administrador navega en la página inicial.</td><td>2.El sistema le permite al administrador:
Crear Carpeta
Modificar Carpeta
Eliminar Carpeta</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Crear Carpeta”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El administrador selecciona la opción “Nueva Carpeta”.

3. El administrador entra la información solicitada y selecciona “Aceptar”.</td><td>2. El sistema muestra la interfaz donde solicita el nombre de la nueva carpeta y una pequeña descripción de la misma.
4. El sistema retorna a la interfaz principal donde muestra la carpeta creada.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>

Flujo Normal de Eventos	
Sección "Modificar Carpeta "	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la carpeta que desea modificar. 3. El administrador selecciona la pestaña "Propiedades" y luego la opción "General". 5. El administrador introduce la información solicitada y selecciona la opción "Aceptar".	2. El sistema muestra una interfaz donde aparecen las pestañas "Contenido" y "Propiedades". 4. El sistema muestra una interfaz donde permite actualizar el nombre y la descripción de la carpeta seleccionada. 6. El sistema guarda los cambios realizados.

Flujo Normal de Eventos	
Sección "Eliminar Carpeta"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la carpeta que desea eliminar y la opción "Eliminar" en la barra de herramientas.	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina la carpeta seleccionada.

3. El administrador selecciona “Aceptar”.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El administrador selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Crear Carpeta” y Sección “Modificar Carpeta” se obtiene la carpeta deseada. Sección “Eliminar Carpeta” se elimina la carpeta.

Caso de Uso:	**Subir Archivo**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere publicar un reporte.
Precondiciones:	Para publicar un reporte, debe haber definiciones de reportes creadas.
Referencias	R25
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la carpeta donde desea cargar el archivo.	2. El sistema muestra una interfaz con el contenido de la carpeta.
3. El administrador selecciona la opción “Subir Archivo” en la barra de herramientas.	4. El sistema le solicita el nombre del archivo que va a subir a la carpeta, y le permite examinar la ubicación del archivo .dr, además de sobrescribirlo si ya existe.
5. El administrador introduce el nombre y busca el archivo .dr, selecciona la opción “Sobrescribir elemento” y después “Aceptar”.	6. El sistema sube el archivo y retorna a la interfaz donde muestra el contenido de la carpeta, incluyendo este último archivo.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5. El administrador no selecciona la opción “Sobrescribir elemento”. 5.2 El administrador selecciona “Aceptar”. Regresar a la línea 6 del flujo normal de eventos. * El administrador selecciona “Cancelar” y termina el caso de uso.	5.1 El sistema muestra un mensaje de confirmación si al intentar subir el archivo este ya existe.
Poscondiciones:	Se publica el reporte en la ubicación deseada.

Caso de Uso:	**Gestionar Reportes**
Actores:	Usuario (inicia)
Resumen:	El caso de uso inicia cuando el usuario requiere modificar, visualizar o eliminar un reporte.
Precondiciones:	Para modificar un reporte, debe haber reportes publicados.

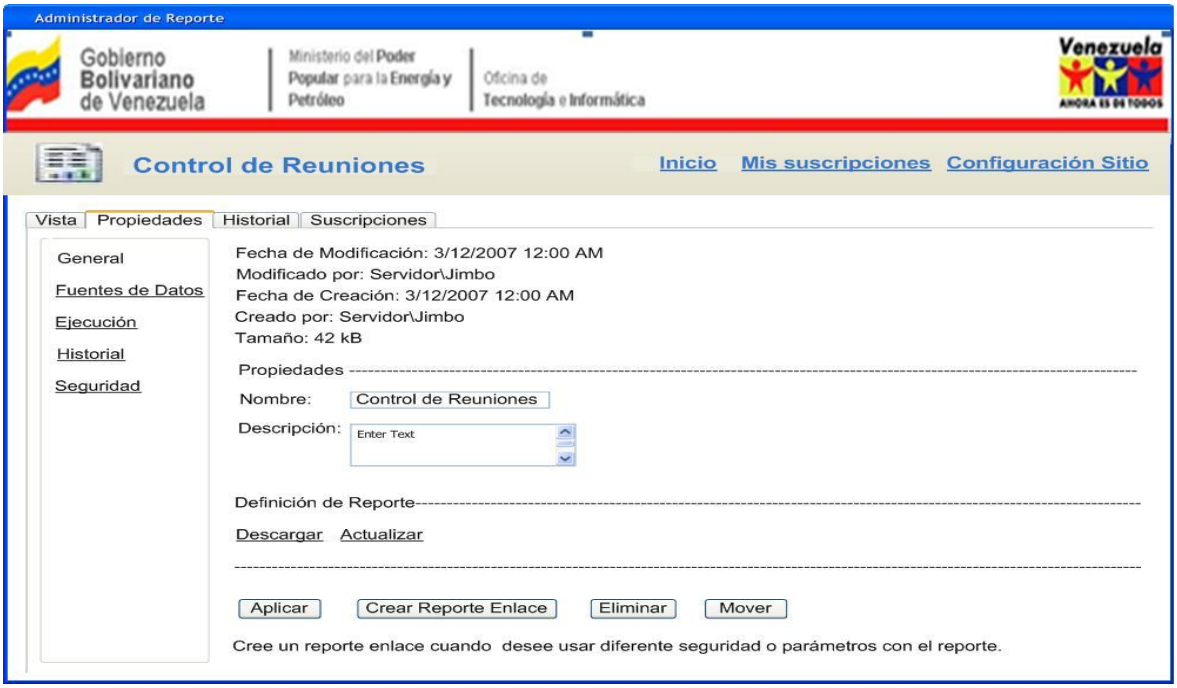
Referencias	R16, 16.1, 16.2, 16.2.1, 16.2.1.1, 16.2.1.2, 16.2.1.3, 16.2.1.4, 16.2.1.5, 16.2.1.6, 16.2.2, 16.3, R17
CU asociados	Actualizar Definición_Reporte (extendido), Imprimir Reporte(extendido), Exportar Reporte(extendido)
Prioridad:	Crítico

Flujo Normal de Eventos	
Sección "Gestionar Reportes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la carpeta en donde se encuentra el reporte que desea modificar, visualizar o eliminar. 3. El usuario selecciona el reporte.	2. El sistema muestra una interfaz con el contenido de la carpeta, además de la pestaña "Propiedades". 4. El sistema muestra las pestañas "Vista", "Propiedades", "Historial" y "Suscripciones" del reporte seleccionado y le permite al usuario: • Modificar reporte • Visualizar reporte • Eliminar reporte

Flujo Normal de Eventos	
Sección "Modificar Reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la pestaña "Propiedades" y después la opción "General". 3. El usuario entra los nuevos datos del reporte y selecciona "Aplicar".	2. El sistema muestra la interfaz donde le permite al usuario modificar el nombre y la descripción del reporte solicitado, además de actualizar la definición de reporte. 4. El sistema guarda todos los cambios realizados en el reporte. Si el usuario desea actualizar la definición de reporte ver el caso de uso Actualizar

	Definición_Reporte.

Prototipo de Interfaz



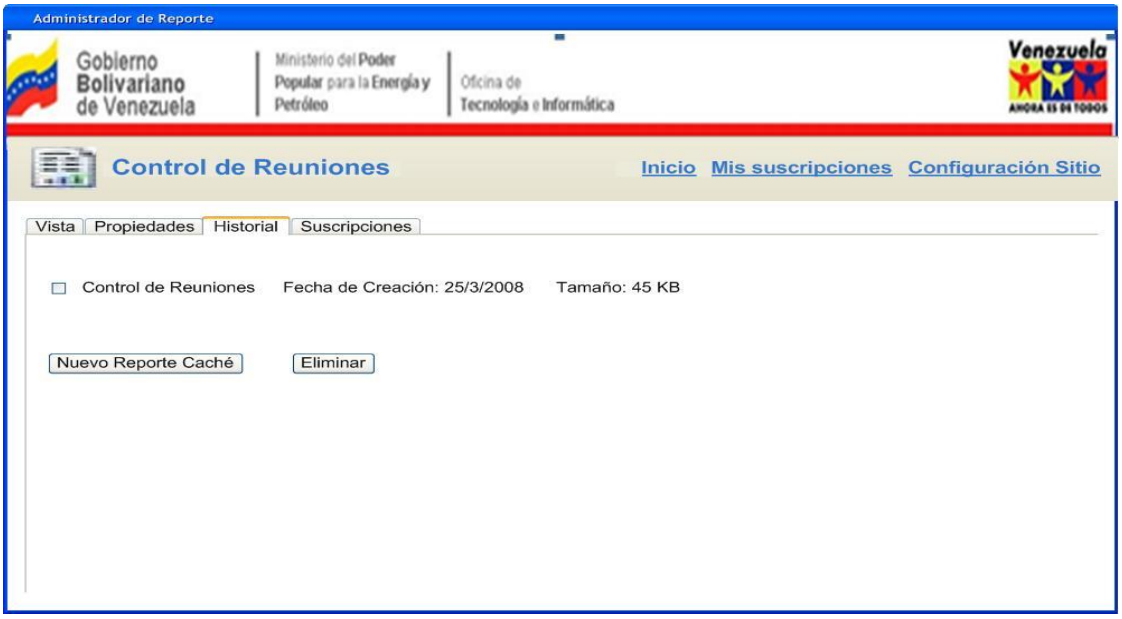
Flujo Normal de Eventos	
Sección "Visualizar Reporte "	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la pestaña "Vista".	2. El sistema genera el reporte y muestra una interfaz con los datos. Si el usuario desea imprimir el reporte ver el caso de uso Imprimir Reporte. Si el usuario desea exportar el reporte ver el caso de uso Exportar Reporte.
Flujo Normal de Eventos	
Sección "Eliminar Reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la pestaña "Propiedades" y	2. El sistema muestra una interfaz que le

después la opción "General". 3. El usuario selecciona "Eliminar". 5. El usuario selecciona la opción "Aceptar".	permite al usuario eliminar reporte. 4. El sistema muestra un mensaje de confirmación. 6. El sistema elimina el reporte.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El usuario selecciona "Cancelar" y termina el caso de uso.	
Poscondiciones:	Sección "Modificar Reporte" el reporte deseado. Sección "Visualizar Reporte" se muestra el reporte seleccionado. Sección "Eliminar Reporte" se elimina el reporte.

Caso de Uso:	**Gestionar Reportes Caché**
Actores:	Usuario (inicia)
Resumen:	El caso de uso inicia cuando el usuario requiere crear, eliminar o visualizar un reporte caché.
Precondiciones:	Para crear un reporte caché, debe haber reportes publicados. Para eliminar o visualizar un reporte caché, debe haber reportes caché creados.
Referencias	R11, 11.1, 11.2, 11.3
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección "Gestionar Reportes Caché"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la carpeta donde se encuentra el reporte al cual le desea crear, eliminar o visualizar un reporte caché.	2. El sistema muestra el contenido de la carpeta seleccionada. 4. El sistema muestra las pestañas "Vista",

3. Selecciona el reporte. 5. Selecciona la pestaña “Historial”.	“Propiedades”, “Historial” y “Suscripciones” del reporte seleccionado. 6. El sistema muestra una interfaz donde le permite al usuario: a) “Crear Reporte Caché”. b) “Eliminar Reporte Caché”. c) “Visualizar Reporte Caché”.

