Ingeniería de Requisitos del Sistema de Gestión Financiera de los Proyectos del Convenio Cuba-Venezuela.

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

Autora: Mayte Medina López

Tutora: Ing. Daymeri Díaz Rodríguez

Asesor: Ing. Ariel E. Morales Malpica

Ciudad de La Habana

2011

Declaración de Autoría

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

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

_____	_____	_____
Mayte Medina López	Ing. Daymeri Díaz Rodríguez	Ing. Ariel E. Morales Malpica

Dedicatoria

A mis padres, a mi hermana y mi sobrina

A Lila y Yonar

A Raulito

Agradecimientos

Gracias a mi familia, a Raulito, a Lila y Yonar por apoyarme en todo.

Gracias a mis amigos por ayudarme, especialmente los que han estado conmigo en esta última etapa.

Gracias a Ariel, Linnet, Daymeri, Pepe, Yurita, Jose Angel, Yadira por contribuir con la realización de mi tesis. Gracias a mi profesora Maydelin por apoyarme tanto.

Resumen

Uno de los proyectos llevados a cabo por la Universidad de las Ciencias Informáticas es el Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela el cual contempla la realización de un módulo para apoyar el proceso de la gestión del financiamiento de los proyectos en Cuba. Actualmente el proceso de financiamiento se realiza de forma manual. Los datos se introducen en documentos Excel que son transportados, para su consulta, aprobación o validación en memorias USB, lo que provoca redundancia en los datos, demora en el proceso, gasto de recursos materiales y humanos, además de descentralización de la información. Por ello se hace necesario hacer un sistema que permita un mejor control del financiamiento de los proyectos antes mencionados.

En el presente trabajo se plantea el análisis realizado para identificar las características que el sistema debe tener, así como para lograr traducir esas características obtenidas de los clientes, a un lenguaje común para el equipo de desarrollo.

Summary

One of the projects made by the University of Informatics Sciences is the System for the Managing and the Following of the projects established in the Cuba-Venezuela Integral Contract. This project includes the construction of a subsystem that supports the financial process of the projects in the contract, but only by the Cuban part. Nowadays the financial process is done manually. The data are entered in Excel documents, which are transported in flash memory for its revision and approving, causing waste of time, money, material resources and human resources. Besides there is redundancy in data and the information is not centralized. Because of that, it is necessary the creation of a system that allows a better control of the finances of the projects.

In this document is presented the analysis that was made in order to identify the characteristics that the system should have, and also the way to translate the characteristics obtained from the clients to a common language for the development team.

Índice

Introducción

Actualmente la creciente evolución de la ciencia y la tecnología ha tenido un impacto explosivo y transformador de carácter social, cultural, económico y productivo, propiciando un incremento en la producción de software. Cuba, como vislumbró el Comandante en Jefe Fidel, hace de la informática uno de los renglones más productivos y aportadores de recursos, siendo la Universidad de Ciencias Informáticas (UCI) un eslabón fundamental en el desarrollo de software y servicios informáticos.

Uno de los proyectos llevados a cabo por la UCI es el Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela, surgido a raíz del Convenio Integral de Cooperación entre Cuba y Venezuela establecido hace ya más de diez años en el marco del ALBA. Este sistema fue desarrollado para gestionar y darle seguimiento a los proyectos establecidos en el Convenio y contempla la realización de un módulo para apoyar el proceso de la gestión del financiamiento de los proyectos en Cuba.

El proceso de financiamiento es de suma importancia ya que refleja las transferencias monetarias de todos los proyectos, el estado financiero de cada proyecto, tanto su comportamiento planificado como el real. Controla, además, los datos financieros de cada proyecto, la forma de pago, el monto acumulado de ingresos y egresos.

Actualmente el proceso de financiamiento se realiza de forma manual. Los datos se introducen en documentos Excel que son transportados, para su consulta, aprobación o validación en memorias USB debido a la inseguridad de otros medios de envío como por ejemplo el correo electrónico. Esto conlleva a que exista redundancia en los datos, demora en el proceso y gasto de recursos materiales y humanos. Además no existe una centralización de la información lo que trae como consecuencia que se dificulte el control de la misma.

Por esta razón el Ministerio de Economía y Planificación (MEP), el Ministerio de Comercio Exterior (MINCEX), los demás ministerios y sus respectivos entes ejecutores han solicitado la realización de un sistema que les permita un mejor control del financiamiento de sus proyectos. Para ello se hace necesario definir correctamente qué debe realizar el sistema, con el objetivo de lograr la aceptación por parte de los clientes y alcanzar el éxito.

Muchas veces los clientes piensan que ellos saben lo que el software tiene que hacer, sin embargo, se requiere de habilidad y experiencia para reconocer requerimientos incompletos, ambiguos o contradictorios. Se ha demostrado que una gran parte de las aplicaciones no se terminan o simplemente no se usan porque existen incongruencias entre lo que el usuario quiere, lo que realmente necesita, lo que interpreta cada miembro del equipo de proyecto y lo que realmente se obtiene. Por eso es preciso establecer una efectiva comunicación entre los clientes y el equipo de desarrollo, para tener una sólida visión del proceso de financiamiento que se lleva a cabo y lograr entender qué es lo que se debe hacer.

Problema de la investigación: ¿Cómo traducir los requisitos del cliente en un lenguaje comprensible para los desarrolladores que permita una posterior implementación de los procesos de financiamiento en los proyectos del Convenio Cuba-Venezuela?

Objeto de Estudio: Ingeniería de software.

Objetivo General: Aplicar la ingeniería de requisitos a los procesos de financiamiento de los proyectos del Convenio Cuba -Venezuela.

Campo de Acción: Ingeniería de requisitos de los procesos de financiamiento en los proyectos del Convenio Cuba-Venezuela.

Idea a Defender: La aplicación de la ingeniería de requisitos a los procesos de financiamiento en los proyectos del Convenio Cuba - Venezuela posibilita desarrollar un sistema que cumpla con los requisitos del cliente.

Objetivos Específicos

- Realizar el marco teórico de la ingeniería de requisitos aplicada a los sistemas de gestión de financiera.
- Desarrollar los requisitos de los procesos de financiamiento en los proyectos del Convenio Cuba - Venezuela.
- Evaluar los resultados obtenidos de la descripción de los procesos de financiamiento en los proyectos del Convenio Cuba - Venezuela.

Tareas de investigación

- Realización del marco teórico de las herramientas de gestión financiera.
- Realización del marco teórico de las técnicas para el levantamiento de requisitos.
- Realización del marco teórico de las metodologías del desarrollo de software.
- Realización del marco teórico de los patrones de casos de uso.
- Realización del marco teórico de los lenguajes de modelado.
- Realización del modelo del negocio.
- Identificación de los requisitos funcionales y no funcionales.
- Especificación de los requisitos.
- Elaboración de prototipos no funcionales.
- Elaboración de las descripciones de casos de uso del sistema (DCUS).
- Estudio y aplicación de las métricas para medir la calidad de los requisitos.

Métodos Científicos de Investigación a utilizar

Métodos teóricos

Analítico- Sintético: Para analizar las teorías, documentos, con el objetivo de extraer los elementos más importantes que se relacionan con el objeto de estudio, analizar toda la información que se investiga y sintetizarla.

Histórico – lógico: Para constatar teóricamente cómo ha evolucionado un determinado fenómeno en un período de tiempo, ya sea en toda su trayectoria o en un fragmento de la misma. Para el estudio del estado del arte de los sistemas de gestión financiera y del proceso de desarrollo de software.

Modelación: Para la creación de modelos y diagramas que representen el sistema y su comportamiento. Para modelar el negocio.

Hipotético – deductivo: Para la elaboración de la idea a defender, con la que se establece una predicción a partir del sistema de conocimientos que se tiene.

Métodos empíricos

Entrevista: Para la comunicación con los clientes en aras de identificar los requisitos del sistema.

Observación: Para la recogida de la información.

Estructura de la tesis

Capítulo 1 Fundamentación Teórica.

En este capítulo se realiza el estudio de herramientas y lenguajes de modelado, metodologías de desarrollo de software más usadas y se abordan algunos aspectos de la ingeniería de requisitos.

Capítulo 2 Solución Propuesta.

En este capítulo se presenta el modelamiento del negocio, los requerimientos funcionales y no funcionales, la especificación de requisitos, diagrama de casos de uso y descripción de los mismos.

Capítulo 3 Análisis de los Resultados

En este capítulo se muestra el resultado de la aplicación de métricas y listas de chequeo y las tareas realizadas para la verificación y validación de los requisitos.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se presenta la investigación realizada sobre los sistemas de gestión en el mundo y en Cuba con el objetivo de determinar si es necesaria la creación de un nuevo sistema o se puede utilizar algunos de los que ya existen. Se presenta además un estudio sobre la ingeniería de requisitos, las técnicas para el levantamiento de requisitos, los lenguajes y herramientas de modelado, la importancia de utilizar una metodología y ejemplos de algunas de ellas. También se aborda el tema de la medición del software, los modelos de calidad, la validación de los requerimientos y la aplicación de métricas.

Sistemas de gestión

Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización. Un sistema de gestión ayuda a lograr los objetivos de la organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado. (1)

La implementación de un sistema de gestión eficaz puede ayudar a gestionar los riesgos sociales, medioambientales y financieros. Posibilita mejorar la efectividad operativa, reducir costos, aumentar la satisfacción de clientes y partes interesadas, lograr mejoras continuas, potenciar la innovación, eliminar las barreras al comercio.

Sistemas de Gestión en el mundo

Actualmente es común el uso de sistemas para la gestión y existen muchos de ellos como por ejemplo:

ERP (Enterprise Resource Planning)

Un sistema ERP es un conjunto de aplicaciones de gestión integradas entre sí que comparten la misma base de datos. De esta manera, los distintos procesos de negocios de la empresa están relacionados, consiguiendo con ello aumentar el beneficio de la información y mejorar los vínculos. (3)

Los ERP son sistemas integrales, modulares y adaptables. Constituyen una agrupación de todos los módulos que los componen, y que agrupan a su vez todos los procesos de gestión de la empresa.

En estos sistemas suele estar presente una arquitectura modular, donde cada módulo gestiona las funciones de un área empresarial diferente, como por ejemplo: nóminas, finanzas, gestión de proyectos, sistema de gestión geográfica, contabilidad, logística, stock, pedidos. Estas áreas de la empresa realizan funciones diferentes pero se interrelacionan entre sí compartiendo información. Debido a la adaptabilidad de este tipo de sistemas, una empresa puede configurar su ERP para que se adapte a sus procesos de negocio. La personalización de este tipo de sistemas, junto con su modularidad y capacidad de integración de procesos, permite una gestión completa de las operaciones empresariales. Los principales beneficios son: la innovación en la gestión empresarial, ayuda a la planificación de la empresa, aumento de la productividad, reinvención del negocio, posicionando al cliente en el centro, autonomía en la gestión. Además facilita las relaciones comerciales, tanto con clientes como con proveedores. (3) Existen muchas variantes de ERP entre las que se encuentran:

ERP Prowin que mecaniza las diferentes áreas de una empresa y cuenta con un módulos como: comercial, ventas, logística, planificación, producción, calidad y gestión documental. Es totalmente personalizable por el usuario y de sencilla interfaz.

Winconta Financials que es un software ERP de contabilidad diseñado para simplificar todos los procesos contables, financieros y fiscales de una empresa. Cuenta con módulos como: conciliación bancaria, balances, analítica, tesorería, presupuestaria. Tiene facilidades como: la exportación de datos a Excel y la generación de gráficos. También posee gestión documental, configuración de informes y consultas personalizables, copias de seguridad, gestión de usuarios.

Asinom Nóminas y Seguros Sociales es un software para nóminas y recursos humanos (RRHH) con el que se puede ejecutar, de manera sencilla y rápida, los recibos salariales de los trabajadores. Esta herramienta para la gestión de nóminas y seguros sociales es multiempresa y multiconvenio, altamente intuitivo e ideal para pequeñas instalaciones así como para grandes volúmenes de datos.

Quality Pro es un módulo flexible y parametrizable que ofrece una gestión avanzada en el control de calidad de los procesos en la producción de su empresa de forma integral y completa, capaz de gestionar indicadores y objetivos para cada área. Es un programa para el control de calidad permitiendo la reducción de costes y mejorar la eficiencia y calidad de los resultados de las actividades de desarrollo. (3)

Sin embargo el uso de muchos de los sistemas vistos anteriormente representaría una dependencia del fabricante y un gasto en cuanto a la compra del producto, la licencia, consultoría, despliegue, entre otros egresos. En el caso de los ERP se podría mencionar que muchas veces son utilizadas las siglas para vender software de gestión que gestionan procesos muy específicos de las empresas y no cumple todas las funcionalidades de un ERP real, que se debe invertir una suma considerable en hardware debido a las prestaciones que requieren algunos de estos sistemas y también la imposibilidad de adaptación total del sistema a los procesos tradicionales de la empresa que lo adquiere.

En cuanto a los ERP libres se puede decir que algunos usan tecnologías que pueden ser riesgosas en la situación actual de bloqueo que sufre Cuba, por ejemplo Open Bravo y Open Xpertya están basados en la plataforma J2EE cuya máquina virtual es propiedad de SUN que es una empresa norteamericana. Además estos sistemas han sido diseñados para empresas capitalistas que tienen un modelo de gestión y de procesos bastante diferente a las entidades cubanas donde la economía es centralizada y operan otros mecanismos. (4)

Sistemas de Gestión en Cuba

En el país se lleva a cabo el proceso de perfeccionamiento empresarial que tiene como objetivo garantizar la implantación de un sistema de dirección y gestión en las empresas estatales cubanas que logre un significativo cambio organizativo al interior de las mismas y gestionar integralmente los sistemas que la componen. (5) En el marco de este proceso se han realizado algunos sistemas de gestión como:

El **Sistema de gestión de capital humano para entidades de alojamiento turístico en Cuba** diseñado para gestionar eficientemente el área de alojamiento de una entidad turística y para el perfeccionamiento del capital humano. (6)

El **VERSAT-Sarasola**: Sistema cubano de gestión contable-financiero que según la definición de sus creadores es un paquete integrado para la gestión económica financiera que permite enviar información eficaz, de forma inmediata, desde lugares apartados, a la vez que ofrece mayor organización, control y disciplina en cada gestión. Este sistema cuenta con diez módulos: configuración, contabilidad general, costos y procesos, control de inventarios, control de activos fijos, finanzas, caja y banco, facturación, planificación económico / productiva, generador de reportes, nómina de salarios.