Prototipo de Interfaz



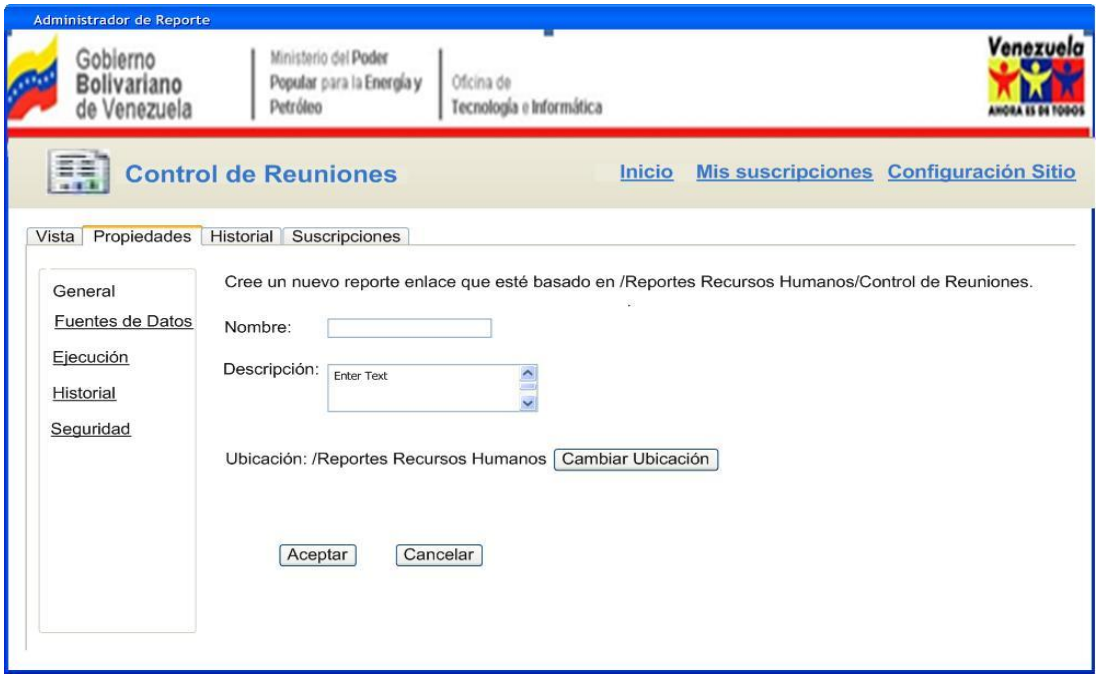
Flujo Normal de Eventos	
Sección “Crear Reporte Caché”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Nuevo Reporte Caché”.	2. El sistema crea el reporte caché y muestra en la interfaz un nuevo reporte caché, con la fecha de creación y el tamaño.
Flujo Normal de Eventos	
Sección “Eliminar Reporte Caché”	
Acción del Actor	**Respuesta del Sistema**

1. El usuario selecciona el reporte caché que desea eliminar y después la opción “Eliminar”. 3. El usuario selecciona “Aceptar”.	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina el reporte caché.
Flujo Normal de Eventos	
Sección “Visualizar Reporte Caché”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona el reporte caché que desea visualizar.	2. El sistema muestra el reporte caché.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
En la Sección “Eliminar Reporte Caché”: 3. El usuario selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Crear Reporte Caché” se obtiene el reporte caché deseado. Sección “Eliminar Reporte Caché” se elimina el reporte caché deseado. Sección “Visualizar Reporte Caché” muestra el reporte caché deseado.

Caso de Uso:	**Crear Reporte Enlace**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere crear un reporte enlace.
Precondiciones:	Debe haber reportes publicados.
Referencias	R18
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la carpeta donde se encuentra el reporte al cual se le realizará el reporte	2. El sistema muestra el contenido de la carpeta seleccionada.

enlace. 3. Selecciona el reporte al que se le creará el reporte enlace. 5. El administrador selecciona la pestaña “Propiedades” y la opción “General”. 7. El administrador selecciona la opción “Crear Reporte Enlace”. 9. El administrador entra la información solicitada y selecciona “Aceptar”.	4. El sistema muestra las pestañas “Vista”, “Propiedades”, “Historial” y “Suscripciones” del reporte seleccionado. 6. El sistema muestra una interfaz donde le permite al administrador crear un reporte enlace. 8. El sistema muestra una interfaz donde aparece la solicitud del nombre que recibirá el reporte enlace y una pequeña descripción del mismo. Y además la ubicación donde se guardará el reporte enlace. 10. El sistema muestra un mensaje de que la acción se realizó satisfactoriamente.

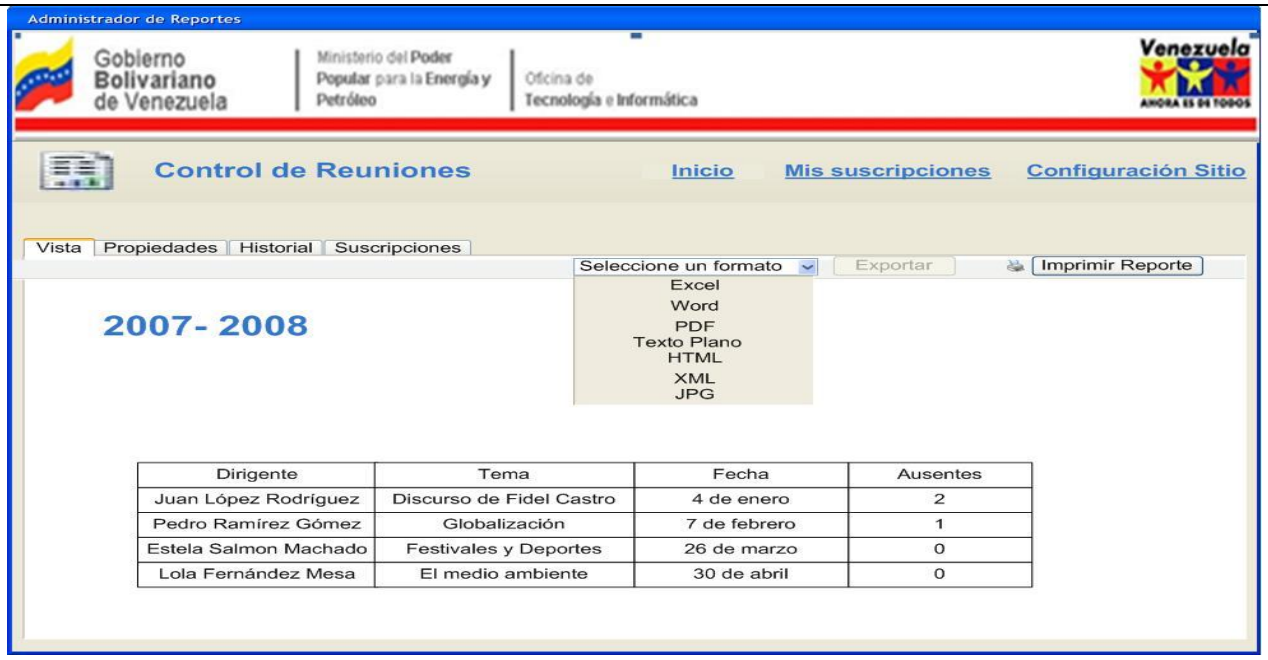
Prototipo de Interfaz



Flujos Alternos

Acción del Actor	Respuesta del Sistema
* El administrador selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Se obtiene el reporte enlace deseado.

Caso de Uso:	**Imprimir Reporte**
Actores:	Usuario (inicia)
Resumen:	El caso de uso inicia cuando el usuario desea imprimir un reporte.
Precondiciones:	Debe haber reportes publicados.
Referencias	R20, 20.1
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona el reporte que desea imprimir. 3. Selecciona la opción “Imprimir”. 5. Elige la opción deseada y acepta la acción.	2. El sistema muestra el contenido del reporte, además da la opción de imprimirlo. 4. El sistema muestra las opciones de impresión. 6. Imprime el reporte solicitado en la opción requerida.
Prototipo de Interfaz	

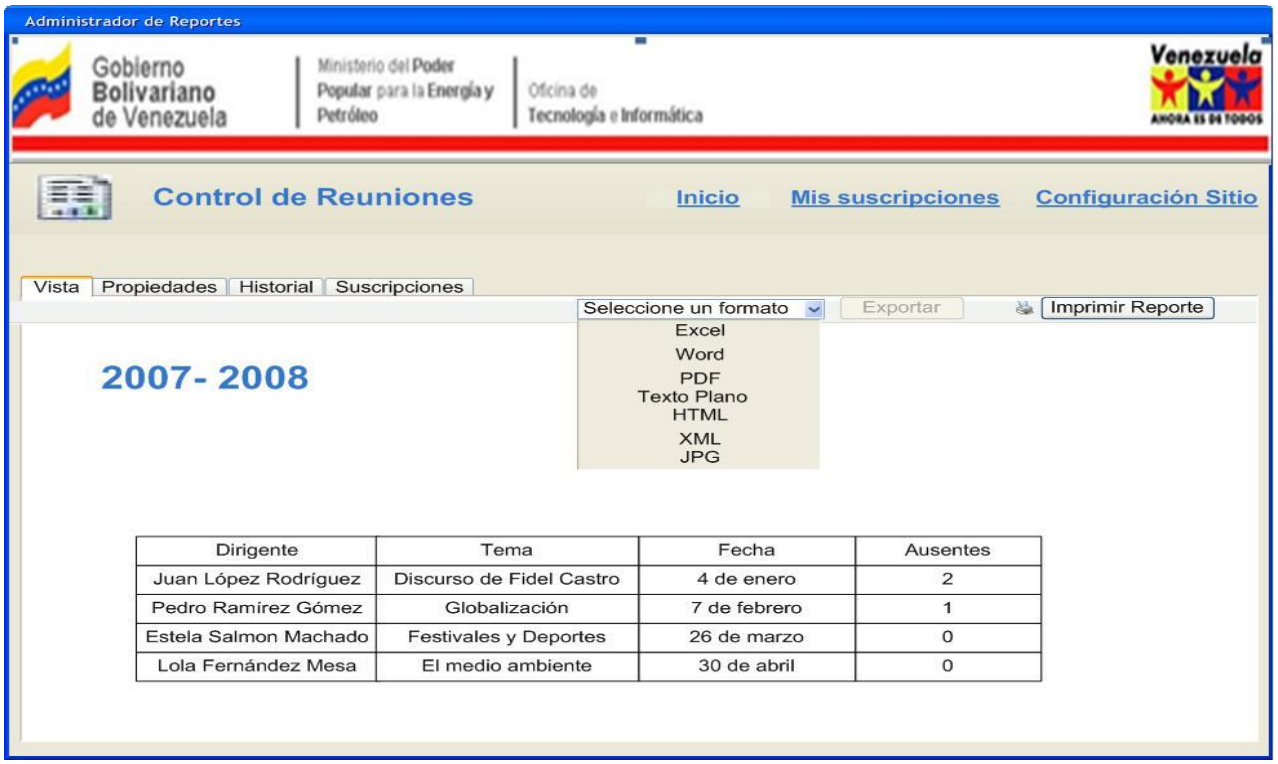


Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El usuario selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Se imprime el reporte.