Fue el primer sistema cubano que logró certificarse en el país y está estructurado con los elementos requeridos por la contabilidad de este, con lo cual se ha ganado la aceptación en el mercado y ha logrado diseminarse por un gran número de entidades nacionales. (7)

ASSETS NS es un sistema de gestión integral estándar y parametrizado que permite el control de los procesos de compras, ventas, producción, taller, inventario, finanzas, contabilidad, presupuesto, activos fijos, útiles y herramientas y recursos humanos. Como sistema integral todos sus módulos trabajan en estrecha relación, generando automáticamente, al módulo de contabilidad los comprobantes de operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de contabilidad al día. (8)

CEDRUX: Es un sistema integral de gestión, especialmente adaptado al control de los recursos empresariales de las entidades. Es un paquete de soluciones integrales de gestión para las entidades presupuestadas y empresariales. Fue desarrollado siguiendo los principios de independencia tecnológica, que se aplica con todo éxito en seis entidades del país como etapa piloto para su posterior despliegue. (9)

Es un proyecto desarrollado en la UCI con el apoyo de varias entidades que hasta el momento llevaban este tema, como Desoft y TEICO Villa Clara. Estos últimos hicieron una transferencia tecnológica de su propio sistema, Versat Sarasola, al grupo de desarrollo de la UCI. (10)

Los sistemas de gestión creados en Cuba representan un gran avance en el desarrollo de software en nuestro país ya que al evitar la importación de sistemas extranjeros se ahorra dinero destinado al pago de licencias, asesoría, despliegue, eliminando así la dependencia de sus fabricantes. Además se logran sistemas más ajustados a nuestra economía.

Sin embargo ninguno de los sistemas expuestos anteriormente tiene procesos similares a los que se llevan a cabo en los proyectos del Convenio Cuba-Venezuela, ya que muchos de estos procesos son específicos de este tipo de financiamiento. Además el sistema deseado debe integrarse con el sistema CCV y sería engorroso configurar o implementar todo eso en uno de los sistemas anteriores. Por tanto se debe crear un nuevo sistema que sea capaz de cumplir lo antes mencionado.

Ingeniería de Requisitos

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. (11)

La ingeniería de requisitos permite que el equipo de trabajo entienda mejor el problema que se solucionará. Incluye tareas para comprender el impacto del software en el proyecto, para entender que es lo que necesita el cliente y como interactuarán los usuarios finales con el sistema.

El proceso de la ingeniería de requisitos se lleva a cabo a través de siete distintas funciones: inicio, obtención, elaboración, negociación, especificación, validación y gestión. (12)

Muchas de estas tareas se llevan a cabo paralelamente y todas están enfocadas a las exigencias del cliente y deben adaptarse a las necesidades del proyecto. Todas ellas constituyen un puente hacia el diseño y construcción del software. Desde una perspectiva general en la fase de inicio se define el ambiente del problema a resolver. La obtención ayuda al cliente a definir sus necesidades. Durante la elaboración se refinan y modifican los requisitos básicos. Una vez definido el problema por el cliente, se lleva a cabo la negociación, donde se define las prioridades. Finalmente se especifica el problema, después se valida y revisa dicha especificación para verificar que cumple con las expectativas del cliente. A continuación se explican detalladamente cada una de las tareas:

Inicio

Cuando comienza el proyecto es preciso definir el ámbito y la naturaleza del problema que se quiere resolver, se necesita comprender el problema, identificar las personas que pueden contribuir a la obtención de requisitos, tener en cuenta los puntos de vista de cada una de esas personas y los elementos comunes en los mismos, formular algunas preguntas para romper el hielo e iniciar la conversación. Todo ello en aras de propiciar la efectividad de la comunicación entre los clientes y los desarrolladores.

Obtención

Obtener requisitos no es tan sencillo como preguntar al cliente, a los usuarios y a los involucrados que es lo que desean que haga el sistema o producto, como puede éste cumplir las necesidades del negocio y cómo será usado en el día a día. Los siguientes problemas (13) definen con exactitud cuán difícil puede llegar a ser esta tarea:

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

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

Problemas de volatilidad. Los requisitos cambian con el tiempo. Para erradicar estos problemas es necesario que los ingenieros de requisitos realicen la recopilación de requisitos de una forma organizada.

Elaboración

La información recopilada durante las fases anteriores se concreta y se refina en la elaboración. En esta etapa se definen las funciones, características y restricciones del sistema y está dada por tareas de modelado y refinamiento. Mediante la creación y refinamiento de escenarios se muestra como los actores interactuarán con el sistema. De cada escenario se determinan las clases del análisis: entidades del dominio de negocio visibles para el usuario final, se identifican los atributos de las mismas y los servicios que requieren. Se identifican relaciones entre clases y se producen diagramas UML complementarios.

Finamente se obtiene un modelo de análisis que contiene el dominio de la información, las funciones y el comportamiento del problema.

Negociación

Muchas veces los clientes piden más de lo que se puede realizar o proponen requisitos que entran en conflicto entre sí. El ingeniero de requisitos debe solucionar estos problemas mediante la negociación de un plan de proyecto realista. El equipo de software, el cliente y otros interesados en el proyecto negocian la prioridad, disponibilidad y costo relativo de cada requisito.

Las mejores negociaciones son aquellas que buscan un resultado del tipo “ganar-ganar”. El cliente gana al obtener el sistema o producto que satisface sus necesidades y el equipo gana al trabajar con presupuestos y límites realistas y alcanzables.

Especificación

Una especificación puede ser un documento escrito, un modelo gráfico, un modelo matemático formal, una colección de escenarios de uso, un prototipo o una combinación de cualquiera de estos (12). La especificación es el producto final de la ingeniería de requisitos y sirve como base para actividades posteriores de la ingeniería de software. En la especificación se describe la funcionalidad del sistema basado en computadora y las restricciones del mismo.

Validación

La validación de requerimientos trata de mostrar que éstos realmente definen el sistema que el cliente desea (14). La validación de requisitos examina la especificación para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, que fueron detectadas inconsistencias, omisiones, y que los errores encontrados fueron corregidos, y que el resultado del trabajo cumple con los estándares establecidos para el proceso, el proyecto y el producto.

Las técnicas de validación de requerimientos pueden ser empleadas tanto individualmente como en conjunto. Algunas de ellas son:

Revisión de requerimientos: Para la revisión técnica formal debe existir un equipo de revisión que incluya ingenieros del sistema, clientes, usuarios, y otros interesados que examinan la especificación. Estos buscan errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información

incompleta, inconsistencias, requisitos contradictorios, o requisitos irreales o inalcanzables. La revisión es esencial para asegurarse que el cliente y el desarrollador tienen el mismo concepto del sistema.

Construcción de prototipos: Mediante la construcción de prototipos el cliente puede comprobar si el sistema cumple sus necesidades. El prototipo puede ser funcional o no.

Generación de casos de prueba: Los requerimientos deben poder probarse. Si las pruebas para éstos se conciben como parte de la validación, pueden revelar problemas en los requisitos. Si una prueba es difícil o imposible de diseñar, los requisitos pueden ser difíciles de implementar y deben ser considerados nuevamente.

Gestión de requisitos

Los requisitos están sujetos a cambios a lo largo del desarrollo del sistema. La gestión de requisitos es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento. (11)

La gestión de requisitos comienza con la obtención de requisitos. A cada requisito se le asigna un identificador y se llevan a cabo las tablas de rastreabilidad (12) donde se relacionan los requisitos con determinados aspectos del sistema o de su ambiente. Las tablas de rastreabilidad permiten entender el impacto del cambio del requisito en diferentes aspectos del sistema. Algunas de ellas son:

Tabla de rastreabilidad de la las características. Muestra la manera en que los requisitos se relacionan con las características del sistema observables para el cliente.

Tabla de rastreabilidad de la fuente. Identifica la fuente de cada registro.

Tabla de rastreabilidad de dependencia. Indica la forma en que los requisitos se relacionan entre sí.

Tabla de rastreabilidad del subsistema. Establece categorías entre los requisitos de acuerdo a los subsistemas que gobiernan.

Tabla de rastreabilidad de la interfaz. Muestra la forma en que los requisitos se relacionan con las interfaces internas y externas del sistema.

Técnicas para la Captura de Requisitos

Las técnicas más habituales en la elicitación de requisitos son las entrevistas, el Joint Application Development (JAD) o Desarrollo Conjunto de Aplicaciones, el brainstorming o tormenta de ideas y la utilización de escenarios, más conocidos como casos de uso. (15)

Entrevistas

Las entrevistas son la técnica de elicitación más utilizada, y una de las formas de comunicación más naturales entre las personas. Las entrevistas necesitan ser elaboradas, se debe estudiar el dominio del problema, seleccionar las personas a las que se va a entrevistar, determinar el objetivo y contenido de las entrevistas. Durante la realización se informa al entrevistado del objetivo de la entrevista y detalles concernientes a la misma y se emplean técnicas como: las preguntas abiertas, utilizar palabras apropiadas evitando tecnicismos, mostrar interés en todo momento y evitar los monólogos y para terminar se debe recapitular para evitar confusiones y dejar abierta la posibilidad de una nueva entrevista. Finalmente se analiza la entrevista revisando las notas tomadas o las grabaciones, reorganizando la información y contrastándola con otras fuentes.

JAD (Joint Application Development)

La técnica JAD (Desarrollo Conjunto de Aplicaciones), desarrollada por IBM en 1977, es una alternativa a las entrevistas individuales que se desarrolla a lo largo de un conjunto de reuniones en grupo durante un periodo de 2 a 4 días. (15) En estas reuniones se ayuda a los clientes y usuarios a formular problemas y explorar posibles soluciones, involucrándolos y haciéndolos sentirse partícipes del desarrollo. Ésta técnica se base en cuatro principios: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (What You See Is What You Get, lo que se ve es lo que se obtiene), por la que durante las reuniones se trabaja directamente sobre los documentos a generar.

Tormenta de Ideas

El brainstorming o tormenta de ideas es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. (15) Las sesiones de brainstorming suelen estar

formadas por un número de cuatro a diez participantes, uno de los cuales es el jefe de la sesión, encargado más de comenzar la sesión que de controlarla. Como técnica de elicitación de requisitos, el brainstorming puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de elicitación, cuando los requisitos son todavía muy difusos.

Casos de Uso

Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con énfasis en el valor añadido para cada usuario individual o para cada sistema externo. (16) Mediante la utilización de los casos de uso, los analistas se ven obligados a pensar en términos de quiénes son los usuarios y qué necesidades u objetivos de la empresa pueden cumplir. Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia.

Patrones de casos de uso

Los patrones permiten reusar la esencia de la solución de un problema al enfrentar nuevos problemas similares. Según Christopher Alexander, originador de la idea de los patrones, “Cada patrón describe un problema que ocurre una y otra vez, y describe la solución a ese problema, de tal manera que dicha solución pueda ser usada un millón de veces más, sin hacerlo necesariamente dos veces del mismo modo”.

Existen diversos tipos de patrones, entre ellos se encuentran los de arquitectura, de diseño (creacionales, estructurales y de comportamiento) y los de casos de uso. Un patrón de caso de uso es un diseño probado en un modelo de casos de uso, junto con una descripción del contexto en el cual será usado y las consecuencias que tendrá su aplicación en el modelo (17). A continuación se expone una descripción de algunos de ellos:

Reglas de negocio: Se basan en la extracción de información originada de las políticas, reglas y regulaciones del negocio de la descripción del flujo y describe la información como una colección de reglas del negocio referenciadas a partir de las descripciones de los casos de uso.

Concordancia: Reutilización (Commonality: Reuse): Es un patrón de estructura que consiste en tres casos de uso. El primero llamado “Sub-secuencia Común”, modela la secuencia de acciones que aparecen en múltiples casos de uso del modelo. Los otros dos casos de uso comparten de esta sub-secuencia común de acciones (dos es la menor cantidad que puede existir). La sub-secuencia tiene que estar en un fragmento, es decir, todo lo que requiere estar incluido tiene que estar en un único fragmento completo. Además no se puede hacer referencia desde la sub-secuencia a donde esta es utilizada, porque el caso de uso incluido tiene que ser independiente del caso de uso base.

Concordancia: Especialización (Commonality: Specialization): Otro patrón de concordancia que contiene casos de uso del mismo tipo. En este caso, ellos son modelados como especializaciones de un caso de uso de uso común. Todas las acciones del caso de uso común son heredadas por sus casos de uso hijos, en donde otras acciones pueden ser agregadas o las acciones heredadas pueden ser especializadas. Este patrón es aplicable cuando las acciones modeladas por los casos de uso son del mismo tipo, y este tipo debería ser hecho visible en el modelo.

Extensión o Inclusión Concreta: Extensión (Concrete Extension or Inclusion: Extension): Consiste en dos casos de uso y una relación de extensión entre ellos. El caso de uso extendido es concreto; es decir, puede ser instanciado por el mismo, así como también puede extender del caso de uso base. El patrón es aplicable cuando un flujo puede extender el flujo de otro caso de uso, así como también puede ser realizado por el mismo.

Extensión o Inclusión Concreta: Inclusión (Concrete Extension or Inclusion: Inclusion): Existe una relación de inclusión desde el caso de uso base al caso de uso incluido. El último puede ser instanciado por el mismo. El caso de uso base podría ser cualquiera, concreto o abstracto. El patrón es usado cuando un flujo podría ser incluido en el flujo de otro caso de uso y además puede ser realizado por el mismo.

CRUD Completo: Consiste en un caso de uso, llamado Gestionar, modelando las diferentes operaciones que pueden ser realizadas en una pieza de información de cierto tipo, como crearla, leerla, actualizarla y eliminarla. Este patrón debería ser usado cuando todos los flujos contribuyen al mismo valor de negocio y son cortos y sencillos.

Actores Múltiples: Roles distintos (Multiple Actors: Distinct Roles): Consiste en un caso de uso y al menos dos actores. Es usado cuando dichos actores juegan diferentes roles en relación con el caso de uso, es decir, ellos interactúan diferentemente con el caso de uso.

Actores Múltiples: Roles comunes (Multiple Actors: Common Roles): Es un patrón alternativo, los dos actores juegan el mismo rol en relación con el caso de uso. Este rol es representado por otro actor, del cual heredan los dos actores anteriores que comparten el mismo rol. El patrón es aplicable cuando, desde el punto de vista del caso de uso, solo existe una entidad externa interactuando con cada instancia del caso de uso.

Lenguajes de modelado

UML (Lenguaje Unificado de Modelado)

UML es un lenguaje para construir modelos, no guía al desarrollador en la forma de realizar el análisis y diseño orientados a objetos ni le indica cuál proceso de desarrollo adoptar (Larman, 1999). Comenzó como una iniciativa de Grady Booch y Jim Rumbaugh en 1994 para combinar las notaciones visuales de sus conceptos (Booch y OMT). En 1995 Ivar Jacobson, creador del OOSE se unió al proyecto. Juntos propusieron en 1997 la versión 1.0 de UML como metamodelo orientado a objetos de semántica y notación estándares. (19)

En Julio de 2003 salió a la luz UML 2.0. Los cambios más obvios del UML 1.x al 2.0 fueron la introducción de nuevos diagramas que incluyen:

- Diagrama de Estructura.
- Diagrama Compuesta.
- Diagrama de Comunicación.
- Diagrama de Oportunidad.
- Diagrama de Interacción por Repaso.

UML permite la reutilización de código, el descubrimiento de fallas, ahorro de tiempo en el desarrollo del software. Además divide cada proyecto en un número de diagramas que representan las distintas vistas del proyecto y juntos representan la arquitectura del mismo y permite describir un sistema en diferentes niveles de abstracción (19). UML es el lenguaje de modelado por excelencia para la representación del software a escala global. Es utilizado desde pequeñas aplicaciones académicas hasta grandes aplicaciones de gestión de la industria.