Caso de Uso:	**Exportar Reporte**
Actores:	Usuario (inicia)
Resumen:	El caso de uso inicia cuando el usuario requiere exportar un reporte a un formato específico.
Precondiciones:	Debe haber reportes publicados.
Referencias	R22, 22.1, 22.2, 22.3, 22.4, 22.5, 22.6, 22.7
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.El usuario selecciona el reporte que desea	2. El sistema muestra el contenido del reporte,

exportar 3. El usuario selecciona el formato y la opción “Exportar”. 5. El usuario selecciona la ubicación y “Aceptar”.	además de permitir exportarlo a un formato específico (Excel, Word, PDF, Texto Plano, HTML, XML, Imagen). 4. El sistema muestra una interfaz solicitando la ubicación donde se guardará el reporte que se va a exportar. 6. El sistema guarda el reporte en el formato y a la ubicación especificada.

Prototipo de Interfaz



Flujos Alternos

Acción del Actor	Respuesta del Sistema
* El usuario selecciona “Cancelar” y termina el caso de uso.	

Poscondiciones:	El reporte es exportado en el formato y a la ubicación especificada.

Caso de Uso:	**Gestionar Suscripciones**
Actores:	Usuario (inicia)
Resumen:	El caso de uso inicia cuando el usuario desea modificar o eliminar sus suscripciones.
Precondiciones:	Debe haber suscripciones creadas.
Referencias	R5, R7, 7.1, 7.2
Prioridad:	Crítico

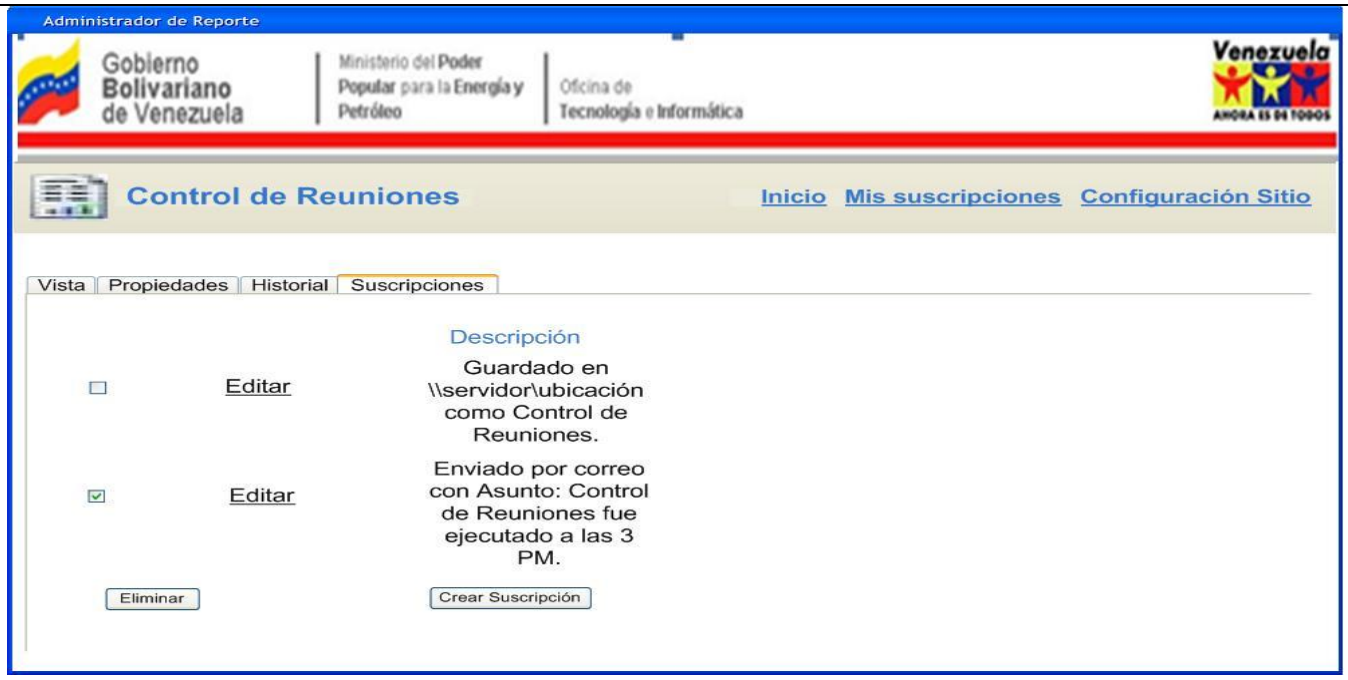
Flujo Normal de Eventos	
Sección "Gestionar Suscripciones"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Mis Suscripciones".	2. El sistema en su interfaz muestra todas las suscripciones que el usuario ha creado. Además muestra las opciones: a) "Modificar Suscripción". b) "Eliminar Suscripción".
Prototipo de Interfaz	

Flujo Normal de Eventos	
Sección "Modificar Suscripción"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Editar" de la suscripción que desea modificar. 3. El usuario introduce los nuevos datos que desea para modificar la suscripción y selecciona "Aceptar".	2. El sistema muestra una interfaz donde permite que el usuario modifique la suscripción. 4. El sistema modifica la suscripción.

Flujo Normal de Eventos	
Sección "Eliminar Suscripción"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la suscripción que desea eliminar y la opción "Eliminar". 3. El usuario selecciona "Aceptar".	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina la suscripción.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
* El usuario selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Modificar Suscripción” se modifica la suscripción deseada. Sección “Eliminar Suscripción” se elimina la suscripción deseada.

Caso de Uso:	**Gestionar Suscripciones Individuales**
Actores:	Usuario (inicia)
Resumen:	El caso de uso inicia cuando el usuario desea modificar o eliminar una suscripción a un reporte.
Precondiciones:	Debe haber suscripciones creadas.
Referencias	R4, 4.2, 4.3, R5
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección “Gestionar Suscripciones Individuales”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona el reporte al que desea modificar o eliminar una suscripción. 3. El usuario selecciona la pestaña “Suscripciones”.	2. El sistema muestra las pestañas “Vista”, “Propiedades”, “Historial”, “Suscripciones”. 4.El sistema muestra las suscripciones , además brinda las opciones: a) Modificar Suscripción. b) Eliminar Suscripción.
Prototipo de Interfaz	



Flujo Normal de Eventos	
Sección "Modificar Suscripción Individual"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Editar" de la suscripción que desea modificar. . 3. El usuario introduce los nuevos datos que desea para modificar la suscripción y selecciona "Aceptar".	2. El sistema muestra una interfaz donde permite que el usuario modifique la suscripción. 4. El sistema modifica la suscripción.
Flujo Normal de Eventos	
Sección "Eliminar Suscripción Individual"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la suscripción que desea eliminar y la opción "Eliminar". 3.El usuario selecciona "Aceptar"	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina la suscripción.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
* El usuario selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Modificar Suscripción” se modifica la suscripción deseada. Sección “Eliminar Suscripción” se elimina la suscripción deseada.

Caso de Uso:	**Gestionar Modelos Semánticos**
Actores:	Administrador (inicia)
Resumen:	El caso de uso inicia cuando el administrador requiere crear o eliminar un modelo semántico.
Precondiciones:	Para crear un modelo semántico, deben existir bases de datos. Para eliminar un modelo semántico, debe haber modelos semánticos creados.
Referencias	R24, 24.1, 24.2
Prioridad:	Crítico

Flujo Normal de Eventos	
Sección “Gestionar Modelos Semánticos”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la carpeta donde desea crear o eliminar un modelo semántico.	2. El sistema muestra el contenido de la carpeta seleccionada, permitiéndole: a) “Crear Modelo Semántico”. b) “Eliminar Modelo Semántico”.

Flujo Normal de Eventos	
Sección “Crear Modelo Semántico”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona en la barra de herramientas la opción “Definir Modelo Semántico”	2. El sistema brinda una interfaz donde muestra las bases de datos del MENPET.

3. El administrador selecciona la base de datos a la cual desea crearle un modelo semántico. 5. El administrador introduce el usuario y contraseña requerida por el sistema y selecciona la opción “Crear Modelo Semántico”.	4. El sistema solicita los datos para acceder a la base de datos y muestra la opción “Crear Modelo Semántico”. 6. El sistema crea automáticamente el modelo semántico deseado.

Prototipo de Interfaz

Flujo Normal de Eventos

Sección “Eliminar Modelo Semántico”

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona el modelo semántico que desea eliminar y la opción “Eliminar” en la barra de herramientas. 3. El administrador selecciona “Aceptar”.	2. El sistema muestra un mensaje de confirmación. 4. El sistema elimina el modelo semántico.

Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
En la Sección “Eliminar Modelo Semántico”: 3. El administrador selecciona “Cancelar” y termina el caso de uso.	
Poscondiciones:	Sección “Crear Modelo Semántico” se obtiene el modelo semántico deseado. Sección “Eliminar Modelo Semántico” se elimina el modelo semántico.

2.5 Prototipo Principal de Interfaz de Usuario.

Un prototipo no funcional es un diseño inicial de la interfaz del software que se utiliza para demostrar los conceptos, probar las opciones de diseño y entender mejor el problema y su solución. Puede revelar errores u omisiones en los requerimientos propuestos, favorece la comunicación entre clientes y desarrolladores, da una primera visión del producto.

El prototipo principal de interfaz de usuario del Sistema de Reportes que se muestra a continuación fue diseñado con la herramienta Microsoft Office Visio 2007, al igual que el resto de los prototipos anteriormente mostrados en las especificaciones de los casos de uso del sistema.