BPMN (Notación para la Gestión de Procesos de Negocio)

Un proceso de negocio es un conjunto de actividades relacionadas dentro de una organización que tienen como objetivo conseguir un determinado resultado. La gestión de estos procesos, BPM (Business Process Management), comprende la definición de los procesos, normalmente mediante una notación formal, y la creación del correspondiente modelo.

BPMN es una notación para el modelado de Procesos de Negocio que fue desarrollada inicialmente por BPMI (Business Process Management Initiative) y se fusionó con OMG (Object Management Group) en Junio de 2005. Permite la visualización del lenguaje BPEL4WS (Business Process Execution Language for Web Services) con una notación orientada a negocios. Define la notación y semántica de un diagrama de procesos de negocio (Business Process Diagram, BPD) (20). BPMN, según sus propios autores, se inspira en la experiencia de varios estándares como: diagramas de actividad de UML, UML EDOC, IDEF, ebXML BPSS, Rosseta Net, entre otros. Tiene como principales objetivos:

- Proveer una notación que sea fácilmente entendida por todos los usuarios, desde el analista de negocio, el desarrollador técnico y hasta la propia gente del negocio.
- Crear un puente estandarizado para el vacío existente entre el diseño del proceso de negocio y su implementación.
- Asegurar que los lenguajes para la ejecución de los procesos de negocio puedan ser visualizados con una notación común.

Los modelos BPMN constan de tres secciones (o submodelos) básicos (40). Ellos son:

Procesos de negocio privados (internos). Son los diagramas de flujo de trabajo o diagramas de workflow.

Procesos abstractos (públicos). Representan las interacciones existentes entre un proceso de negocio privado y, o bien otro proceso de negocio o bien un participante del proceso.

Procesos de colaboración (globales). Muestra la interacción entre distintas entidades de negocio.

BPMN abarca únicamente los procesos de negocio, lo que significa que otro tipo de modelos relacionados (estructura de la organización, recursos, modelos de datos, estrategias, reglas de negocio, etc.) quedan fuera de la especificación. BPMN es usado para comunicar una amplia variedad de información a una amplia variedad de audiencias y proporciona a los negocios la capacidad de entender sus procedimientos internos en una notación gráfica, facilitando a las organizaciones la habilidad para comunicar esos procedimientos de una manera estándar.

IDEF

IDEF (ICAM Definition Languages, siendo ICAM Integrated-Aided Manufactoring) ideado por la Fuerza Aérea de los Estados Unidos (40), tiene como objetivo: modelar, gestionar y mejorar procesos de negocio. Entre los distintos métodos que ha logrado producir cabe destacar:

- IDEF0 para el modelado de procesos dentro de una organización.
- IDEF1 para el modelado de información.
- IDEF1X para el modelado de datos.
- IDEF2 para el diseño de modelos de simulación.
- IDEF3 para la captura de descripciones de procesos.
- IDEF4 para el diseño orientado a objetos.
- IDEF5 para describir ontologías para la captura de descripciones.

En lo referente a notaciones y lenguajes de procesos los más importantes son:

IDEF0: permite modelar actividades y no depende del tipo de organización y del tiempo, por lo que hay que tener en cuenta que desde ese punto de vista no es ni un organigrama ni un diagrama de flujo. Se

aplica principalmente a: comunicar reglas y procesos de negocio, obtener una visión estratégica de un proceso y facilitar un análisis para identificar puntos de mejora.

IDEF3: sirve como herramienta para analizar procesos existentes y para diseñar y probar nuevos procesos antes de iniciar cambios reales que pueden ser muy costosos. Se aplica principalmente a: documentar un proceso actual (a nivel de detalle), identificar y capturar conocimiento crítico sobre un proceso, facilitar un análisis de un proceso en particular, proponer alternativas a un proceso, planear cambios en un proceso.

Selección del lenguaje

Para modelar el sistema se selecciona UML debido a que es un lenguaje ampliamente usado, fue desarrollado a la par de RUP, es multiplataforma y permite realizar diagramas a todo lo largo del ciclo de vida del software. Sin embargo, para modelar el negocio se selecciona BPMN que se basa en el diagrama de actividad de UML, que cubre casi totalmente los patrones de workflow lo que propicia una gran expresividad a la hora de especificar procesos, es más expresivo. Además, es gráficamente más rico que UML, con menos símbolos fundamentales, pero con más variaciones de estos, lo que facilita su comprensión por parte de personal no experto. Además BPMN está a la vanguardia de modelado de negocio y ha sido adoptada por la OMG como una norma oficial.

Herramientas de Modelado

Se puede definir a las herramientas CASE (Computer Aided Software Engineering: Ingeniería de Software Asistida por Computadora) 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 (21). El objetivo de las herramientas CASE es apoyar y hacer más fácil el desarrollo de software. Su utilización trae consigo favorables ventajas como el incremento en la velocidad de desarrollo de los sistemas, la mejora de la calidad de los desarrollos realizados y el aumento de la productividad.

Las herramientas CASE, en función de las fases del ciclo de vida abarcadas (21), se pueden agrupar de la forma siguiente:

1. Herramientas integradas, I-CASE (Integrated CASE, CASE integrado): abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.
2. Herramientas de alto nivel, U-CASE (Upper CASE – CASE superior) o front-end, orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.
3. Herramientas de bajo nivel, L-CASE (Lower CASE – CASE inferior) o back-end, dirigidas a las últimas fases del desarrollo: construcción e implantación.
4. Juegos de herramientas o Tools-Case, son el tipo más simple de herramientas CASE. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

Las herramientas CASE posibilitan a las compañías que las usan, una competencia efectiva en un mercado altamente competitivo. Lo cual puede hacer la diferencia entre el éxito y el fracaso. También proveen a los analistas de más tiempo para el análisis y diseño y contribuyen a minimizar el tiempo para codificar y probar. Algunas de estas herramientas son: Axure, Rational Rose, Enterprise Architect y Visual Paradigm for UML.

Axure RP (Rapid Prototyping)

Axure RP Pro 5.6 es una aplicación ideal para crear prototipos y especificaciones muy precisas para páginas web. Se trata de una herramienta especializada en la tarea que cuenta con todo lo que se puede necesitar para crear los prototipos de forma más eficiente.

Axure RP Pro 5.6 permite componer la página web visualmente, añadiendo, quitando y modificando los elementos con suma facilidad. Donde esta herramienta demuestra su grado de especialización es en las anotaciones. En este punto, permite especificar el estado de cada elemento (propuesto, aceptado, incorporado), el beneficio esperado (crítico, importante, útil), el riesgo, la estabilidad, a quién va dirigido y a quién se le asignará la tarea (22). Axure RP Pro 5.6 cuenta entre sus características el troceo de imágenes, la corrección de errores, y la no limitación de variables. (23)

Rational Rose

Rational Rose Enterprise es una herramienta CASE desarrollada por Rational Corporation basada en el Lenguaje Unificado de Modelación (UML), que permite crear los diagramas que se van generando durante el proceso de Ingeniería en el Desarrollo del Software. Además de UML, admite como notaciones OMT y Booch. Permite el desarrollo multiusuario y es una herramienta disponible en múltiples plataformas. (24)

Rational posibilita el modelado en UML para diseñar bases de datos, que integra los requisitos de datos y aplicaciones mediante diseños lógicos y analíticos. También posee complementos de modelado Web que incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones Web. Además, presenta funciones de análisis de calidad de código.

Características adicionales incluidas:

1. Soporte a modelos de análisis, ANSI C++, Rose J y Visual C++ según el documento "Design Patterns: Elements of Reusable Object – Oriented Software".
2. Los componentes del modelo se pueden controlar independientemente, lo que permite una gestión y un uso de modelos más granular.
3. Soporte para compilación y descompilación de las construcciones más habituales de Java 1.5.
4. Generación de código en lenguaje Ada, ANSI C++, C++, CORBA, Java y Visual Basic, con funciones configurables de sincronización entre los modelos y el código.
5. Soporte para Enterprise Java Beans 2.0.
6. Creación de definiciones de tipo de documentos DTD en XML.
7. Integración con otras herramientas de desarrollo de IBM Racional.
8. Integración con cualquier sistema de control de versiones compatibles con SSC, como IBM Rational Clear Case. (25)

Enterprise Architect

Enterprise Architect es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Es una herramienta multiusuario (aunque solamente para ediciones profesional

y corporativa), está basada en Windows y presenta documentación de alta calidad compatible con MS Word. Además es una herramienta con bajo costo de licencia.

Características Principales

1. Casos de Uso, Modelos Lógico, Dinámico y Físico.
2. Extensiones personalizadas para modelado de procesos y más.
3. Modelado de Datos, Ingeniería directa de Base de Datos a DDL e ingeniería inversa de Base de Datos desde ODBC.
4. Ingeniería de Código Directa e Inversa (ediciones corporativa y profesional solamente)
5. Soporte para Action Script 2.0, Java, C#, C++, VB.Net, Delphi, Visual Basic, Python y PHP.
6. Facilidad de Importación/Exportación XMI.
7. Corrector Ortográfico (26)

Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (27). Permite la realización de ingeniería inversa para: JAVA, .NET, XML, Esquemas XML, Hibernate, C++. Por ejemplo: código a modelo, código a diagrama. Esto es posible también para las bases de datos: desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

Lista de características

- Modelado colaborativo con CVS (Concurrent Versions System) y Subversion
- Diagramas EJB (Enterprise Java Beans) - Visualización de sistemas EJB.
- Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
- Generador de informes para generación de documentación

- Importación y exportación de ficheros XMI (XML Metadata Interchange).
- Editor de figuras
- Exportación de imágenes jpg, png y svg(w3g estándar). (27)

Se integra con diversos IDE's como son: NetBeans (de Sun), JDeveloper (de Oracle), Eclipse (de IBM), JBuilder (de Borland) se intregra, además, con Visio para el dibujo de diagramas UML con plantillas (stencils) de MS Visio. Cuenta con la edición Community que es libre, aunque está limitada en algunas cosas como la ingeniería inversa y la compatibilidad con diversas IDE`s. (28) Esta herramienta en particular ahorra tiempo y trabajo al equipo de desarrollo ya que posibilita la generación de código, tanto de modelo a código como de diagrama a código. Además En la generación de bases de datos permite la transformación de diagramas de Entidad-Relación en tablas de base de datos. También tiene soporte ORM: se pueden generar objetos Java desde la base de datos.

Selección de la herramienta

La herramienta seleccionada para modelar los artefactos es el Visual Paradigm, que a diferencia del Axure que tiene la desventaja de ser solamente para la creación de prototipos, éste soporta el ciclo de vida completo del software. En comparación con Rational Rose, Enterprise Architect y Axure que son de propietarios y no baratos, Visual Paradigm es libre en su edición Community. Además es una herramienta excepcional que presenta unas características gráficas muy cómodas que hacen más fácil la realización de los diagramas de modelado que sigue el estándar de UML que son: diagramas de clase, casos de uso, comunicación, secuencia, estado, actividad, componentes, y también diagramas de procesos de negocio y diagramas de flujo de datos. Permite la reorganización de las figuras y conectores de los diagramas UML y contiene un editor de detalles de casos de uso, para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso. El motivo de la selección es principalmente que nuestra universidad tiene licencia para la versión VP 6.4, además esta herramienta es de dominio del equipo de trabajo, que ha trabajado con anterioridad en la misma, lo que tributa a un desarrollo más rápido al no tener que emplear tiempo en capacitación.

Metodologías

Una metodología se define como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Para proyectos pequeños, en lugar de usar una metodología, se separa rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función se determina un tiempo aproximado de desarrollo. Para proyectos mayores sí tiene sentido basarse en una metodología de desarrollo. Muchas veces no se encuentra la más adecuada y se diseña una metodología propia, lo que no está mal, siempre y cuando cumpla con el objetivo (29). Algunas de las características deseadas en una metodología son:

- Existencia de reglas predefinidas
- Cobertura total del ciclo de desarrollo
- Verificaciones intermedias
- Planificación y control
- Comunicación efectiva
- Herramientas CASE
- Actividades que mejoren el proceso de desarrollo

Actualmente las metodologías se clasifican en dos grandes grupos: las tradicionales o pesadas y las ligeras o ágiles. Las primeras se basan en el orden y la documentación, hacen mayor énfasis en la planificación y control del proyecto, en la especificación precisa de requisitos y modelado. Las segundas se basan en una comunicación directa con los involucrados en el proceso, son menos orientadas al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada, son más bien orientados al código. (30) (31) (32)

Entre las metodologías tradicionales o pesadas se encuentran: RUP (Rational Unified Process, Proceso Unificado de Rational), MSF (Microsoft Solution Framework, Marco de trabajo de Solución de Microsoft). Entre las metodologías ágiles se tienen: XP (Extreme Programming, Programación Extrema), FDD (Feature Driven Development, Desarrollo Manejado por Rasgos). A continuación se abordan aspectos generales de las mismas.

Microsoft Solution Framework (MSF)

MSF es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas (29). Consta de cinco fases distintas (32): previsión, planeamiento, desarrollo, estabilización, implementación.

Fig. 1 Fases de MSF

MSF está compuesto por varios modelos que planifican las diferentes partes implicadas en el desarrollo de un proyecto: modelo de arquitectura del proyecto, modelo de equipo, modelo de proceso, modelo de gestión del riesgo, modelo de diseño de proceso y finalmente el modelo de aplicación. Es una metodología:

- Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un lugar específico.
- Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
- De tecnología agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. (29)

El modelo de proceso MSF combina los mejores principios del modelo en cascada y del modelo en espiral. Combina la claridad que planea el modelo en cascada y las ventajas de los puntos de transición del modelo en espiral.

FeatureDrivenDevelopment(FDD)

El Desarrollo Manejado por Rasgos (FDD por su sigla en inglés de Feature Driven Development) fue desarrollado por Peter Coad, Erich Lefebvre y Jeff De Luca. Como las otras metodologías adaptables, se enfoca en iteraciones cortas (2 semanas en FDD) que entregan funcionalidad tangible, que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar. Las iteraciones se deciden en base a features (de ahí el nombre del proceso) o funcionalidades, que son pequeñas partes del software con significado para el cliente. Un proyecto que sigue FDD se divide en 5 fases: desarrollo de un modelo general, construcción de la lista de funcionalidades, plan de releases en base a las funcionalidades a implementar, diseñar en base a las funcionalidades e implementar en base a las funcionalidades.