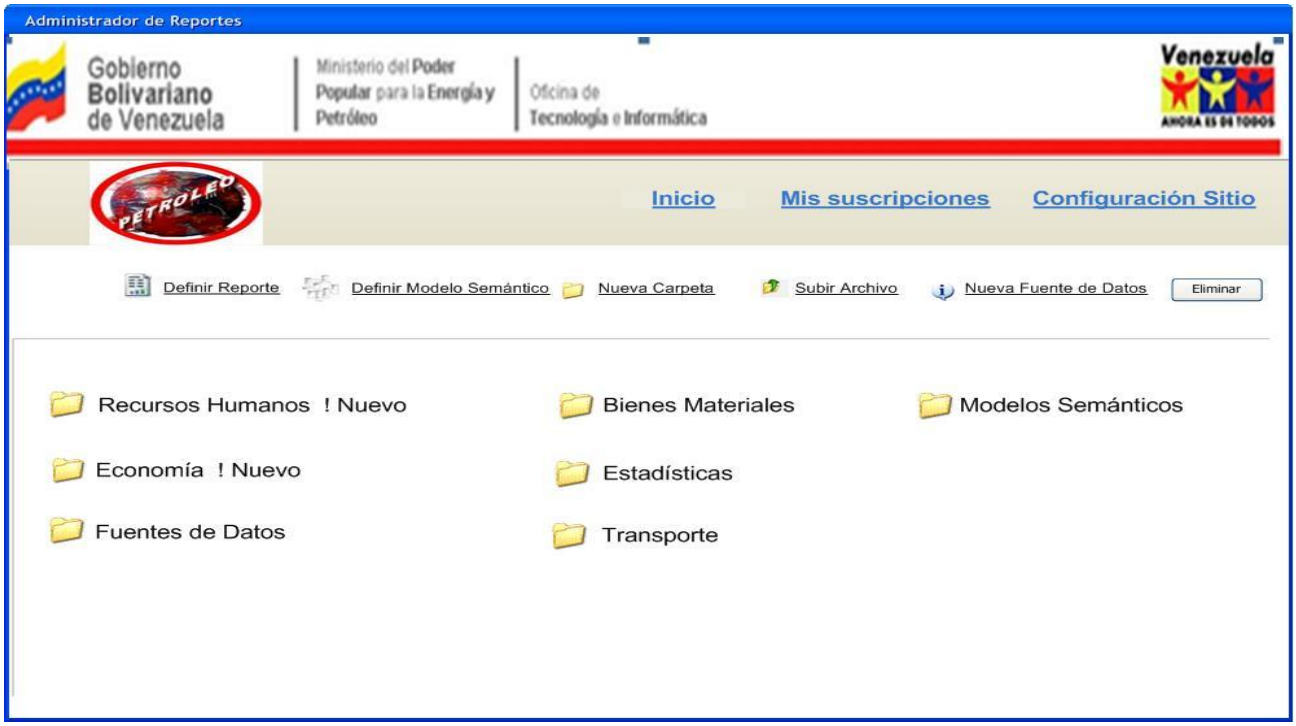


Figura 6: Prototipo Principal de Interfaz de Usuario.

2.6 Conclusiones.

Durante este capítulo se desarrollaron todas las actividades de la IR, generándose los artefactos correspondientes en cada una de ellas, y cumpliéndose de esta manera el objetivo general que impulsó este trabajo.

La utilización de estrategias de captura de requisitos como: la Tormenta de Ideas, las Entrevistas y Sistemas Existentes permitieron captar todas las necesidades de los clientes, lo cual es muy importante pues la aceptación del sistema depende de ello y de cuán bien asista a la automatización del trabajo.

La aplicación de patrones de casos de uso al DCUS facilitó el modelado y la interpretación del mismo.

El uso de la técnica de Prototipos permitió mostrarle al usuario de forma más específica las funcionalidades del sistema propuesto de acuerdo a sus necesidades, así como la técnica Casa de Calidad facilitó verificar que todo requerimiento es implementado a través de algún caso de uso y, que todo caso de uso satisface algún requerimiento; ambas técnicas posibilitaron validar los requerimientos hallados.

Capítulo 3

Análisis de los Resultados.

3.1 Introducción

La mayoría de los desarrolladores de software se preguntan ¿por qué es tan importante medir el proceso de ingeniería del software y el producto que produce? La respuesta es relativamente obvia. Si no se mide, no hay una forma real de determinar si se está mejorando. Y si no se está mejorando, se está perdido.

Lord Kelvin en una ocasión expresó:

Cuando pueda medir lo que está diciendo y expresarlo con números, ya conoce algo sobre ello, cuando no pueda medir, cuando no pueda expresar lo que dice con números, su conocimiento es precario y deficiente: puede ser el comienzo del conocimiento, pero en sus pensamientos, apenas está avanzando hacia el escenario de la ciencia.

La medición es fundamental para la Ingeniería del Software pues permite una visión más profunda proporcionando un mecanismo para la evaluación objetiva.

Las métricas del software, al igual que la medición, son muy importantes ya que proporcionan una manera cuantitativa de valorar la calidad de los atributos internos del producto, permitiendo por tanto al ingeniero valorar la calidad antes de construir el producto.

Durante este capítulo se aplicarán diferentes métricas a los artefactos obtenidos durante el desarrollo de las actividades de la Ingeniería de Requerimientos. Dichas métricas ayudarán a controlar o validar los requisitos que se obtuvieron del cliente y los casos de uso que se obtuvieron en el Modelo del Sistema.

3.2 Métricas.

El IEEE Standard Glossary of Software Engineering Terms define métrica como <<una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado>>.

Las métricas del software se refieren a un amplio elenco de mediciones para el software de computadora. Relatan de alguna forma las medidas individuales sobre algún aspecto (por ejemplo: el número medio de errores encontrados por revisión).

La medición se puede aplicar al proceso del software con el intento de mejorarlo sobre una base continua. Además puede utilizarse en el proyecto del software para ayudar en la estimación, el control de calidad, la evaluación de productividad y el control de proyectos.

Se define calidad como la concordancia con los requisitos funcionales y de rendimiento, explícitamente establecidos, los estándares de desarrollo explícitamente documentados y las características implícitas que se esperan de todo software desarrollado profesionalmente. (Pressman, 2005.)

3.3 Métricas de la Calidad de la Especificación de Requisitos.

Davis y sus colegas proponen una lista de características que pueden emplearse para valorar la calidad de la especificación de requisitos: especificidad (ausencia de ambigüedades), compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. Además los autores apuntan que las especificaciones de alta calidad deben estar almacenadas electrónicamente, ser ejecutables o al menos interpretables, anotadas por importancia y estabilidad relativa, con su versión correspondiente, organizadas, con referencias cruzadas y especificadas al nivel correcto de detalle.

Aunque muchas de las características anteriores parecen ser de naturaleza cualitativa, Davis sugiere que todas puedan representarse usando una o más métricas.

Por ejemplo, se asume que hay $\mathbf{n_r}$ requisitos en una especificación, tal como:

$$\mathbf{N_r = n_f + n_{nf}}$$

Donde $\mathbf{n_f}$ es el número de requisitos funcionales y $\mathbf{n_{nf}}$ es el número de requisitos no funcionales.

3.3.1 Especificidad.

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

$$\mathbf{Q_1 = n_{ui} / n_r}$$

Donde $\mathbf{n_{ui}}$ es el número requisitos para los que todos los revisores tuvieron interpretaciones idénticas. Cuanto más cerca de 1 esté el valor de $\mathbf{Q_1}$ menor será la ambigüedad de la especificación. (Pressman, 2005.) (Davis,1993)

Para evaluar la métrica de la especificidad de los requisitos se realizaron dos revisiones, en las cuales se consultaron a varios revisores. Estas revisiones tenían como principal objetivo obtener requisitos con el menor nivel de ambigüedad y la mayor claridad posible, para lograr describir de forma más exacta las necesidades del cliente.

A continuación se exponen los resultados de las revisiones realizadas:

$$N_r = n_f + n_{nf}$$

$$N_r = 90 + 23=113$$

Revisión 1: En esta revisión algunos de los requisitos funcionales y no funcionales presentaron problemas de redacción y de ambigüedad. Por lo que de una totalidad de 113 requisitos entre funcionales y no funcionales los revisores tuvieron la misma interpretación para 109 de ellos.

$$Q_1 = 109 / 113=0.96$$

Revisión 2: Se hicieron algunos cambios en los requisitos para corregir los problemas presentados en la revisión anterior, obteniéndose así una segunda versión de los mismos. Esta versión fue entregada a los revisores quienes tuvieron la misma interpretación para 112 requisitos, por lo que Q tiene un valor de 0.99 que refleja la especificidad de los requisitos capturados.

$$Q_1 = 112 / 113=0.99$$

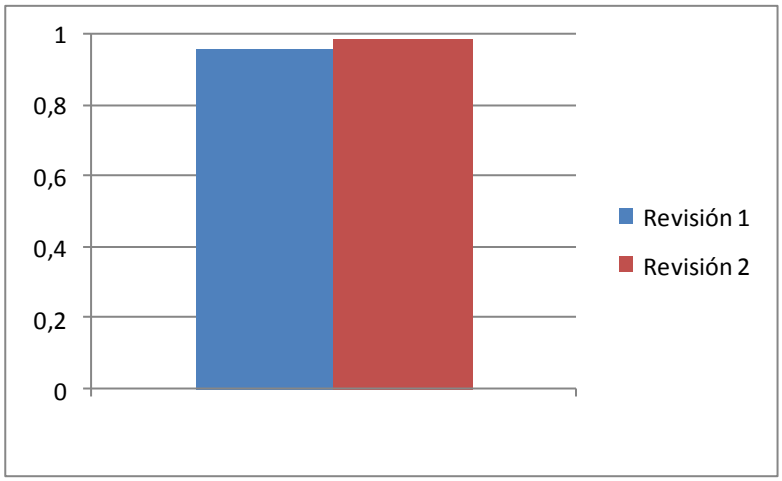


Figura 7: Gráfico de Control de la Calidad de la Especificación de Requisitos.

3.3.2 Grado de validación de los requisitos.

Para incorporar los requisitos no funcionales a una métrica global completa, se debe considerar el grado de validación de los requisitos:

$$Q_3 = n_c / (n_c + n_{nv})$$

Donde n_c es el número de requisitos que se han validado como correctos y n_{nv} el número de requisitos que no se han validado todavía.

$$Q_3 = 112 / (112 + 0) = 1$$

3.4 Métricas de los Casos de Uso.

Uso apropiado de la técnica

La técnica de casos de uso se debe usar cuando el requisito funcional que se representa requiere la interacción del sistema con un actor, bien sea un usuario o bien otro elemento externo al sistema en construcción. La cuestión asociada a esta característica de calidad es: ¿Representa el caso de uso un comportamiento que requiere la participación de otro elemento externo al propio sistema?

Todos los casos de uso representan requisitos funcionales que requieren la interacción del sistema con un actor por lo que esta técnica fue usada.

Completitud

Adaptando la definición de [Davis *et al.* 1993] a los casos de uso, un caso de uso es completo si especifica todo lo que deben hacer el actor y el sistema (externamente) para alcanzar el objetivo del caso de uso y si se consideran todas las respuestas del sistema a situaciones anormales. Para comprobar si un caso de uso es completo se propone la siguiente lista de preguntas:

- ¿Hay respuestas a todas las peticiones que el actor del caso de uso hace al sistema y viceversa?
- ¿Se contemplan todos los posibles escenarios para poder alcanzar el objetivo del caso de uso?
- ¿Se especifican todas las secuencias alternativas a la secuencia normal?
- ¿Se contemplan todas las posibles excepciones a la secuencia normal?

Al aplicar esta métrica se puede observar, que la completitud de los casos de uso del sistema se garantiza en un 100%.

Comprensibilidad

Un caso de uso es comprensible si todos los tipos de lectores (cliente, usuario, jefe de proyecto, desarrollador o responsable de pruebas) pueden entenderlo fácilmente con una mínima explicación del autor [Davis et al. 1993]. Para ver si un caso de uso es comprensible, se propone la siguiente lista de cuestiones:

. ¿Es posible leer el caso de uso sin volver atrás en repetidas ocasiones?

¿Es difícil seguir la secuencia normal del caso de uso por la presencia de las relaciones include o extend?

¿Es difícil seguir la secuencia de pasos por la existencia de demasiados pasos alternativos?

¿Se han desglosado demasiado los pasos de algún actor o del sistema provocando que el caso de uso avance a un ritmo muy lento?

¿Aparecen pasos condicionales para expresar que el sistema comprueba una situación que permite al caso de uso continuar su realización?

Cuando se aplica esta métrica a los casos de uso obtenidos en el sistema se observa que se cumple en un 100 %.

Concisión

Un caso de uso es conciso si no incluye información superflua o innecesaria. Para saber si un caso de uso es conciso se deben comprobar las siguientes cuestiones:

¿Podría el caso de uso ser expresado con menos palabras?

¿Existen elementos que se puede obviar o aparecen anotaciones innecesarias y que dificultan la lectura del caso de uso?

¿Aparecen demasiadas interacciones entre el actor principal del caso de uso y otros elementos del entorno?

Los casos de uso del sistema cumplen esta métrica en un 100%.

No Trivialidad

Un caso de uso es no trivial si su secuencia de pasos conduce al actor a conseguir el objetivo que persigue la realización del caso de uso. Para comprobar si un caso de uso es no trivial la siguiente lista de cuestiones debe ser comprobada:

¿Expresa el nombre del caso de uso un objetivo de un usuario que el sistema debe implementar?

¿Conduce el caso de uso al actor a conseguir alguno de sus objetivos sin representar un conjunto de interacciones triviales?

Los resultados de la aplicación de esta métrica a los casos de uso indican su cumplimiento al 100%.

Heurística basada en NOAS/NOS

Esta métrica es una relación entre las acciones del actor y las totales. (**NOAS**: según sus siglas en inglés, *Number of Actor Steps*, número de pasos del actor y **NOS**: *Number of Steps*, número de pasos.

Esta heurística se basa en la idea de que un caso de uso sirve básicamente para expresar una interacción actor–sistema. Por ello, el número de pasos de actor y el de pasos de sistema deben estar en torno al 50%, considerando también la posibilidad de que existan pasos de inclusión o extensión en los que se realice otro caso de uso.

Las situaciones que llevan a esta métrica fuera del rango habitual son:

• El hecho de obviar la participación del sistema, por lo que el caso de uso resulta incompleto. Es la situación más habitual.

• El hecho de haber desglosado demasiado las acciones de un actor determinado. En este caso aparecen varios pasos seguidos del mismo actor, lo cual se podría haber evitado uniéndolos en uno solo, separando las acciones por comas en el texto del paso.

• El hecho de incluir interacciones de actores con el entorno del sistema o con otros actores. Este hecho, que en principio no puede considerarse un defecto, sino más bien una información interesante aunque no esencial del caso de uso, será un defecto cuando se produzca con excesiva frecuencia. (Una propuesta para la verificación de requisitos basada en métricas., 2004)

Al aplicar la heurística basada en NOAS/NOS a los casos de uso del sistema se observa que se cumple esta métrica en un 99 % y se obtiene un promedio de un 49.5 %. El valor óptimo es alrededor de un 50 % por lo que el valor alcanzado se encuentra en el rango aceptable.

Todos los resultados anteriormente obtenidos se muestran en la siguiente gráfica.

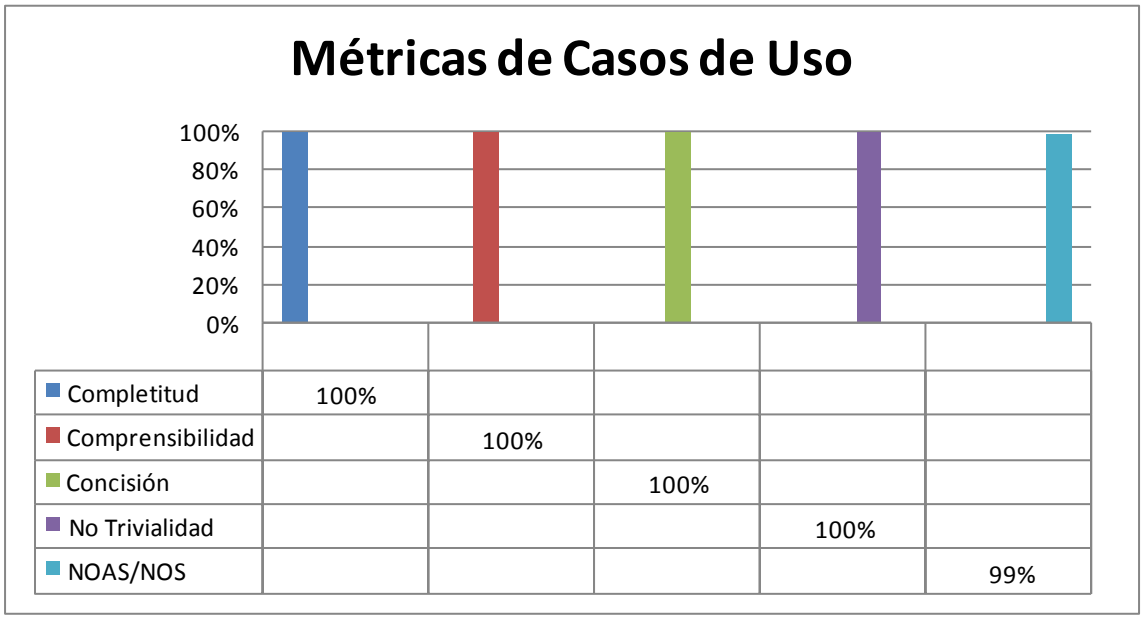


Figura 8: Gráfico de Control de la Calidad de los Casos de Uso.

3.4.1 Comprobación De La Calidad Del Modelo De Casos De Uso Del Sistema.

Para evaluar la calidad de la funcionalidad a partir del diagrama de casos de uso del sistema, se aplicó un Modelo de métricas Orientado a Objetos (OO) de la Universidad EAFIT de Medellín en Colombia. Este modelo tiene en cuenta cuatro atributos fundamentales para medir la calidad: consistencia, correctitud, completitud y complejidad.

Completitud*:* Grado en que se ha logrado detallar todos los casos de uso relevantes.

Consistencia*:* Grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema

Correctitud: Grado en que las interacciones actor / sistema soportan adecuadamente el proceso del negocio

Complejidad: Grado de claridad en la presentación de los elementos que describen el contexto y funcionalidad del sistema.

	Factores	Métricas Asociadas

Completitud	Factor 1. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 1: Número de requisitos omitidos por caso de uso. Métrica 2: Número de casos de uso que tienen requisitos omitidos. Acción sugerida: Revisar la lista de requisitos para determinar cuáles serán apoyados por cada caso de uso.
	Factor 2. ¿Existen requisitos que no han sido considerados en algún caso de uso?	Métrica 3: Número de requisitos que no son considerados en ningún caso de uso.
	Factor 3. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia (primario / secundario / opcional)?	Métrica 4: Número de casos de uso que no han sido clasificados. Acción sugerida: Hacer reuniones con los usuarios para analizar y priorizar los requisitos de acuerdo a su relevancia.
	Factor 4. ¿Se presenta una descripción detallada (descripción extendida esencial) de todos los casos de uso del sistema?	Métrica 5: Número de casos de uso que no poseen una descripción extendida. Acción sugerida: Interactuar con el usuario para realizar la definición extendida del caso de uso que sea consistente con la definición a alto nivel.
	Factor 5. ¿Están todas las acciones del flujo de eventos redactadas en función del	Métrica 6: Número de acciones del flujo de eventos que no están redactadas en función del responsable. Acción sugerida: Revisar las

	responsable?	responsabilidades tanto del actor (actores) como del sistema.
	Factor 6. ¿Se describen las condiciones de excepción relevantes que debe contemplar cada flujo de eventos?	Métrica 7: Número de casos de uso que no describen condiciones de excepciones relevantes.
	Factor 7. ¿Se presenta una descripción resumida (descripción de alto nivel) de todos los casos de uso?	Métrica 8: Número de casos de uso que no tienen descripción resumida. Acción sugerida: Completar la descripción resumida del caso de uso.
	Factor 8. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 9: Número de casos de uso que tienen un nombre incorrecto. Acción sugerida: Modifique el nombre del caso de uso de tal manera que signifique una acción desde el punto de vista del usuario.
	Factor 9. ¿Representa el caso de uso una interacción observable por un actor?	Métrica 10: Número de casos de uso que no representan una interacción observable por un actor. Acción sugerida: Elimine el caso de uso e incorpore su funcionalidad como una responsabilidad del sistema dentro de otro caso de uso.

Consistencia	Factor 10. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 11: Grado de adecuación de la descripción del flujo de eventos para un caso de uso. Acción sugerida: Revise la descripción para que sea definida en el lenguaje del usuario. Asegúrese de definir el responsable de la acción. Establezca claramente las acciones de inicio y fin del caso de uso.
	Factor 11. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 12: Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema. Acción sugerida: Complete la definición del caso de uso incluyendo la acción fuera del sistema que da inicio al caso de uso o la condición interna que el sistema tiene que controlar para dar inicio al caso de uso.
	Factor 12. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Métrica 13: Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos. Acción sugerida: Estructure el caso de uso de manera que separe su funcionalidad básica (caso de uso base) de la funcionalidad repetitiva o

		alternativa. Si hay pasos repetitivos forme un caso de uso que lo incluye y los pasos alternativos formen un caso de uso que lo extienda.
Correctitud	Factor 13. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 14: Número de casos de uso en que los requisitos representados no son comprensibles por el usuario. Acción sugerida: Discuta con el usuario la interacción que describe el caso de uso y ajuste dicha descripción de manera que sea comprensible por el usuario.
	Factor 14¿Existe para cada caso de uso por lo menos un usuario responsable?	Métrica 15: Número de casos de uso que no tienen un usuario responsable. Acción sugerida: Analice la responsabilidad que representa el caso de uso y acuerde con los usuarios cuál es el responsable directo de éste.
	Factor 15. ¿Las interacciones definidas describen la funcionalidad requerida del sistema?	Métrica 16: Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.
	Factor 16. ¿Las interacciones definidas introducen mejoras al proceso actual?	Métrica 17: Número de casos de uso que deben ser modificados para mejorar el proceso actual. Acción sugerida: Analice la situación descrita en la interacción y estudie la manera

		de mejorar el proceso con el uso de la tecnología informática.
Complejidad	Factor 17. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 18: Número de elementos del diagrama que requieren reubicación. Acción sugerida: Modifique la ubicación de los elementos del diagrama de manera que los elementos relacionados se encuentren lo más cercano posible.