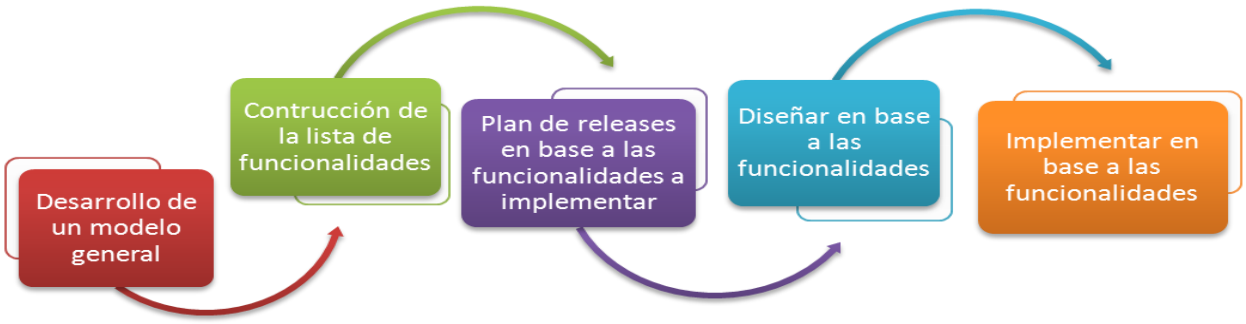


Fig. 3 Fases FDD

El trabajo se realiza en grupo, teniendo un responsable último con mayor experiencia (arquitecto jefe o jefe de programadores dependiendo de la fase), que tendrá la última palabra en caso de no llegar a un acuerdo. Al trabajar en grupo se consigue que todos formen parte del proyecto y que los menos inexpertos aprendan de las discusiones de los más experimentados, y al tener un responsable último, se asignan las responsabilidades que todas las empresas exigen. (30)

Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Entre sus aspectos fundamentales presenta la comunicación, entre los usuarios y los desarrolladores, la simplicidad, al desarrollar y codificar los módulos del sistema, la retroalimentación del equipo de desarrollo, el cliente y los usuarios finales, la programación en pares que consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Se basa en pruebas unitarias que se realizan a los principales procesos, y en la refabricación o reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio. El manejo del cambio se convierte en parte fundamental del proceso, mientras que el costo del cambio no depende de la fase o etapa.

XP cuenta con 6 fases:

1. Exploración: Los clientes plantean las historias de usuario que son de interés para la primera entrega del producto. El equipo de desarrollo se capacita en las herramientas, tecnologías y prácticas a usar en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Esta fase consta de poco tiempo, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
2. Planificación de la Entrega: El cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.
3. Iteraciones: Incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega contiene iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto, escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración. Al final de la última iteración el sistema estará listo para entrar en producción.

4. Producción: Requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
5. Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para ello se requiere de tareas de soporte para el cliente. Así, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. Esta fase puede requerir nuevo personal dentro del equipo y cambios en su estructura.
6. Muerte del Proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo. (34)

Fig. 4 Fases de XP

XP no introduce funcionalidades antes que sean necesarias, (29) solo se programa la funcionalidad que es requerida para la release actual. Se sigue un diseño evolutivo con la siguiente premisa: conseguir la funcionalidad deseada de la forma más sencilla posible. Este diseño evolutivo hace que no se le dé apenas importancia al análisis como fase independiente, puesto que se trabaja exclusivamente en función de las necesidades del momento. (30)

Rational Unified Process (RUP)

RUP es un marco de trabajo genérico que puede adaptarse a diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños del proyecto. Está basado en componentes, o sea que el software está basado en componentes software interconectados a interfaces bien definidas (16). Además utiliza UML y presenta tres características fundamentales (33) que lo identifican:

- Iterativo e incremental: Cada fase se desarrolla en iteraciones. Se divide el trabajo en partes más pequeñas o mini-proyectos, donde cada uno es una iteración que resulta en un incremento.
- Centrado en la arquitectura: La arquitectura describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Guiado por los casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

Un proyecto realizado siguiendo RUP se divide en cuatro fases:

1. Inicio (determinar la visión del proyecto)

2. Elaboración (determinar la arquitectura óptima)

3. Construcción (obtener la capacidad operacional inicial)

4. Transición (obtener el release del proyecto)

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo de cascada o *waterfall* para los flujos de trabajo (30). El ciclo de vida que se desarrolla por cada iteración se realiza bajo dos disciplinas (33) que agrupan los flujos de trabajo:

Disciplina de Ingeniería

- Modelamiento del negocio: describe los procesos del negocio, identificando quiénes participan y las actividades que requieren automatización.

- Requerimientos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Pruebas: busca los defectos a lo largo del ciclo de vida del producto de software.
- Despliegue: produce liberaciones del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

Disciplina de Apoyo

- Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: control de versiones, utilización y actualización concurrente de elementos y otros.
- Administración del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Ambiente: contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

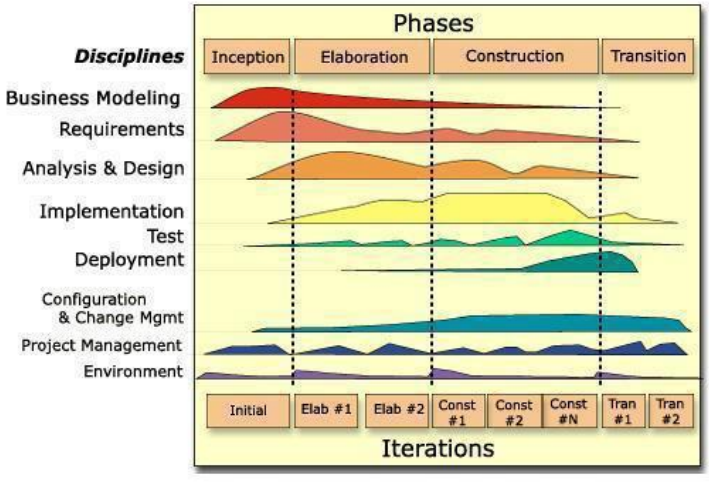


Fig. 2 Fases y Flujos de trabajo de RUP.

RUP es uno de los procesos más generales de los existentes actualmente, ya que está pensado para adaptarse a cualquier proyecto, no solamente de software. Es un proceso muy general y muy grande, pero puede ser adaptado al proyecto que se desee. Además tiene una sólida estructura y la ventaja de que en cada ciclo de iteración se hace imprescindible el uso de artefactos, lo que propicia que RUP sea una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Selección de la metodología

La metodología seleccionada para desarrollar el software es RUP, que a diferencia de XP donde el desarrollo de software es riesgoso y difícil de controlar, ésta contiene gestión de riesgo, control de la calidad y realiza un levantamiento exhaustivo de requerimientos. Además al tener un desarrollo iterativo e incremental, elimina muchos de los errores que otras metodologías dejan en el tiempo. Por ejemplo XP hace que no se le dé apenas importancia al análisis como fase independiente, puesto que se trabaja exclusivamente en función de las necesidades del momento, mientras que RUP hace énfasis en la especificación precisa de requisitos y modelado, realiza el análisis y diseño tan completo como sea posible. Además en las distintas fases se generan artefactos que van siendo desarrollados y completados en cada iteración. Uno de los principales motivos para seleccionar RUP es que el equipo de trabajo tiene experiencia en el desarrollo de software usando esta metodología y que utiliza UML que es el lenguaje de modelado seleccionado.

Validación de requisitos

La validación de requisitos se realiza con el objetivo de probar que los mismos definen lo que el cliente solicita y es de suma importancia ya que el coste de arreglar un error en los requerimientos es mayor al de solucionar problemas en el diseño y la implementación del sistema. Para realizar esta tarea se llevan a cabo verificaciones sobre el documento de especificación de requisitos (14), entre ellas:

1- *Verificaciones de validez:* Se debe llegar a un entendimiento común con los clientes de lo que debe hacer el sistema. Puede que los usuarios piensen que el sistema debe realizar ciertas funciones y en el análisis se identifiquen funciones adicionales o diferentes.

2- *Verificaciones de consistencia:* Los requerimientos en el documento no deben contradecirse. No debe haber restricciones o descripciones contradictorias de la misma funcionalidad del sistema.

3- *Verificaciones de completitud:* En el documento debe tener requisitos que abarquen todas las funciones y restricciones propuestas por el cliente.

4- *Verificaciones de realismo:* Los requisitos se deben verificar teniendo en cuenta la tecnología existente para comprobar que pueden implementarse.

5- *Verificabilidad:* Los requisitos deben redactarse de manera que sean verificables.

Mediante la técnica de validación revisión de requerimientos (36), abordada con anterioridad en este capítulo, el equipo de revisores puede comprobar la:

1- *Verificabilidad.* ¿Puede probarse el requerimiento de modo realista?

2- *Compresibilidad.* ¿Las personas que adquieren el sistema o los usuarios finales comprenden correctamente el requisito?

3- *Rastreabilidad.* ¿Está claramente establecido el origen del requerimiento? La rastreabilidad es importante porque permite evaluar el impacto del cambio en el resto del sistema.

4- *Adaptabilidad.* ¿Es adaptable el requerimiento? ¿Puede cambiarse el requerimiento sin causar efectos de gran escala en los otros requerimientos del sistema?

Modelos de calidad

Un modelo de calidad es la descripción, en términos de un conjunto estructurado de características y sub-características, de los atributos que debe tener un producto software y que contribuyen a que dicho producto sea bueno dentro de los de su clase (ISO/IEC 1991). Los modelos de calidad para ingeniería de requisitos se basan en características de calidad de los requisitos o en taxonomías de defectos. Los primeros aportan una lista de características de calidad de los requisitos, y los segundos, en contraposición, proporcionan una taxonomía de defectos, entendiendo que un defecto en los requisitos es la ausencia de alguna de las características de calidad.

Una de las principales clasificaciones de características de calidad en los requisitos es la propuesta por Alan M. Davis en 1993 que consta de veinticuatro propiedades deseables de los requisitos así como

ciertas medidas para estimar hasta qué punto los requisitos de una especificación cumplen esas propiedades. Algunas de ellas son:

No ambigüedad: tiene una única interpretación posible

Completitud: la especificación de requisitos contempla todo lo que se supone que debe hacer el software, todas las respuestas a posibles entradas y situaciones, todas las páginas están numeradas y todos los elementos identificados, no hay secciones por determinar

Corrección: todo requisito contribuye a la satisfacción de una necesidad y todas las necesidades se hallan recogidas en la especificación

Comprensibilidad: todo lector puede entenderlo con un mínimo de explicación

Verificabilidad: existen técnicas finitas y eficientes (de coste razonable) para verificar si el sistema satisface el requisito, un requisito no es verificable si es ambiguo, es un problema no resuelto o no se puede valorar el coste de probarlo

Alcanzabilidad: existe al menos un diseño e implementación del sistema que implemente correctamente todos los requisitos

Concisión: es tan corto como sea posible sin disminuir la calidad de la especificación

Independencia del diseño: describe de forma pura comportamientos externos, por lo que existe más de un diseño e implementación posible

Modificabilidad: su estructura y estilo es tal que cualquier cambio se puede hacer sin dificultad, completa y consistentemente

No redundacia: cada requisito aparece una sola vez

Al nivel correcto de detalle: tan específico como para que cualquier sistema que satisfaga los requisitos, satisfaga las necesidades del cliente, pero lo suficientemente abstracto como para que cualquier sistema que satisfaga las necesidades del cliente satisfaga los requisitos

Reutilizabilidad: las frases, párrafos y secciones pueden ser fácilmente adoptadas o adaptadas para futuras especificaciones

Trazabilidad: está claro el origen del requisito, generalmente documentos anteriores del proyecto como requisitos de nivel superior, informes de investigación, etc.

Medición del software

La medición es fundamental para ayudar a evaluar la calidad de los resultados obtenidos. Si no se mide solo se podrá juzgar en base a una evaluación subjetiva. Por ello se hace necesaria la obtención de medidas indirectas que den lugar a métricas, las cuales a su vez proporcionen una indicación de la calidad del producto.

Resulta difícil imaginar cómo medir un software, cómo medir algo que no es tangible. Sin embargo existen métricas de software que proporcionan una manera cuantitativa de valorar la calidad de los atributos internos del producto, lo que permite valorar la calidad aún antes de construir el producto.

Existen varias métricas para medir un software: métricas del modelo de análisis, métricas del modelo del diseño, métricas del código fuente, métricas para pruebas y métricas de mantenimiento. Entre las del modelo de análisis están las métricas basadas en función para predecir el tamaño del sistema que se obtendrá. Ejemplo de ellas son la métrica de punto de función y la métrica bang, las cuales proporcionan medidas cuantitativas para evaluar el modelo de análisis. Otra de las métricas es la de la calidad de la especificación.

Para valorar la calidad de la especificación se mide: especificidad (ausencia de ambigüedad), compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. Además, las especificaciones de alta calidad deben estar almacenadas electrónicamente, ser ejecutables o al menos interpretables, anotadas por importancia y estabilidad relativas, con su versión correspondiente, organizadas, con referencias cruzadas y especificadas al nivel correcto de detalle.

Estos parámetros pueden ser representados usando métricas. Por ejemplo, si hay n_r requisitos en una especificación, tal como $n_r=n_f+n_{nf}$ donde n_f es el número de requisitos funcionales y n_{nf} es el número de

requisitos no funcionales. Para determinar la especificidad (no ambigüedad) de los requisitos se aplica una métrica basada en la consistencia de la interpretación de los revisores para cada requisito: $Q_1=n_{ui}/ n_{r.}$ Siendo n_{ui} el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca esté de 1 el valor de Q, menor será la ambigüedad de la especificación.

La compleción de los requisitos funcionales pueden determinarse calculando la relación $Q_2=u_n/(n_i*n_s)$ donde u_n es el número de requisitos únicos de función, n_i es el número de entradas (estímulos) definidos o implicados por la especificación y n_s es el número de estados especificados. La relación Q_2 mide el porcentaje de funciones necesarias que se han especificado para un sistema. Sin embargo, no trata los requisitos no funcionales(11). Para incorporarlos 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.

Para determinar la corrección (Q_4), se define la cantidad de requisitos correctos(n_c) y la cantidad de requisitos incorrectos (n_i), y se calcula $Q_4 = n_c /n_c + n_i * 100$ obteniendo el porciento de corrección de los requisitos. (39)

Para medir el grado de redundancia (Q_5) se identifican los requisitos repetidos (n_{rr}) y los requisitos no repetidos (n_{nr}), entonces se calcula $Q_5 = n_{rr} / n_{nr} * 100$ obteniendo así el porciento de redundancia de los requisitos. (39)

Selección de las métricas

Se seleccionan las métricas: Especificidad, Corrección y Redundancia para ser aplicadas a la especificación de requisitos debido a que miden algunos de los aspectos fundamentales en una especificación como son la no ambigüedad de los requisitos, que los mismos estén correctos y que no estén repetidos. Además se aplicarán listas de chequeo prepuestas por la universidad para evaluar la calidad de las especificaciones de requisitos y casos de uso.

Conclusiones del capítulo

La aplicación de la ingeniería de requisitos en la construcción de un sistema es trascendental para una correcta implementación del mismo. A través de las diferentes etapas de la misma se pueden obtener, especificar, validar y gestionar los requisitos.

Las herramientas CASE posibilitan un desarrollo más rápido y fácil de los artefactos del sistema. No obstante, no basta con emplear una herramienta para lograr el éxito, también es preciso utilizar una metodología de desarrollo de software. Es válido destacar que no es obligatorio ajustarse estrictamente a una metodología sino que esta puede ser adaptada a las necesidades y características de cada proyecto.