Tabla 4: Atributos y factores para la medición de la calidad del Diagrama de Casos de Uso del Sistema.

Se le realizaron dos revisiones al Diagrama de Casos de Uso del Sistema.

En el siguiente gráfico se muestran los resultados obtenidos de las métricas aplicadas en la primera revisión.

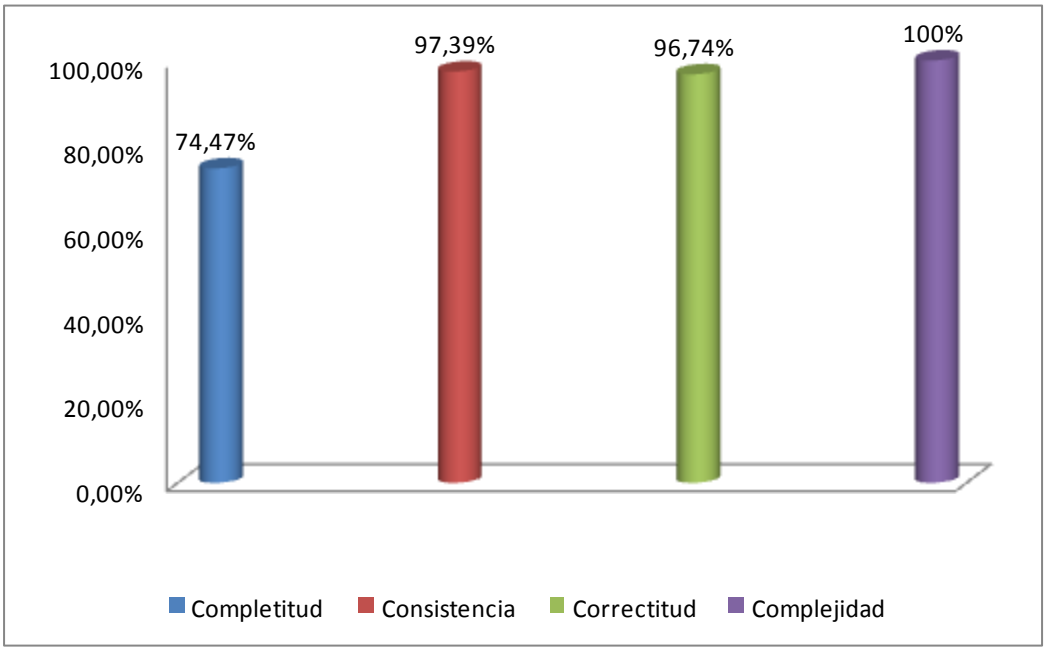


Figura 9: Primera revisión: Gráfico de Control de la Calidad del Modelo de Casos de Uso.

Observaciones:

Para el atributo Completitud

No se encontraban representados todos los requisitos que justifican la funcionalidad de los casos de uso. Setenta de ellos se encontraban bien definidos, por lo que dicho factor representa en esta revisión un valor igual al 77.78 % del total de requisitos que deben justificar la funcionalidad de los casos de uso.

No se presentaba una descripción detallada de todos los casos de uso del sistema sino sólo de 10 de estos representando el 43.48% del total de casos de uso.

No se presentó una descripción resumida (descripción de alto nivel) de los casos de uso del sistema ya que se tienen las descripciones detalladas de los casos de uso, que brindan una información más profunda y a su vez le permiten al usuario un mejor entendimiento de estos.

Todos los requisitos han sido considerados en algún caso de uso. Las acciones del flujo de eventos de todos los casos de uso se encuentran redactadas en función del responsable y se describen las condiciones de excepción relevantes que debe contemplar cada flujo de eventos. Los casos de usos han sido clasificados de acuerdo a su relevancia (Crítico, Secundario, Opcional). La calidad del atributo Completitud tiene un valor de 74.47% al sumar todos estos resultados y dividirlos por la cantidad total de factores que incluye este atributo.

Para el atributo Consistencia

En todos los casos de uso el flujo de eventos está adecuadamente redactado en el lenguaje del usuario y se inicia con la descripción de una acción originada por un actor, además existe una adecuada separación entre el flujo básico de eventos y los flujos alternos. Cada uno de ellos representa una interacción observable por un actor. Solo veinte casos de uso estaban representados por una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario, representando un valor para este factor igual al 86.96% del total de casos de uso. La calidad del atributo Consistencia tiene un valor de 97.39% al sumar todos estos resultados y dividirlos por la cantidad total de factores que incluye este atributo.

Para el atributo Correctitud

Todos los casos de uso poseen al menos un usuario responsable, representan requisitos comprensibles por el usuario e introducen mejoras al proceso actual. Sólo veinte de ellos describen la

funcionalidad requerida del sistema representando un valor para este factor del 86.96% del total de casos de uso. La calidad del atributo Correctitud tiene un valor de 96.74% al sumar todos estos resultados y dividirlos por la cantidad total de factores que incluye este atributo.

Para el atributo Complejidad

Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación, representando un 100% de la calidad del atributo.

A continuación se muestra la gráfica de los resultados obtenidos en la segunda revisión.

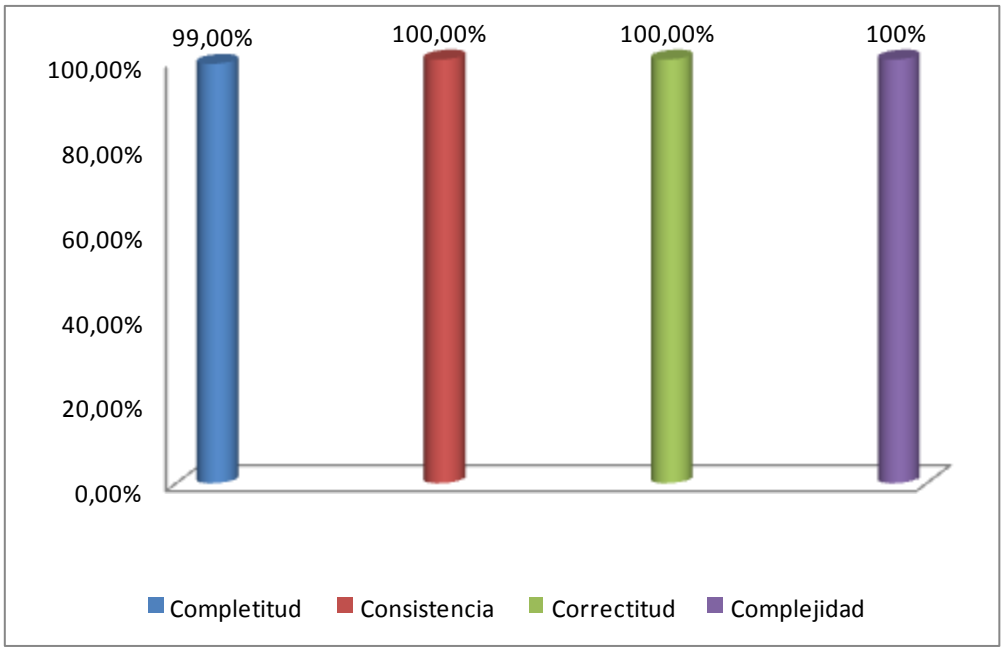


Figura 10: Segunda revisión: Gráfico de Control de la Calidad del Modelo de Casos de Uso.

Observaciones:

No se realizó la descripción detallada (descripción extendida esencial) de todos los casos de uso del sistema.

Los resultados de la evaluación del Diagrama de Casos de Uso demuestran que se ha obtenido un modelo con un alto grado de funcionalidad, que cumple con los atributos principales que definen la calidad, representando un 99,75 %.

3.5 Métricas de diseño de interfaz.

La calidad del diseño de una interfaz de usuario se puede comprobar aplicando algunas métricas como son: la existencia de un patrón único para todas las interfaces, la cual consiste en evaluar si las diferentes pantallas o interfaces del sistema cumplen con un solo patrón de diseño. Además se puede evaluar el tamaño de letra y los colores utilizados para la elaboración de estas interfaces.

Al aplicar estas métricas a los prototipos de interfaces no funcionales del Sistema de Reportes para el MENPET se cumplen en un 100%.

3.6 Opinión del cliente.

Ciudad Caracas, 21 de Abril de 2008.

A: Lic. Ridosbey Milian Iglesias
Gerente del Proyecto

Estimado Ridosbey,

Nos complace comunicarle que el proyecto para el desarrollo del sistema de gestión de reportes, el cual hemos llamado de forma abreviada Olympia, marcha muy bien.

En estos momentos el equipo de trabajo, ha concluido y entregado el segundo hito de pago correspondiente al levantamiento de requisitos del sistema, el cual permite la puesta en marcha en la construcción del sistema, tal y como se les había pedido.

Nuestra valoración es altamente positiva, ya que el desarrollo de esta futura herramienta nos permitirá gestionar los reportes en el Ministerio del Poder Popular para la Energía y Petróleo, así como poder satisfacer las demandas constantes de nuevos reportes en las aplicaciones emergentes en nuestra entidad.

En estos momentos el equipo de trabajo se encuentra en fase de terminación de algunos documentos para finiquitar el flujo de Requisitos en el desarrollo del proyecto para la primera iteración del mismo.

Sin más, se despide de Ud., atentamente,

Ing. Carlos Caro
Ministerio del Poder Popular Para La Energía y Petróleo.

3.7 Conclusiones.

Para descubrir los requerimientos de una aplicación se debe obtener en profundidad como es que este sistema afectará a las partes del negocio que estarán involucradas y cómo puede contribuir a lograr las

metas de la empresa, además de comprender las necesidades y restricciones de los usuarios del sistema.

Las actividades involucradas en el descubrimiento de los requerimientos del sistema son muy complejas por lo que se debe evaluar la calidad de los atributos internos del producto de manera sistemática, y así determinar si se está mejorando.

Para poder medir el proceso de ingeniería del software y el producto que produce se deben aplicar métricas, y precisamente ese fue el objetivo del presente capítulo.

La aplicación del Modelo de Métricas OO permitió obtener una medida cuantitativa de la calidad del DCUS, el mismo alcanzó una calificación máxima del 99,75 % teniendo en cuenta los atributos fundamentales utilizados para su medición: consistencia, correctitud, completitud y complejidad. Este resultado muestra el alto nivel de funcionalidad del DCUS, el cual no depende únicamente del cumplimiento de los factores que plantean estos atributos sino también de la correcta especificación de los requisitos del sistema. Para medir la calidad de la especificación de requisitos se aplicaron una serie de métricas que demostraron que estos son comprensibles, posibles de verificar, concisos, consistentes y se encuentran libres de ambigüedades, características que todo requisito debe tener. Se aplicaron también métricas de diseño de interfaz de usuario, y de casos de uso para saber si son completos, comprensibles, entre otras características.