Mediante la verificación y validación de requisitos y el uso de métricas y modelos de calidad se puede medir la calidad de los requerimientos, de las especificaciones de requisitos y de casos de uso, contribuyendo así a comprobar la calidad del sistema incluso antes de implementarlo.

Capítulo 2: Solución Propuesta

En este capítulo se presenta la propuesta al problema planteado, siguiendo la metodología de desarrollo de software RUP y utilizando Visual Paradigm para modelar los artefactos que se generan en dicho proceso de desarrollo. Se muestran los resultados de las etapas obtención, elaboración y especificación de la Ingeniería de Requisitos. Entre ellos el diagrama de procesos de negocio, los requisitos funcionales y no funcionales, los diagramas de casos de uso. Así como las especificaciones de requisitos y casos de uso. Además se presenta la matriz de trazabilidad realizada durante la gestión de requisitos.

Modelamiento del Negocio

El modelamiento del negocio es el primer flujo de trabajo en la fase de Inicio de RUP. Este flujo es imprescindible para comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema, derivar los requerimientos del sistema que va a soportar la organización y fundamentalmente para asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.

Para el modelado del negocio se utilizó la herramienta CASE Visual Paradigm que incorpora BPMN. El mismo estuvo orientado a la actividad/rol, haciendo énfasis en el modelado de los procesos y actores de la empresa. El modelo de procesos de negocio fue de gran ayuda para mejorar la comunicación tanto entre el analista y los desarrolladores como entre el analista y los clientes. A continuación se muestra el diagrama de procesos de negocio con las actividades más importantes. El diagrama de procesos de negocio en su totalidad se puede encontrar en el anexo 6.

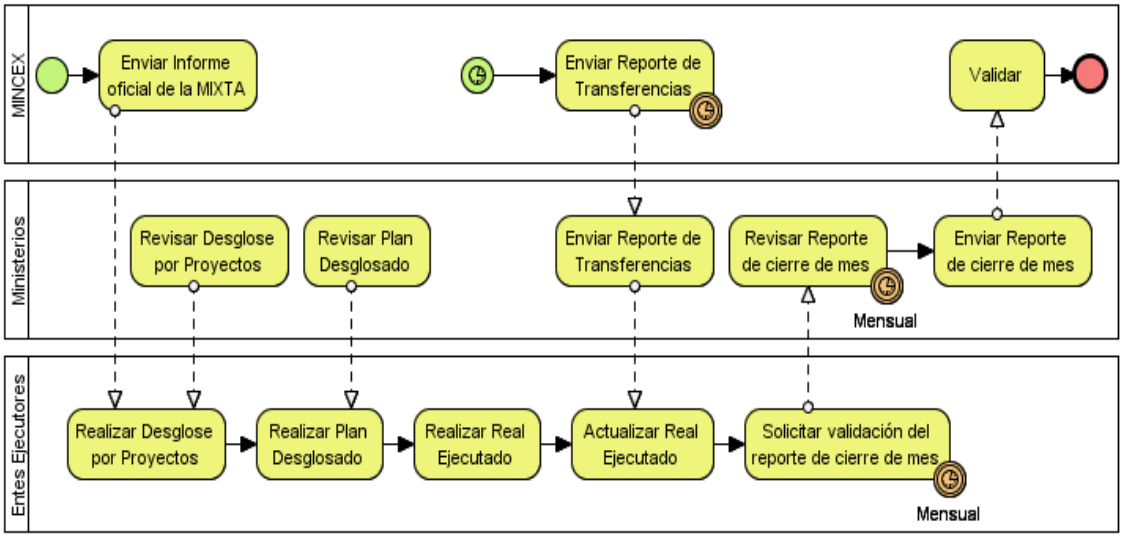


Fig. 5.1 Diagrama de procesos del negocio

Reglas del negocio

Una regla de negocio es una declaración completa y atómica, (es decir, indivisible en otras Reglas) que permite ser expresada de forma inteligible y que, al juntarse con las demás reglas, conforman el marco estructural, la política, la estrategia y la operativa de una empresa u organización. (35)

Las reglas de negocio están presentes en una organización, tanto de manera explícita (una política de salarios, el horario laboral, el descuento a aplicar en función de las condiciones de la venta, etc.) como de manera implícita (el trato cortés con los clientes, la responsabilidad del supervisor sobre sus supervisados, etc.) implicando en general la participación directa o indirecta de personas.

Durante la elaboración de los requisitos fueron identificadas un total de 13 reglas del negocio. Las cuales fueron clasificadas en textuales y de derivación. Las reglas del negocio pueden consultarse en el anexo 7. Algunas de ellas son las que siguen:

No	Tipo	Nombre	Descripción
2	Textual	RN2	El Plan Desglosado una vez que se aprueba no puede ser cambiado, a menos que sea con autorización del MEP.
3	Textual	RN3	El Plan Desglosado en el que se trabaja debe ser del año que transcurre.
10	Textual	RN10	Los ingresos o egresos se deben reflejar en el mes en que llegan al país no a la empresa.
11	Textual	RN11	En el Real Ejecutado es obligatorio hacer una observación cuando los ingresos o egresos superan o son menores que la cantidad planificada.
12	Textual	RN12	Cuando se va a hacer el Plan Desglosado de un nuevo año se tiene que planificar en base al resto que quedo del año anterior.
13	Derivación	RN13	Los montos en el sistema se introducirán en CUP pero el sistema automáticamente los registrará como USD, hará el cambio de moneda de forma automática, teniendo en cuenta el cambio.

Requisitos funcionales

Con el objetivo de capturar los requerimientos del sistema fueron utilizadas las técnicas: entrevista y casos de uso. La entrevista fue la más sencilla y una de las más efectivas ya que permitió lograr un acercamiento a los clientes y establecer la comunicación con ellos. Al finalizar cada entrevista se realizaron breves reuniones entre los miembros del equipo de trabajo para analizar las notas tomadas y las grabaciones realizadas, permitiendo así llegar a un entendimiento común de lo que los clientes necesitaban. Por otra parte, los casos de uso proporcionaron el modo ideal para capturar los requisitos funcionales. A través de los mismos se pudo identificar quiénes eran los actores y qué acciones del sistema llevaban a cabo.

Durante la obtención de requisitos fueron identificados 82 requisitos funcionales:

1. Convertir montos a USD
2. Calcular costo por peso
3. Gestionar Desglose por Proyectos (DPP)
4. Crear DPP
5. Modificar DPP
6. Mostrar información del DPP
7. Efectuar cierre de DPP
8. Pedir confirmación de cierre de DPP
9. Aprobar DPP
10. Pedir confirmación de aprobación del DPP
11. Rechazar DPP
12. Pedir confirmación de rechazo del DPP
13. Especificar motivo de rechazo del DPP
14. Generar reporte del DPP
15. Exportar reporte del DPP
16. Mostrar DPP rechazados con motivo de rechazo
17. Mostrar DPP a revisar
18. Gestionar Plan Desglosado (PD)
19. Crear PD
20. Modificar PD
21. Mostrar información del PD
22. Efectuar cierre de PD
23. Pedir confirmación de cierre de PD
24. Aprobar PD
25. Pedir confirmación de aprobación del PD
26. Rechazar PD
27. Pedir confirmación de rechazo del PD
28. Especificar motivo de rechazo del PD
29. Generar reporte del PD
30. Exportar reporte del PD
31. Mostrar PD rechazados con motivo de rechazo
32. Mostrar PD a revisar
33. Verificar que el monto de ingresos o egresos en el PD esté de acuerdo a lo planificado.
34. Mostrar notificación de variación en el monto de ingresos o egresos en el PD.
35. Gestionar Real Ejecutado (RE)
36. Crear RE
37. Modificar RE

38. Mostrar información del RE
39. Efectuar cierre de RE
40. Pedir confirmación de cierre de RE
41. Aprobar RE
42. Pedir confirmación de aprobación del RE
43. Rechazar RE
44. Pedir confirmación de rechazo del RE
45. Especificar motivo de rechazo del RE
46. Generar reporte del RE
47. Exportar reporte del RE
48. Mostrar RE rechazados con motivo de rechazo
49. Mostrar RE a revisar
50. Verificar que el monto de ingresos o egresos en el RE esté de acuerdo a lo planificado.
51. Mostrar notificación de variación en el monto de ingresos o egresos en el RE.
52. Registrar observación
53. Mostrar Historial de Operaciones
54. Mostrar Historial de Operaciones de Desglose por Proyectos
55. Mostrar Historial de Operaciones de Plan Desglosado
56. Mostrar Historial de Operaciones de Real Ejecutado
57. Mostrar Reportes
58. Mostrar reporte de ingresos
59. Mostrar reporte de egresos
60. Mostrar reporte de exportaciones
61. Mostrar reporte de importaciones
62. Mostrar reporte mixto
63. Mostrar Datos de Proyectos
64. Establecer Tasa de Cambio
65. Habilitar edición de PD
66. Pedir confirmación de habilitación de PD
67. Deshabilitar edición de PD
68. Pedir confirmación de deshabilitación de PD
69. Habilitar edición de RE
70. Pedir confirmación de habilitación de RE
71. Deshabilitar edición de RE
72. Pedir confirmación de deshabilitación de RE
73. Crear usuario
74. Modificar usuario
75. Eliminar usuario
76. Crear rol

77. Modificar rol
78. Eliminar rol
79. Asignar permisos
80. Asignar usuarios
81. Autenticar usuario
82. Extraer información de CCV

Requisitos no funcionales

Se identificaron 24 RNF los cuales pueden ser consultados en el anexo1, y se agruparon en:

Usabilidad

RNF.1 Cumplir con las pautas de diseño de las interfaces de CCV.

El sistema deberá tener una interfaz gráfica uniforme a través del mismo incluyendo pantallas, menús y opciones. Las pautas de diseño serán las mismas definidas en CCV, para lograr una similitud entre los dos sistemas.

RNF.2 Agrupar vínculos y botones por grupos funcionales.

La consistencia de la interacción entre usuario y sistema estará determinada por el diseño de la interfaz de usuario que mantendrá los elementos como menús, banners y zona de trabajo, en posiciones fijas, además de la mayor uniformidad posible entre cuadros de texto y botones.

El sistema deberá ser de uso intuitivo, de tal forma que se reduzca los tiempos de entrenamiento, soporte y prueba por parte del usuario. La agrupación de los botones por funcionalidad determinará además la capacidad de componer la interfaz de acuerdo a las funciones requeridas por un rol determinado.

RNF.3 Mostrar los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema en idioma español.

Tanto los títulos de los componentes de la interfaz, como los mensajes para interactuar con los usuarios, así como los mensajes de error, deberán ser en idioma español y tener una apariencia uniforme en todo el

sistema. Los mensajes de error deberán ser lo suficientemente informativos para dar a conocer la severidad del error.

RNF.4 Utilizar campos de selección en la interfaz en los casos que sea posible.

El sistema deberá facilitar la entrada de datos a los usuarios, presentando campos de selección que permitan escoger los valores. Estos campos contendrán los valores posibles con los que se podrá llenar un determinado elemento en la interfaz, haciendo que el proceso de llenado de datos sea lo más intuitivo posible de tal forma que los usuarios se puedan adaptar con facilidad al sistema.

RNF.5 Permitir uso del teclado para realizar operaciones sobre el sistema (Permitir acceso rápido al sistema usando el teclado)

El sistema debe permitir su uso desde el teclado con funciones básicas como Tab para moverse por los campos de una página determinada, Enter para accionar un botón seleccionado. Se utilizarán tres funciones, dentro de las que se encuentran Ctrl+C para copiar una información seleccionada en un campo específico, Ctrl+V para pegar una información específica y Ctrl+X para cortar la información seleccionada.

RNF.6 Mostrar los valores de los campos numéricos utilizando separadores, según estándares.

El sistema debe ser preciso en cada una de las salidas debido al manejo de dígitos e información referente a diferentes montos asociados a los proyectos. Para esto debe seguir un estándar único en la codificación de las cifras y la separación de los dígitos, para la separación de miles se utilizará la coma y para las cifras decimales el punto (ej. 100. 000.000,00).

Fiabilidad

RNF.7 Prever contingencias para eventos de caída del sistema.

El sistema deberá prever contingencias que pueden afectar la prestación estable y permanente del servicio. Se deberán realizar salvas diarias al sistema para prever posibles pérdidas de la información. La siguiente es la lista de las contingencias que se deben tener en cuenta y se pueden considerar críticas:

- Caída del sistema por sobrecarga de procesos
- Caída del sistema por sobrecarga de transacciones

- Caída del sistema por volumen de datos excedido en la base de datos.

Estas consideraciones implicarán que la infraestructura técnica sobre la que se implantará el sistema garantice una alta disponibilidad del mismo.

Eficiencia

RNF.8 Responder en tiempos aceptables las peticiones que se realicen en el sistema.

El sistema debe ser capaz de dar respuestas a las peticiones con un nivel aceptable de desempeño, como máximo de seis a ocho segundos. Teniendo en cuenta el nivel de concurrencia que pueda existir, debe ser capaz de prestar servicio sin que se deterioren los tiempos de respuestas.

Seguridad

RNF.9 Permitir el intercambio de datos entre el cliente y el servidor por canales cifrados.

El sistema deberá permitir la transmisión por canales cifrados cuando se trate de información confidencial (específicamente por el protocolo HTTPS que utiliza un cifrado basado en SSL), de manera que no viaje en texto plano por la red.

RNF.10 Almacenar de manera cifrada las claves de los usuarios en la base de datos.

Las contraseñas de los usuarios no deben ser almacenadas en texto plano, esta información debe ser privada y específica de cada uno, de manera que nadie pueda reemplazar la identidad de un usuario en el sistema.

RNF.11 Registrar cada una de las operaciones llevadas a cabo por un usuario en el sistema.

Se debe realizar un registro de cada una de las acciones que realiza un usuario en el sistema de manera que se tenga la fecha y hora en se realizó una determinada operación y quien la realizó.

RNF.12 Definir una jerarquía de usuarios para el manejo centralizado de los mismos en el sistema.

Se debe crear una jerarquía de usuarios, los cuales deben ser manejados por administradores de cada uno de los niveles. Las acciones que podrá realizar un usuario estarán determinadas por los roles que tendrá el usuario.

RNF.13 Permitir la creación de roles de usuarios

Los roles serán creados por un administrador central. Los roles contendrán funcionalidades específicas del sistema que determinarán el acceso a las mismas por parte de los usuarios.

RNF.14 Restringir el acceso al sistema.

Para acceder al sistema cada usuario debe ser autenticado, de manera que se pueda determinar si tiene acceso al mismo y en cado de ser así determinar las funcionalidades a las que tiene acceso y restringir su actividad al uso de las mismas las cuales están determinadas por los roles que contiene el usuario. Para autenticarse el usuario debe colocar: nombre de usuario y contraseña.

RNF.15 Identificar autenticaciones forzadas realizadas por programas.

El sistema debe permitir identificar las autenticaciones que puedan ser realizadas por programas con el objetivo de forzar la entrada al sistema.

RNF.16 Gestión de las contraseñas

El sistema debe permitir que cada usuario pueda cambiar su contraseña, así como debe exigir que la misma tenga un nivel aceptable de complejidad (mínimo ocho caracteres, incluyendo caracteres especiales).

Portabilidad

RNF.17 El sistema debe ser una aplicación Web

El sistema se debe ejecutar sobre un entorno web de manera que se permita su acceso desde cualquier punto del país utilizando la red.

RNF.18 Garantizar compatibilidad con navegadores de uso común

El sistema deberá ser compatible con los navegadores:

- Microsoft Internet Explorer 6.0 o superior
- Mozilla Firefox 2.0 o superior.

RNF.19 Utilizar estándares de codificación

El código fuente del sistema deberá cumplir con un estándar de codificación. El estándar especificado debe considerar puntos como: Estándares de nombres utilizados en todos sus objetos: programas, formas, tablas, campos, índices, procedimientos, paquetes. Codificación de los comentarios para la generación automática de la documentación. Empleo de las características del IDE (Interface Development Enviroment) para el formato del código.

RNF.20 Diseñar un sistema compuesto por módulos, que agrupen funcionalidades.

El sistema deberá garantizar que cada subsistema, aplicación y componente tienen fronteras claramente definidas y funciones relacionadas.

RNF.21 Garantizar que las actualizaciones del sistema sean a nivel central (Servidor)

El sistema deberá estar orientado a que las actualizaciones sólo se hagan en el sitio del servidor, de tal manera que no sea necesario actualizar todos y cada uno de los clientes que acceden a la información del sistema.

Reusabilidad

RNF.22 Garantizar que los formatos de los archivos de salida del sistema sean compatibles con los programas más comunes.

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

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

RNF.23 Definir un modelo tres capas para el sistema

El sistema deberá considerar en su arquitectura un modelo tres capas, donde se definen tres componentes lógicos de manera independiente: capa de presentación o interfaz de usuario, capa de lógica de negocio y capa de datos. Esta arquitectura determina una separación entre la lógica del negocio

y la presentación de la aplicación, lo que permite el uso de servicios desde la capa de lógica del negocio ya sea por la capa de presentación del sistema o por otros sistemas que requieran el mismo servicio.

Capacidad

RNF.24 Considerar características técnicas mínimas para la ejecución en clientes

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

- Procesador 2.0 GHz
- Memoria 512 MB
- Disco duro 20 GB
- Sistema Operativo: Windows 98, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows Server 2003 o 2008 o Linux
- Navegador: Internet Explorer 6.0 o posterior o Mozilla Firefox 2.0 o posterior
- Conexión a Internet: mínimo 56 Kbps

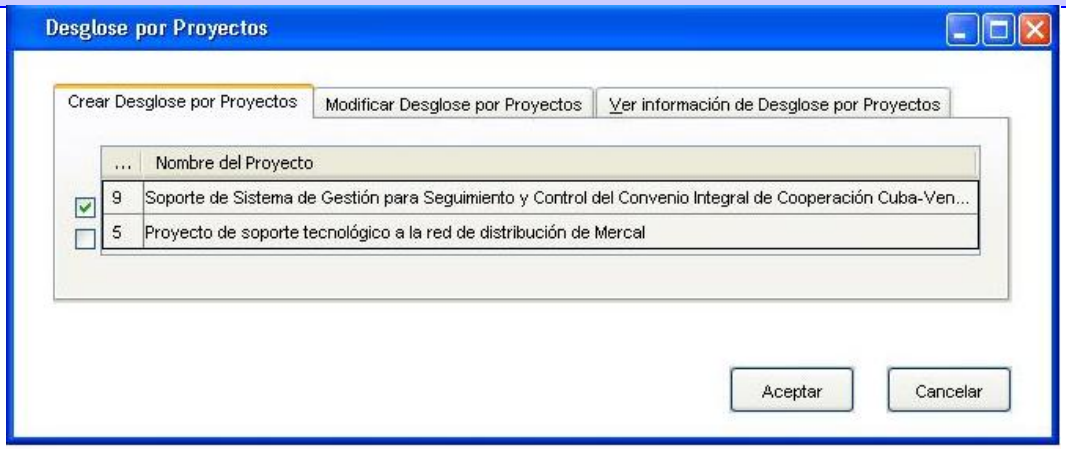
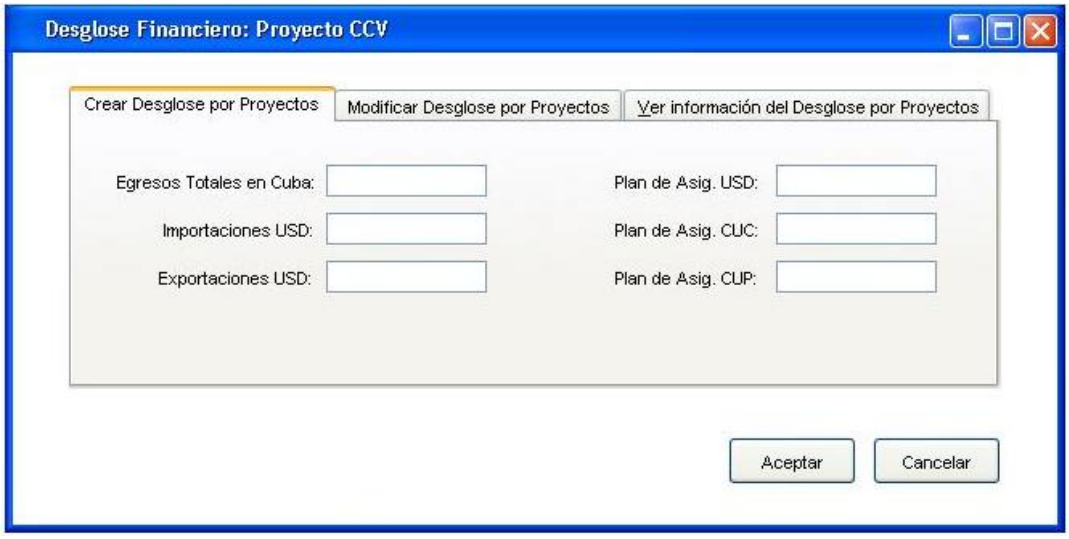
Para que un cliente de la aplicación pueda observar y analizar resultados de los procesos consignados en documentos electrónicos, el punto de acceso deberá cumplir con los siguientes requisitos mínimos:

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

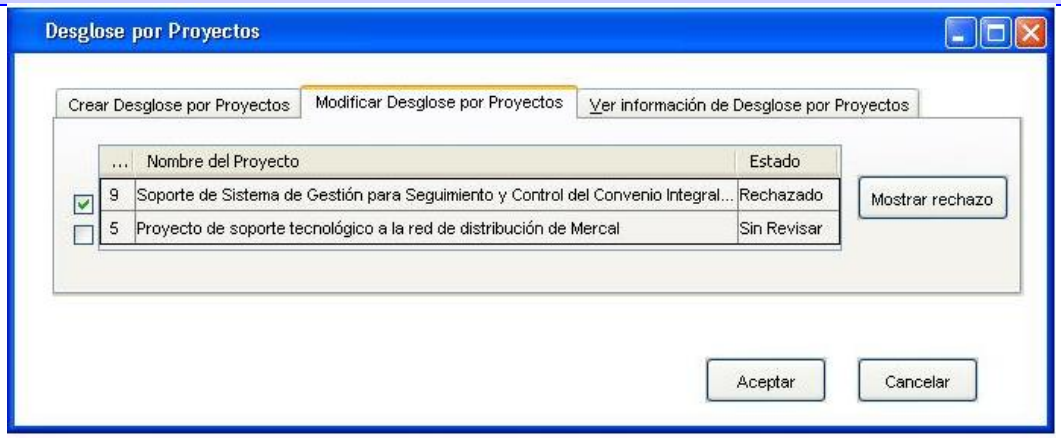
Especificación de requisitos

A continuación se muestra un fragmento de la especificación de requisitos. La especificación de requisitos completa se puede consultar en el anexo1.

RF.3	Gestionar Desglose por Proyectos	El sistema deberá permitir crear el desglose por proyectos y modificarlo en caso de que los datos del mismo varíen	Alta	Alta

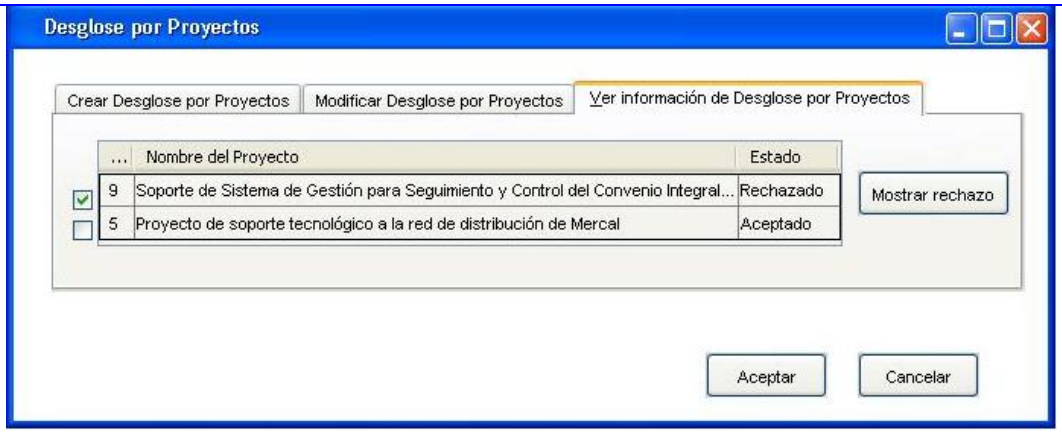
	Observaciones			
RF.4	Crear Desglose por Proyectos	El sistema permitirá crear el Desglose por Proyectos que incluirá información extraída del sistema CCV, el costo por peso en USD y otros datos como los gastos totales en Cuba, importaciones en USD, plan de asignación en USD, en CUC, en CUP.	Alta	Alta
	Prototipos			
				
				
	Campos	**Tipo de Datos**	**Reglas o Restricciones**	

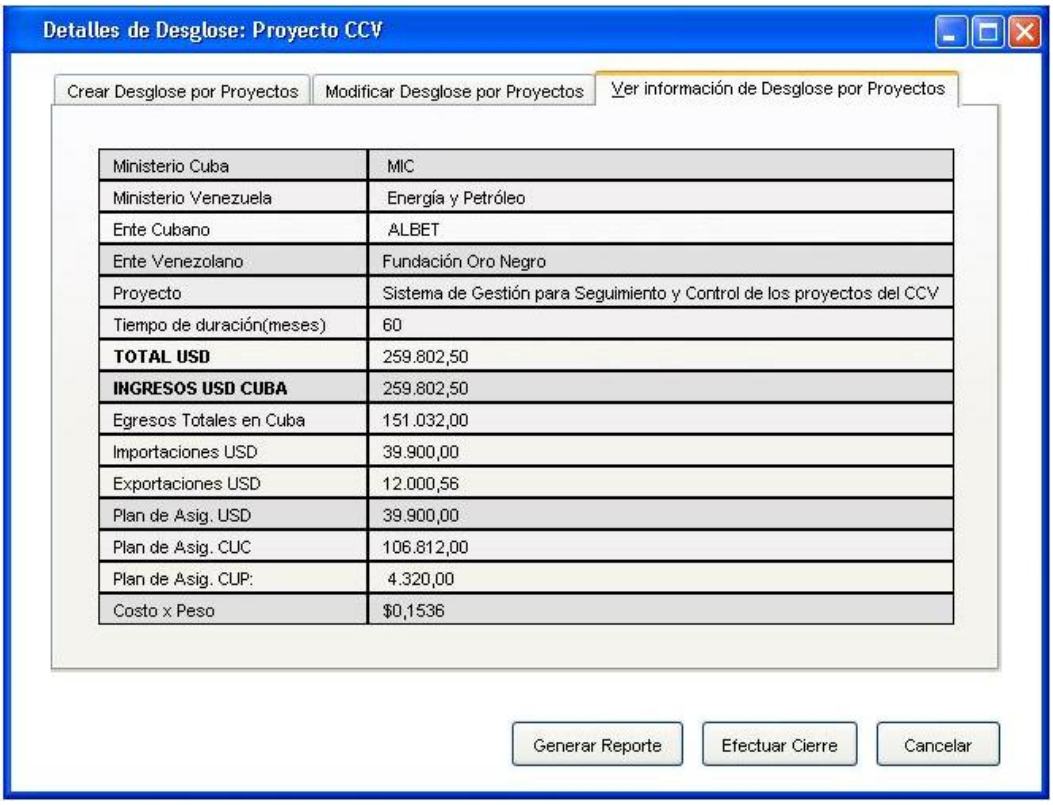
	Egresos Totales en Cuba	Double	No puede ser un valor negativo		
	Importaciones USD	Double	No puede ser un valor negativo		
	Exportaciones USD	Double	No puede ser un valor negativo		
	Plan de Asig. USD	Double	No puede ser un valor negativo		
	Plan de Asig. CUC	Double	No puede ser un valor negativo		
	Plan de Asig. CUP	Double	No puede ser un valor negativo		
	Observaciones				
RF.5	Modificar Desglose por Proyectos	El sistema deberá permitir modificar datos del desglose por proyectos como los gastos totales en Cuba, importaciones en USD, plan de asignación en USD, en CUC, en CUP.		Alta	Alta
	Prototipo				



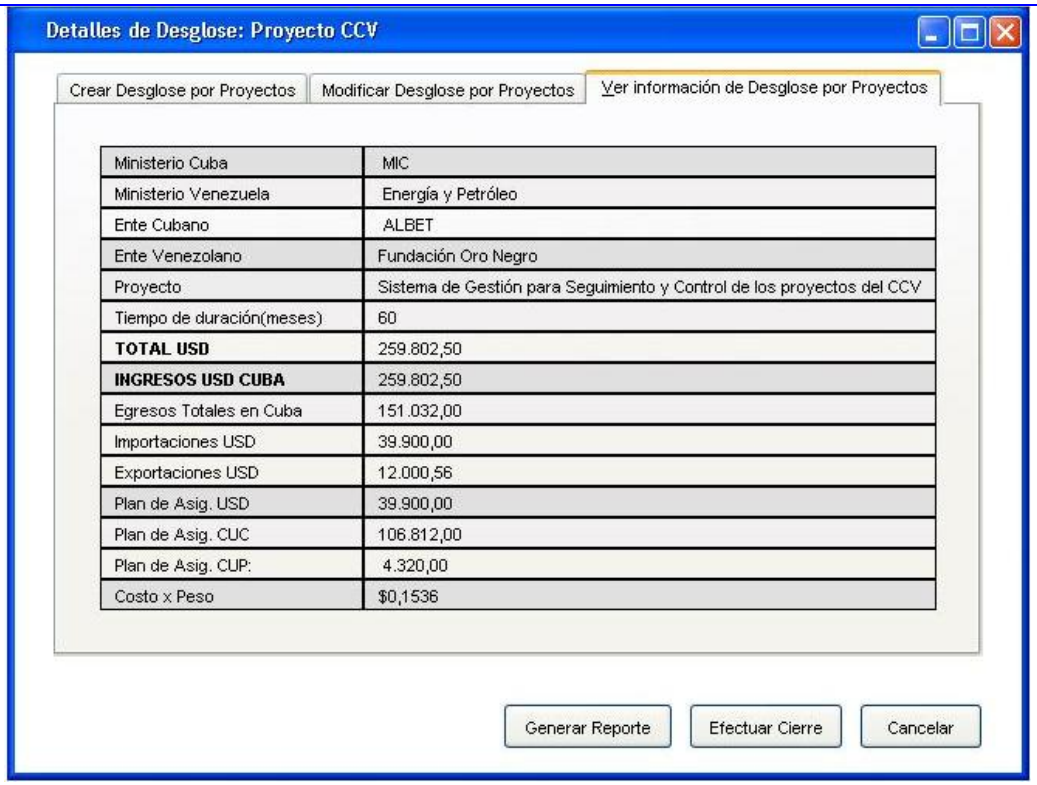
Campos	**Tipo de Datos**	**Reglas o Restricciones**
Egresos Totales en Cuba	Double	No puede ser un valor negativo
Importaciones USD	Double	No puede ser un valor negativo
Exportaciones USD	Double	No puede ser un valor negativo
Plan de Asig. USD	Double	No puede ser un valor negativo
Plan de Asig. CUC	Double	No puede ser un valor negativo
Observaciones		