Para verificar que el desarrollo de las actividades de la Ingeniería de Requerimientos garantizó un entendimiento entre clientes y desarrolladores respecto a los requisitos que debe cumplir el software, todos los artefactos generados fueron entregados a los clientes para que los evaluaran y expresaran el nivel de aceptación de la aplicación. Dicha aceptación por parte de los clientes se encuentra plasmada en este capítulo.

Conclusiones Generales

El desarrollo de las actividades de la Ingeniería de Requerimientos permitió mejorar la comunicación entre equipos, pues la especificación de requisitos representa una forma de consenso entre clientes y desarrolladores, necesario para el éxito del proyecto.

Se aseguró que usuarios finales y desarrolladores tuvieran un entendimiento respecto a la organización facilitando la obtención de los requisitos del Sistema, que verdaderamente responden a las necesidades y expectativas de los clientes. Se aplicaron varias técnicas de la Ingeniería de Requerimientos para la captura de requisitos como son: el análisis de sistemas ya desarrollados que están relacionados con la aplicación a ser construida, la Tormenta de ideas que suele ofrecer una visión general de las necesidades del sistema pero no es muy útil para obtener detalles concretos del mismo, por lo que sólo se aplicó en los primeros encuentros con el cliente. Además se utilizó la Entrevista, de preguntas abiertas en la etapa de identificación de requerimientos, lográndose que los encuestados respondieran con su propia terminología y revelaran así mayor información, y las cerradas en la etapa de análisis y negociación a la hora de establecer el criterio de priorización de los casos de uso.

La selección de RUP, como metodología de desarrollo de software, permitió obtener los artefactos necesarios para la modelación del Sistema de Reportes a partir de la captura de requisitos.

La herramienta CASE Visual Paradigm Enterprise Edition permitió la modelación visual del sistema, por lo que se logró una mejor comunicación entre los miembros del equipo, al usar un lenguaje gráfico.

La realización del glosario de términos al igual que el modelo del dominio posibilitó que los usuarios, clientes y desarrolladores utilizaran un vocabulario común, siendo esto muy importante a la hora de construir un sistema software de cualquier tamaño, pues los ingenieros deben “fundir” el lenguaje de todos los participantes en uno solo consistente.

El DCUS, las especificaciones de los CUS y el diseño de los prototipos no funcionales de interfaz de usuario, muestran una descripción detallada de las funcionalidades del sistema.

La utilización de patrones de casos de uso facilitó la modelación del sistema y posibilitó la ubicación adecuada de los elementos del diagrama de casos de uso del sistema de manera que los desarrolladores interpretaran el sistema según lo deseado por el cliente.

La aplicación de métricas para evaluar la calidad de la Especificación de Requisitos y de la funcionalidad a partir del diagrama de casos de uso del sistema mediante los atributos completitud, consistencia, correctitud y complejidad, mostró resultados satisfactorios.

La aceptación del sistema por parte del cliente se logró, fundamentalmente porque la IR obliga a los mismos a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema durante todo el desarrollo del proyecto. Dicha aceptación sirvió como constancia final para disminuir los posibles cambios en los requisitos, cumpliéndose así el objetivo de este trabajo.

Recomendaciones

El constante refinamiento y redefinición de requerimientos, porque aunque se le dio cumplimiento al objetivo general de este trabajo, el mismo sólo forma parte de la primera fase o versión del proyecto, y la metodología de desarrollo de software utilizada (RUP) plantea, dentro de sus características fundamentales, que el proceso de desarrollo de software debe ser iterativo e incremental.

Continuar con las investigaciones para añadir nuevas funcionalidades al sistema y obtener mejoras en futuras versiones, logrando adecuarlo cada vez más a las necesidades del MENPET.

Continuar el ciclo de desarrollo, a partir de los artefactos generados durante el proceso de Ingeniería de Requerimientos, para implementar el sistema.

Utilizar una Herramienta CASE para la gestión de los requisitos de software de los restantes módulos del MENPET que esté basada en la metodología de desarrollo de software RUP e integrada dentro de la Herramienta Visual Paradigm.

Referencias Bibliográficas

Arango, J. 2002. Universidad EAFIT. *http://www.eafit.edu.co/.* [En línea] 2002. [Citado el: 26 de enero de 2008.] http://www.eafit.edu.co/tda/boletin/TORMENTA%20DE%20IDEAS.htm.

Arboleda Jiménez, Hugo F. 2005. Asociación Colombiana de Ingenieros de Sistemas(ACIS). *http://www.acis.org.co/.* [En línea] 2005. [Citado el: 9 de diciembre de 2007.] http://www.acis.org.co/index.php?id=551.

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

Canós, José H, Penadés, María Carmen y Letelier, Patricio. 2003. Ingenieria del Software y Sistemas de Informacion. *http://issi.dsic.upv.es/.* [En línea] 2003. [Citado el: 8 de diciembre de 2007.] http://issi.dsic.upv.es/tallerma/actas.pdf.

Cantone, Dante. 2006. *La biblia del programador: Implementación y debugging.* s.l. : MP Ediciones, 2006. 9789872299576.

Cataldi, Zulma. 2000. Facultad de Ingeniería.Universidad de Buenos Aires. *http://www.fi.uba.ar/.* [En línea] 2000. [Citado el: 18 de febrero de 2008.] http://www.fi.uba.ar/laboratorios/lsi/cataldi-tesisdemagistereninformatica.pdf.

Checkland, Peter. 1989. *"An Application of Soft Systems Methodology. Rational Analysis for a Problematic World".* Chapter 5. . New York : John Wiley & Sons, 1989.

Colmenero González, José Ignacio, Pérez Herrero, Carlos y Bravo Sánchez, José Luis. 2007. Kybele.Grupo de Investigación.Departamento de Lenguajes y Sistemas Informáticos II. *http://kybele.escet.urjc.es/.* [En línea] 2007. [Citado el: 1 de abril de 2008.] http://kybele.escet.urjc.es/documentos/HC/Exposiciones/MagicDrawUML.pdf.

Computación, Departamento de. 2006. Facultad de Ciencias Exactas, Físico-Químicas y Naturales.Universidad Nacional de Río Cuarto. *http://dc.exa.unrc.edu.ar/.* [En línea] 2006. [Citado el: 6 de febrero de 2008.] http://dc.exa.unrc.edu.ar/nuevodc/materias/proyecto/Repositorio%20de%20archivos/BPMN-UML-6Trans.pdf.

David, M. 1998. University of Waterloo. Management Sciences. *http://www.mansci.uwaterloo.ca/.* [En línea] 1998. [Citado el: 4 de febrero de 2008.] http://www.mansci.uwaterloo.ca/~msci432/Notes/F_Fish_bone.htm.

Davis, A. 1993. *Identifying and Measuring Quality in a Software Requirements Specification.* s.l. : Proc. 1st Intl. Software Metrics Symposium, 1993.

Davyt Dávila, Nicolás. 2001. *Ingeniería de Requerimientos:Una Guía para Extraer, Analizar, Especificar y Validar los Requerimientos de un Proyecto.* [pdf] 2001.

Fernández Escribano, Gerardo. 2002. Departamento de Sistemas Informáticos(DSI). Universidad de Castilla- La Mancha. *http://www.info-ab.uclm.es/.* [En línea] 2002. [Citado el: 8 de diciembre de 2007.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

Fernández-Medina Patón, Eduardo. 2006. Grupo Alarcos.Universidad de Castilla-La Mancha.Sede en Escuela Superior de Informática de Ciudad Real. *http://alarcos.inf-cr.uclm.es/.* [En línea] 2006. [Citado el: 16 de febrero de 2008.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema03.pdf.

García Rubio, Félix Óscar y Bravo Santos, Crescencio. Grupo Alarcos.Universidad de Castilla-La Mancha. *http://alarcos.inf-cr.uclm.es/.* [En línea] [Citado el: 9 de diciembre de 2007.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/tema3_1xh.pdf.

Giraldo, Luis y Zapata, Yuliana. 2005. [En línea] 2005. [Citado el: 1 de abril de 2008.] http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17.

González Campos, Saúl y Fernández Martínez, Luis Felipe. 2006. Universidad Autónoma de Ciudad Juárez. *http://www.uacj.mx/.* [En línea] 2006. [Citado el: 8 de diciembre de 2007.] http://www.uacj.mx/IIT/CULCYT/mayo-agosto2006/8ArtProg.pdf.

Griselda Báez, M. y Barba Brunner, Silvia I. 2001. Workshop en Ingeniería de Requerimientos. *http://wer.inf.puc-rio.br/.* [En línea] 2001. [Citado el: 6 de febrero de 2008.] http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER01/baez.pdf.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2002. *EL Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : EDUNIV.Editorial Universitaria., 2002. 959-16-0343-6.

Herrera J, Lizka Johany. 2003. [En línea] 2003. [Citado el: 18 de febrero de 2008.] http://www.monografias.com/trabajos6/resof/resof.shtml.

Hidalgo, Mauricio. SQL MAX Connections. *http://www.sqlmax.com/.* [En línea] [Citado el: 28 de febrero de 2008.] http://www.sqlmax.com/reportin_services1.asp.

Instituto Nacional de Estadística e Informática. 1999. Instituto Nacional de Estadística e Informática(INEI). *http://www.inei.gob.pe/.* [En línea] 1999. [Citado el: 9 de diciembre de 2007.] http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

Ishikawa, K. 1969. [En línea] 1969. [Citado el: 4 de febrero de 2008.] http://imedia.vuse.vanderbilt.edu/mt322/library2/ishikawa.htm.

Jacobson, Ivar. 1992. *Object Oriented Software Engineering. A Use Case Driven Approach.* s.l. : Addison Wesley, 1992.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley, 2000.

—. 2004. *El Proceso Unificado de Desarrollo de Software.* La Habana : Félix Varela, 2004.

Kendall, Kenneth E y Kendall, Julie E. 1997. *Análisis y Diseño de Sistemas.* Tercera Edición. México : Prentice Hall, 1997.

Koch, N y Escalona, M J. Universidad Católica del Maule.Escuela de Ingeniería Civil Informática. *http://www.eici.ucm.cl/.* [En línea] [Citado el: 15 de enero de 2008.] http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/Ing_Requisitos_para_Web.pdf.

Komer, P. 1993. *Dirección de la Mercadotecnia.* s.l. : Prentice Hall., 1993.

Kotonya, G y Sommerville, I. 1998. *Requirements Engineering. Processes and techniques.* New York : John Wiley & Sons, 1998.

La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. **Arias Chaves, Michael. 2006.** Costa Rica : s.n., 2006, Revista InterSedes.