RF.6	Mostrar información del Desglose por Proyectos	El sistema mostrará los detalles de la información insertada o modificada en el Desglose por Proyectos: Ministerio Cuba, Ministerio Venezuela, Ente cubano, Ente venezolano, Proyecto, Tiempo de duración, Total USD, Ingresos USD Cuba, Egresos Totales en Cuba, Importaciones USD, Exportaciones USD, Plan de Asignación USD, Plan de Asignación CUC, Plan de Asignación CUP, Costo por Peso.	Media	Alta
	Prototipo			





	Observaciones			
RF.7	Efectuar cierre de Desglose por Proyecto	El sistema permitirá efectuar el cierre, lo que significa que la información está completa y lista para ser revisada.	Media	Alta
	Prototipo			



	Observaciones			
RF.8	Pedir confirmación de cierre del Desglose por Proyecto	El sistema pedirá al usuario una confirmación sobre el cierre del Desglose por Proyecto.	Baja	Media
	Observaciones			

Diagrama de Paquetes

El siguiente diagrama de paquetes muestra cómo el sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.

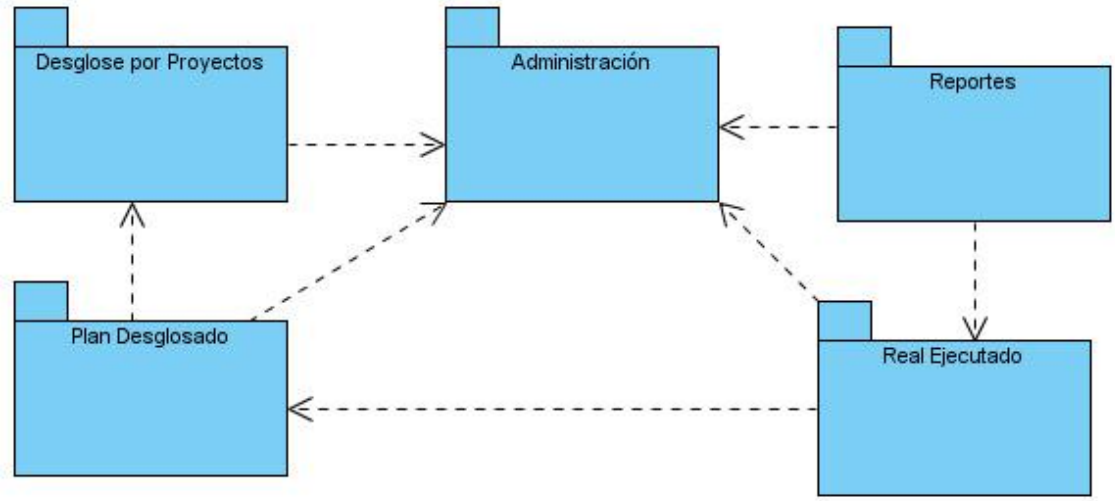


Fig. 7 Módulos del sistema

Actores

ACTORES	DESCRIPCION
Administrador	Es el encargado de gestionar roles, usuarios, permisos.
MINCEX **(Ministerio de Comercio Exterior)**	Crea y envía el informe oficial de la Mixta y el reporte de transferencias.
MEP **(Ministerio de Economía y Planificación)**	Aprueba, habilita o deshabilita el PD y RE. Es el único que autoriza modificaciones al PD.
Ministerios	Validan la información presentada por los entes ejecutores. Conforman su DPP, PD o RE a partir de la información enviada por sus entes ejecutores.

Entes Ejecutores	Entran o modifican los datos del DPP, PD y RE. Visualizan información sobre ellos y efectúa el cierre de los mismos.
Aplicación CCV	De esta aplicación se extraen los datos de los proyectos de los que se desea controlar el financiamiento.
Usuario	Todo el usuario que interactúa con el sistema. Puede generar reportes de DPP, PD y RE en dependencia de la jerarquía que posea.
Revisor	Es un rol común para los ministerios y para el MINCEX. Aprueba o rechaza el Desglose por Proyectos, Plan Desglosado y Real Ejecutado.

Diagrama de casos de uso

Para la realización del diagrama de casos de uso se utilizaron patrones de casos de uso como Reglas del negocio, Actores múltiples: Roles comunes, Extensión o Inclusión Concreta: Extensión y Extensión o Inclusión Concreta: Inclusión, lo cual es una buena práctica, ya que siempre es bueno utilizar una solución común y probada con anterioridad para resolver un problema, en lugar de encontrar una solución propia.

Como se muestra en la Fig. 8 se identificaron un total de 22 casos de uso. Este diagrama representa los actores del sistema y su relación con las funcionalidades que dicho sistema debe realizar. Sin embargo, el anterior diagrama de casos de uso resulta un poco extenso. Por ello se decidió agrupar los casos de uso en paquetes mas pequeños que contienen funcionalidades similares de acuerdo a los módulos que contiene el sistema y de su relación con los actores.

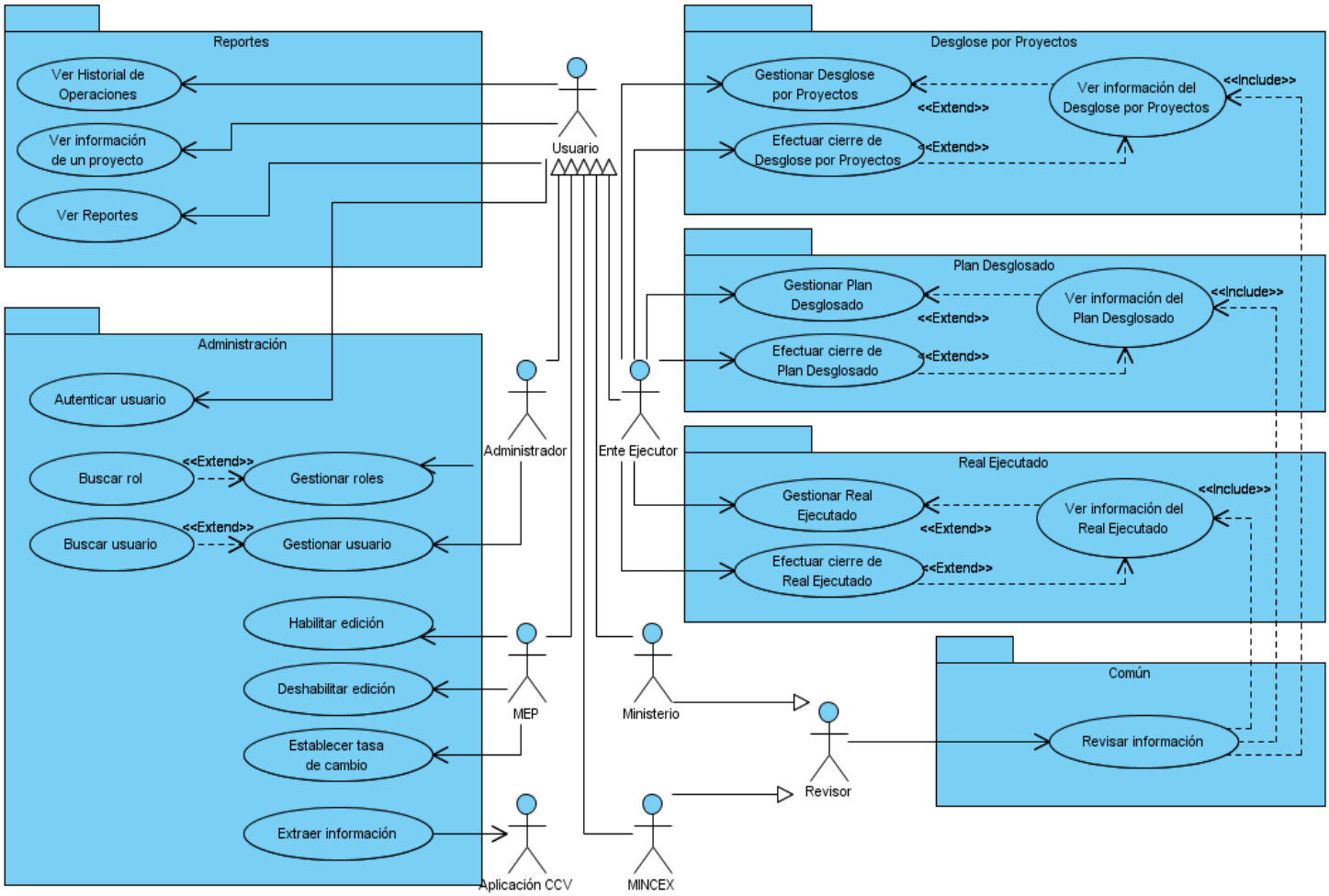


Fig. 8 Diagrama de casos de uso del sistema.

Descripción de casos de uso

A continuación se muestra un fragmento de la especificación de casos de uso, para el caso de uso Gestionar Rol del módulo Administración. La totalidad de las descripciones de casos de uso se muestra en el anexo 2.

CU Gestionar Desglose por Proyectos

Objetivo	Introducir o modificar los datos del Desglose por Proyectos
Actores	Ente Ejecutor: (Inicia) Adiciona y modifica los datos del Desglose por Proyectos en el sistema.
Resumen	Permite introducir o modificar los datos del Desglose por Proyectos, a partir de los datos cargados de la aplicación CCV.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El sistema ha sido instalado y ejecutado correctamente. El usuario se ha autenticado con el permiso correspondiente. Se han cargado los datos de la Aplicación CCV
Postcondiciones	Se adicionaron o modificaron datos en el Desglose por Proyectos
Flujo de eventos	
Flujo básico Gestionar Desglose por Proyectos	

	Actor	**Sistema**
1.	Solicita la gestión del Desglose por Proyectos	
2.		Muestra una interfaz que permite realizar las acciones: - Crear el Desglose por Proyectos. Ver Sección 1: “Crear Desglose por Proyectos”. - Modificar el Desglose por Proyectos. Ver Sección 1: “Modificar Desglose por Proyectos”.

Flujos alternos	
Sección 1: “Crear Desglose por Proyectos”	

Flujo básico Crear Desglose por Proyectos		
	Actor	**Sistema**
1.	Solicita la creación del Desglose por Proyectos	
2.		Muestra una interfaz con la lista de proyectos.
3.	Selecciona el proyecto y acepta	
4.		Muestra una interfaz para entrar los siguientes datos: - Egresos totales en Cuba - Importaciones USD - Exportaciones USD - Plan de asignación en USD - Plan de asignación en CUC - Plan de asignación en CUP
5.	Introduce los datos	
6.		Calcula el costo por peso a partir de los datos introducidos
7.	Acepta los datos	
8.		Guarda los datos
9.		Termina el caso de uso
Flujos alternos		
3 a. Los datos están incompletos o son incorrectos		
	Actor	**Sistema**
1.		Muestra un mensaje de error indicando que los datos son incorrectos o están incompletos
2.	Retorna al paso 5 del Flujo básico Crear Desglose por Proyectos	

Sección 2: "Modificar Desglose por Proyectos"		
Flujo básico Modificar Desglose por Proyectos		
	Actor	**Sistema**
1.	Solicita la modificación del Desglose por Proyectos	
2.		Muestra una interfaz con la lista de proyectos con estado Rechazado o Sin Revisar, para que el usuario seleccione el proyecto del cual desea modificar el Desglose por Proyectos
3.	Selecciona el proyecto y acepta	
4.		Muestra una interfaz para entrar los siguientes datos: - Egresos totales en Cuba - Importaciones USD - Exportaciones USD - Plan de asignación en USD - Plan de asignación en CUC - Plan de asignación en CUP
5.	Introduce los datos	
6.		Calcula el costo por peso a partir de los datos introducidos
7.	Acepta los datos	
8.		Guarda los datos
9.		Termina el caso de uso
Flujos alternos		
3 a. Los datos están incompletos o son incorrectos		
	Actor	**Sistema**
1.		Muestra un mensaje de error indicando que los datos

		son incorrectos o están incompletos
2.	Retorna al paso 5 del Flujo básico Modificar Desglose por Proyectos	
Relaciones	**CU Incluidos**	
	CU Extendidos	*Ver información de Desglose por Proyectos*

Gestión de Requisitos

Los requisitos fueron sometidos a cambios en varias ocasiones. Para lograr gestionarlos se les asignó un identificador y se llevó a cabo una tabla de rastreabilidad donde se relacionaron los requisitos con determinados aspectos del sistema, en este caso, con los casos de uso, lo que propició identificar, controlar y seguir los requisitos y sus cambios en los momentos en que tuvieron lugar.

Matriz de trazabilidad

La trazabilidad de requisitos es considerada un proceso imprescindible para la adecuada gestión de requisitos. La matriz de trazabilidad es una tabla que relaciona cada requisito con su origen, y permite trazarlo durante todo el proyecto. En ocasiones suele incluir el estado del requisito: si está pendiente, si ya se ha incluido en el producto y cuándo, si se ha cancelado.

A continuación se muestra un fragmento de la matriz de trazabilidad, la cual consiste en relacionar cada caso de uso con sus requisitos correspondientes para comprobar que las funcionalidades definidas incluyen todos los requisitos y no se omite ninguno, verificando así la completitud de las mismas. La matriz de trazabilidad de requisitos se puede consultar en su totalidad en el anexo 3.