La Web Semántica como apoyo a la Gestión del Conocimiento y al Modelado Organizacional. **Barceló Valenzuela, Mario, Sánchez Schmitz, Guzmán Gerardo Alfonso y Pérez Soltero, Alonso. 2006.** 12 de abril de 2006, Revista Ingeniería Informática.

Larman, Craig. 1999. *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999.

López Barrio, C. 2005. [En línea] 2005. [Citado el: 8 de diciembre de 2007.] http://www-lsi.die.upm.es/~carreras/ISSE/programacion_extrema_1.x2.pdf.

Marcela Daniele. 2007. Departamento de Computación.Universidad Nacional de Río Cuarto. *http://dc.exa.unrc.edu.ar/.* [En línea] 2007. [Citado el: 8 de diciembre de 2007.] http://dc.exa.unrc.edu.ar/nuevodc/materias/sistemas/2007/TEORICOS/TEORIA_1_Introduccion_AyDS 2007.pdf.

Marcelo Hernán, Schenone. 2004. Facultad de Ingeniería.Universidad de Buenos Aires. *http://www.fi.uba.ar/.* [En línea] 2004. [Citado el: 8 de diciembre de 2007.] http://www.fi.uba.ar/materias/7500/schenone-tesisdegradoingenieriainformatica.pdf.

Mendoza Sánchez, María A. 2004. Informatizate. *http://www.informatizate.net/.* [En línea] 2004. [Citado el: 8 de diciembre de 2007.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

Mendoza, Luis Eduardo. Universidad Simón Bolívar.Departamento de Procesos y Sistemas. Cuerpo de Profesores. *http://prof.usb.ve/.* [En línea] [Citado el: 9 de diciembre de 2007.] http://prof.usb.ve/lmendoza/Documentos/PS-6116/Teoria%20PS6116%20O-O%20y%20RUP.pdf.

Obeso Agüera, Ivan. Web EUITIO.Escuela Universitaria de Informática de Oviedo. *http://www.euitio.uniovi.es/.* [En línea] [Citado el: 1 de abril de 2008.] http://petra.euitio.uniovi.es/~i2133798/hd/archivos/disenio/estudioHerramientasDisenio.pdf.

Overgaard, Gunnar and Palmkvist, Karin. 2004. *Use Cases Patterns and Blueprints.* s.l. : Addison Wesley Professional, 2004. 0-13-145134-0.

Palacio, Juan. 2005. Navegapolis.net. *http://www.navegapolis.net.* [En línea] 2005. [Citado el: 9 de diciembre de 2007.] http://www.navegapolis.net/files/articulos/gestion_y_procesos.pdf.

Politécnica, Facultad. 2005. Facultad Politécnica.Universidad Nacional de Asunción. *http://www.pol.una.py/.* [En línea] 2005. [Citado el: 10 de diciembre de 2007.] http://www.pol.una.py/archivos/IngeInfo/ingeSoftI/MaterialPrimeraC.pdf.

Pressman, Roger S. 2005.. *Ingeniería del Software. Un enfoque práctico.* Quinta Edición . La Habana : Félix Varela, 2005.

Rational Software Corporation y Microsoft Corporation. 1998. IBM. *http://www.ibm.com/.* [En línea] 1998. [Citado el: 8 de diciembre de 2007.] http://www.rational.com/uml/papers.

Robertson, S y Robertson, J. 1999. *Mastering the Requirements Process.* s.l. : Pearson., 1999.

Rodríguez, Alfonso, Fernández-Medina, Eduardo y Piattini, Mario. Escuela de Ingeniería Civil Informática.Universidad Católica del Maule. *http://www.eici.ucm.cl/.* [En línea] [Citado el: 7 de febrero de 2008.] http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/calidad.produccion/Clases.de.Analisis.desde.modelos.proc.negocios.pdf.

Rojas Véliz, Alfredo. [En línea] [Citado el: 1 de diciembre de 2007.] http://yorkers.es.tripod.com/sitebuildercontent/sitebuilderfiles/analisiscapitulo1.pdf.

Santos, Ernesto. 1980. *Procesamiento de Datos.* Buenos Aires : Macchi, 1980.

Saravia Aramayo, Roger. 2007. Postgrado en Informática.Universidad Mayor de San Andrés. *http://pgi.umsa.bo/.* [En línea] 2007. [Citado el: 8 de diciembre de 2007.] http://pgi.umsa.bo/enlaces/investigacion/pdf/INGSW3_33.pdf?.

SENN, James A. 1992. *Análisis y Diseño de Sistemas de Información.* Segunda Edición. México : McGrawHill, 1992.

Sommerville, Ian. 2005. *"Ingeniería del Software".* Séptima edición. s.l. : Editorial Pearson, 2005.

Una propuesta para la verificación de requisitos basada en métricas. **Bernárdez, B, Durán, A and Toro, M. 2004.** Agosto 2004, Revista de Procesos y Métricas de las Tecnologías de la Información., Vol. 1. 1698-2029.

Universidad EAFIT de Medellín en Colombia. *http://www.eafit.edu.co/. .* [Online] [Cited: enero 23, 2008.]

Vizcaíno, Aurora and Caballero, Ismael. Grupo Alarcos.Universidad de Castilla La Mancha.Escuela Superior de Informática de Ciudad Real. *http://alarcos.inf-cr.uclm.es/.* [Online] [Cited: abril 5, 2008.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_RationalRose.pdf.

Glosario de Términos

Administrador de Reportes

Herramienta administrativa que se utiliza para gestionar los informes vía Web.

Caché

Es una clase de memoria que proporciona acceso rápido a los datos de uso más frecuente.

Cola de Eventos y Notificaciones

Suscripciones que se hacen y deben estar activas, en un orden de creación, esperando que el procesador de entrega y programación las atienda.

Cron y Anacron

Demonios del sistema operativo Linux que se usan para programar tareas automáticas en el sistema.

CUS

Casos de uso del sistema.

DCUS

Diagrama de casos de uso del sistema.

Demonios

Servicios que corren sobre el sistema operativo Linux.

DR

Definición de reporte.

Definición de reporte

Un archivo .dr que contiene información sobre la consulta y el diseño de un reporte.

Extensiones

Plug-ins que requiere el servidor de reportes para su correcto funcionamiento.

Extensión de entrega

El Procesador de entrega y programación utiliza las extensiones de entrega para entregar los reportes

en diversas ubicaciones. El Sistema de Reportes contiene una extensión de entrega por correo electrónico y una extensión de entrega a recursos compartidos de archivos.

Extensiones de Procesamiento de Datos

Componentes o módulos que se utilizan para consultar una fuente de datos.

Extensiones de Representación

Componentes o módulos que convierten los datos y la información de diseño del Procesador de reportes en el formato específico, ejemplo HTML, Excel, XML, imagen, PDF.

Extensiones de Seguridad

Componentes o módulos que se utilizan para autenticar y autorizar usuarios y grupos de usuarios para utilizar el servidor de reportes según sus privilegios.

Fuentes de Datos

Una fuente de datos es una conexión a Base de datos, que se guarda y administra en el proyecto**,** esta contiene los detalles necesarios para comunicarse con la Base de Datos, como el driver o la localización física.

HTML

Lenguaje de Etiquetado de Hipertexto (HyperText Markup Language). Es un lenguaje comúnmente utilizado para la publicación de hipertexto en la Web y desarrollado con la idea de que cualquier persona o tipo de dispositivo pueda acceder a la información en la Web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento.

HTTP

Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol). Es un protocolo utilizado para la transferencia de datos a través de Internet, y que está basado en operaciones sencillas de solicitud y respuesta.

Interfaces Programáticas

Conjunto de funciones y procedimientos que ofrece un sistema o Framework para ser utilizado por otro software como una capa de abstracción.

IR

Ingeniería de Requerimientos.

Metadatos

Datos que describen otros datos en un contexto dado.

Modelación Visual

Es el proceso que permite representar gráficamente el sistema software, permitiendo resaltar los detalles más importantes.

Modelos Semánticos

Un modelo semántico es una descripción de la estructura de la base de datos del negocio.

OTI

Oficina de Tecnología e Informática.

PHP

Es un acrónimo recursivo que significa PHP Hypertext Pre-processor, lenguaje de programación interpretado usado normalmente para la creación de páginas Web dinámicas.

Plug-in

Es un componente informático que interactúa con otra aplicación adicionándosele a esta para aportarle una nueva funcionalidad especifica o conjunto de funcionalidades, este componente es acoplado al la nueva aplicación, la cual incrementara sus utilidades.

Procesador de Entrega y Programación

El Procesador de entrega y programación está incluido para permitir operaciones programadas y controlar las extensiones de entrega utilizadas para insertar reportes en buzones de correo electrónico o en recursos compartidos en la red.

Procesador de Reportes

El procesador de reportes es un componente del servidor de reportes que procesa reportes y modelos de reporte.

Reporte Caché

Es un reporte que contiene los datos capturados en un punto específico de tiempo. Un reporte

caché se almacena en un formato intermedio que contiene datos recuperados en lugar de una consulta y el diseño de información.

Reporte Enlace

Es una "copia" de un reporte con un conjunto de valores de los parámetros u otras propiedades. Un reporte enlace comparte la misma definición DR con el reporte fuente, y, como tal, cuando la DR de este se actualiza, se actualiza la del reporte enlace también.

Reporte Representado

Es un reporte procesado completamente que contiene los datos y el diseño de información, en un formato adecuado para ver, por ejemplo en HTML, Excel, XML, imagen, PDF.

Roles

Los roles son grupos de tareas.

Schedules

Programación de realización automática de actividades o procesos.

Seguridad Basada en Roles

Modelo de seguridad basado al rol que desempeña el usuario, teniendo acceso solo a las opciones del sistema, que el rol le concierne.

Servicio Web

Los Servicios Web son un conjunto de aplicaciones o de tecnologías con capacidad para inter-operar en la Web y que intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario.

SGBD

Sistema Gestor de Bases de Datos

Sistema de Reportes.

Un sistema Web basado en la administración de reportes que combine la capacidad de un generador

de reportes tradicional con la de un sistema de gestión de los mismos.

SMTP

Protocolo Simple de Trasferencia de Correo (Simple Mail Transfer Protocol), se usa para trasmitir correo electrónico entre servidores.

SO

Sistema Operativo

URL

Localizador Uniforme de Recursos (Uniform Resources Locator). Es la secuencia de caracteres con la cual se asigna dirección única a cada uno de los recursos de información disponibles en la Web.

XML

Lenguaje de Etiquetado Extensible (eXtensible Markup Language). Es un lenguaje con una importante función en el proceso de intercambio, estructuración y envío de datos en la Web. Describe los datos de tal manera que es posible estructurarlos utilizando para ello etiquetas, como lo hace HTML, pero que no están predefinidas, delimitando de esta manera los datos, a la vez que favoreciendo la interoperabilidad de los mismos.