Requisitos	CU 2	CU 6	CU 7	CU 8	CU 9	CU 10	CU 11	CU 12
RF 64								x
RF 65						x		
RF 66						x		
RF 67							x	
RF 68							x	
RF 73		x						
RF 74		x	x					
RF 75		x	x					
RF 76				x				
RF 77				x	x			
RF 78				x	x			

Fig. 9 Matriz de trazabilidad

Conclusiones del Capítulo

La aplicación de la ingeniería de requisitos permitió la obtención de requisitos funcionales y no funcionales, tarea que se hizo más fácil al aplicar las técnicas de identificación de requisitos como la entrevista y casos de uso. También se llevó a cabo la especificación de los requisitos, los cuales fueron controlados y se les dio seguimiento a través de la matriz de trazabilidad.

Como producto del empleo de la metodología y herramienta, seleccionadas previamente en el Capítulo 1, se obtuvieron varios artefactos que ayudaron a una mayor comprensión del negocio y del problema a solucionar.

Capítulo 3: Análisis de los Resultados

En el presente capítulo se presentan los resultados de la validación de los requisitos a través del empleo de las técnicas revisiones técnicas formales y creación de prototipos, y de la utilización de métricas presentadas por Alan M. Davis en su modelo de calidad, para medir la calidad de la especificación de requisitos. Además se muestra el resultado de la aplicación de listas de chequeo propuestas por la UCI para comprobar distintos indicadores de calidad en la especificación de requisitos y la especificación de casos de uso.

Validación de requisitos

Como técnica de validación se utilizó la revisión de requerimientos por parte de un equipo de revisores integrado por los ingenieros Linnet Acosta (Jefa del proyecto), Ariel Morales (Arquitecto) y Daymeri Díaz (Analista), los cuales revisaron sistemáticamente el documento de especificación de requisitos, llegando a la conclusión de que los requisitos son posibles de probar de manera realista, que son comprensibles y que son adaptables, o sea, que si se cambia un requisito, esto no repercute altamente en los demás requerimientos del sistema. También se utilizó la técnica de validación construcción de prototipos, mediante la cual el cliente puso constatar si el sistema abarcaba sus necesidades. En este caso fueron prototipos no funcionales que fueron mostrados a los clientes y con los cuales estuvieron de acuerdo.

Fig. 10 Prototipo no funcional de Desglose por Proyectos

Métricas para la especificación de requisitos

Para comprobar la calidad de la especificación de requisitos se utilizaron métricas definidas por Davis en su modelo de calidad, para medir varios criterios como la especificidad o no ambigüedad, la corrección y el porciento de redundancia. Teniendo la cifra de 82 requisitos funcionales (n_f) y 24 requisitos no funcionales (n_{nf}) se arriba a un total (n_r) de 106 requisitos. De dichos requisitos, 104 tuvieron interpretaciones idénticas por todos los revisores. Una vez aplicada la métrica para determinar la especificidad (Q_1), se obtuvo un porciento de 0.98, el cual es un valor muy cercano a 1. Mientras más cerca esté de 1 el valor de Q, menor será la ambigüedad de la especificación, con lo cual se puede concluir que el grado de especificidad de la especificación de requisitos es correcto.

Cantidad de requisitos	**n_r**	**106**
Cantidad de requisitos de interpretaciones idénticas	**n_{ui}**	**104**
Especificidad	**$Q_1=n_{ui}/ n_r$**	**0.98**

Para determinar la corrección (Q_2), siendo la cantidad de requisitos correctos 106 (n_c) y 0 la cantidad de requisitos incorrectos (n_i), se calcula $Q_2= n_c /n_c + n_i * 100$ obteniendo una corrección de 100%, lo que significa que los requisitos están correctamente especificados.

Cantidad de requisitos correctos	**n_c**	**106**
Cantidad de requisitos incorrectos	n_i	0
Corrección	$Q_2 = (n_c/n_c+ n_i)* 100$	100%

Para medir el porciento de redundancia (Q_3) se identificaron 0 requisitos repetidos (n_{rr}) y 106 requisitos no repetidos (n_{nr}), de lo cual se puede deducir que el porciento de redundancia es 0%.

Cantidad de requisitos no repetidos	**n_{nr}**	**106**
Cantidad de requisitos repetidos	n_{rr}	0
Redundancia	$Q_3 = n_{rr} / n_{nr} * 100$	0 %

Listas de Chequeo

Al documento de especificación de requisitos le fue aplicada, además, una lista de chequeo propuesta por la UCI. La lista está compuesta por varios indicadores como la estructura del documento y los elementos definidos por la metodología. En la misma la columna Peso define si el indicador a evaluar es crítico o no, mientras que la columna Eval proporciona una evaluación para el indicador, que puede ser: 1 en caso de mal y 0 en caso que elemento revisado no presente errores. La columna Cantidad de elementos afectados muestra la cantidad de errores encontrados sobre el mismo indicador y en Comentario se incluyen los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo. El siguiente es solo un fragmento de dicha lista, la cual se puede consultar en su totalidad en el anexo 4.

Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Se han identificado los requisitos funcionales que son las características que el sistema debe cumplir?	0			
crítico	2. ¿Se han identificado los requisitos no funcionales del sistema, que son las cualidades que el sistema debe tener?	0			
crítico	3. ¿Están todos los requisitos redactados de forma simple y clara para aquellos que vayan a consultarlo en un futuro? Ver documento de Especificación de Requisitos.	0			
crítico	4. ¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo? Ver documento de Especificación de Requisitos.	0			

La anterior lista de chequeo fue aplicada en varias revisiones realizadas por el equipo de calidad compuesto por Ivian, Daymara y Eilianis. Dicho equipo detectó un total de nueve no conformidades en la primera revisión, dos no conformidades en la segunda revisión y ninguna no conformidad en la tercera revisión, en la cual todas fueron solucionadas, llegando a la conclusión de que la especificación de requisitos es correcta. Un ejemplo de esas no conformidades lo constituye:

Elemento	No conformidad	Aspecto correspondiente	Clasificación	Estado	Respuesta del Equipo
Especificación de requisitos	Falta especificar los campos necesarios a extraer del sistema CCV	Requisito Funcional 82	Significativa	Resuelta 21-4-2011	Se agregaron los campos que corresponden a los datos a extraer

También se aplicó una lista de chequeo propuesta por la UCI al documento de especificación de casos de uso, con formato similar a la anterior. El siguiente es un fragmento de dicha lista, la cual se puede consultar en su totalidad en el anexo 5.

Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Nombre del Caso de Uso					
crítico	1. ¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?	0			
Actores del CU					
crítico	1. ¿El Caso de Uso está relacionado con al menos un actor?	0			

crítico	2. ¿Si hay dos actores interactuando con el caso de uso está generalizado en uno solo?	0			
Casos de uso incluidos y extendidos					
crítico	1. ¿Al describir el caso de uso base se mencionan todos los casos de Uso que Extienden, se incluyen o se generalizan del Caso de Uso?	0			
Información General					
crítico	1. ¿El diagrama de casos de uso expresa en detalles lo que debe hacer el sistema?	0			

La anterior lista de chequeo fue aplicada en varias revisiones realizadas por el equipo de calidad el cual detectó un total de seis no conformidades en la primera revisión, una no conformidad en la segunda revisión y ninguna no conformidad en la tercera revisión, en la cual todas fueron solucionadas, llegando a la conclusión de que la especificación de casos de uso es correcta. Un ejemplo de esas no conformidades lo constituye:

Elemento	No conformidad	Aspecto correspondiente	Clasificación	Estado	Respuesta del Equipo
Diagrama de Casos de Uso del Sistema	Algunos nombres de casos de uso están con minúscula y otros con mayúscula.	Casos de uso	Significativa	Resuelta 23-4-2011	Se renombraron los casos de uso

Conclusiones del capítulo

El uso de las técnicas revisión de expertos y la construcción de prototipos contribuyó a la validación de los requisitos pudiendo comprobar que los mismos son posibles de probar, comprensibles y adaptables.

Mediante la aplicación de la métrica que mide la no ambigüedad a la especificación de requisitos, se comprobó el alto grado de especificidad que presenta dicho documento.

La aplicación de listas de chequeo contribuyó a medir la corrección de la especificación de requisitos y la de la especificación de casos de uso, teniendo en cuenta distintos indicadores propuestos en las listas, con lo cual se comprobó que ambas especificaciones están correctamente redactadas.

Conclusiones

A partir de la investigación realizada se evidenció la necesidad de construir un nuevo sistema para la gestión de los procesos de financiamiento para los proyectos del Convenio Cuba-Venezuela. Mediante la utilización de la metodología RUP y la utilización de la herramienta CASE Visual Paradigm se facilitó el desarrollo de los artefactos.

La aplicación de la ingeniería de requisitos permitió la obtención de requisitos funcionales y no funcionales, a través del empleo de las técnicas de identificación de requisitos como entrevista y casos de uso. También se obtuvo como producto la especificación de los requisitos y la especificación de casos de uso que detallaron explícitamente los requerimientos y casos de uso.

La verificación y validación de requisitos unido a la aplicación de métricas y listas de chequeo a las especificaciones de requisitos y casos de uso, resultaron el mecanismo propicio para evaluar los resultados obtenidos y comprobar la calidad de los mismos.

La obtención, especificación y validación de los requisitos contribuyó a traducir las necesidades del cliente a un lenguaje común para los desarrolladores, lo que incide de manera positiva para una futura implementación del sistema.

Para el posterior desarrollo del sistema SIGEF se recomienda:

- Utilizar la ingeniería de requisitos llevada a cabo en este trabajo como base para el diseño y la implementación del sistema.

- Profundizar en la gestión de requisitos para seguir el desarrollo de los mismos durante todo el ciclo de vida del producto y para controlar los cambios que puedan surgir.

Referencias Bibliográficas

1. BSI Mexico. [En línea] [Citado el: 14 de 3 de 2011.] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

2. De Finanzas. [En línea] [Citado el: 9 de 3 de 2011.] http://definanzas.com/2008/06/02/erps-sistemas-de-gestion-empresarial/.

3. Aner Sistemas Informáticos. [En línea] [Citado el: 14 de 3 de 2011.] http://www.aner.com/software-de-gestion-empresarial/software-erp-pymes-prowin.html.

4. **Brito, Ing. Henry Raúl González.** Scribd. *ERP cubano, un paso estratégico para la consolidación del Software Libre en Cuba.* [En línea] 10 de 2006. http://es.scribd.com/doc/38668119/ERP-Cubano.

5. **Cuba, Consejo de Estado de la Republica de.***Sobre la continuidad y el fortalecimiento del sistema de dirección y gestión empresarial cubano. Decreto – Ley 252 de 7 de agosto de 2007. 8pp.* 2007.

6. **Ing.Ruíz Torres, Diana Rosa y Dr. Ing Crist, Carlos.** Gestiopolis. [En línea] 27 de 9 de 2010. [Citado el: 9 de 3 de 2011.] http://www.gestiopolis.com/organizacion-talento-2/sistema-gestion-capital-humano-entidades-alojamiento-turistico-cuba.htm.

7. El Eco del Contador. [En línea] 18 de 10 de 2008. [Citado el: 9 de 3 de 2011.] http://elecodelcontador.blogspot.com/2008/10/el-versat-sarasola-sistema-cubano-de.html.

8. Assets Sistema de Gestión Integral. [En línea] [Citado el: 15 de 3 de 2011.] http://www.assets.co.cu/assets.asp.

9. Informática-Habana. [En línea] [Citado el: 15 de 3 de 2011.] http://www.informaticahabana.cu/node/2887.

10. Desarrollo ERP/CRM. [En línea] 10 de 2 de 2009. [Citado el: 15 de 3 de 2011.] http://desarrolloerp.blogspot.com/2009/02/el-erp-cubano-tiene-nombre-cedrux.html.

11. **Pressman, Roger S.***Ingeniería de Software. Un enfoque práctico. Quinta Edición.* s.l. : McGraw-Hill .

12. —. *Ingeniería de Software. Un enfoque práctico.Sexta Edición.* s.l. : McGraw-Hill , 2005. ISBN: 9789701054733.

13. **Christel, Michael G. y Kang, Kyo C.***Issues in Requirements Elicitation.* Pittsburgh, Pennsylvania : Software Engineering Institute, CarnegieMellonUniversity, 1992. 15213.

14. **Sommerville, Ian.***Ingeniería del Software. Séptima edición.* Madrid : Addison Wesley, 2005. ISBN 84-7829-074-5.

15. **Durán Toro, Amador y Bernárdez Jiménez, Beatriz .***Metodología para la Elicitación de Requisitos de Sistemas Software.* Sevilla : Ministerio de Educación y Ciencia de España, 2000. LSI–2000–10.

16. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.***El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2000. ISBN: 84-7829-036-2.

17. **Leiva Zuñiga, Yurisleidys.***Análisis del Módulo de Soporte del Proyecto Informatización del Convenio Cuba Venezuela.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

18. **Larman, Craig.***UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice hall, 1999. ISBN: 970-17-0261-1.

19. Universidad de Carabobo, FACYT. [En línea] [Citado el: 17 de 3 de 2011.] http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/download/exposiciones2006-2007/presentacion%20UML.pdf.

20. **Ruiz, Francisco.***Proceso Software y Gestión del Conocimiento. Procesos de Negocio.* Ciudad Real : Universidad de Castilla-La Mancha.

21. Scrbd. [En línea] [Citado el: 14 de 3 de 2011.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

22. MundoInformática. [En línea] [Citado el: 16 de 3 de 2011.] http://mundoinformatica.portalmundos.com/axure-rp-pro-5602089-disena-prototipos-navegables-de-paginas-web/.

23. Portalnet.cl. [En línea] 9 de 10 de 2007. [Citado el: 16 de 3 de 2011.] http://www.portalnet.cl/comunidad/showthread.php?t=319311.

24. EcuRed. [En línea] [Citado el: 16 de 3 de 2011.] http://www.ecured.cu/index.php/Rational_Rose_Enterprise_Edition.

25. Rational Rose Enterprise. [En línea] [Citado el: 16 de 3 de 2011.] http://www.rational.com.ar/herramientas/roseenterprise.html.

26. Sparx Systems. [En línea] [Citado el: 16 de 3 de 2011.] http://www.sparxsystems.com.ar/products/ea.html.

27. Free Download Manager. [En línea] 3 de 5 de 2007. [Citado el: 16 de 3 de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

28. **Ing. Valdez Altamirano, Alfonso.** ubicuos.com. [En línea] 5 de 2009. [Citado el: 16 de 3 de 2011.] http://ubicuos.com/wp-content/uploads/2009/05/comparativoides.pdf.

29. **Mendoza Sanchez, María A.** informatizate. *Metodologías De Desarrollo De Software.* [En línea] 7 de 6 de 2004. [Citado el: 23 de 2 de 2011.] http://www.informatizate.net.

30. **Molpeceres, Alberto.** javaHispano. *Procesos de desarrollo: RUP, XP y FDD.* [En línea] 15 de 12 de 2002. [Citado el: 23 de 2 de 2011.] http://www.javaHispano.org.

31. Grupo EUMEDNET de la Universidad de Málaga. [En línea] 2009. [Citado el: 17 de 3 de 2011.] http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.

32. Scribd. [En línea] [Citado el: 17 de 3 de 2011.] http://es.scribd.com/doc/11445983/Procesos-Software.

33. **Martínez García, José Angel y Carrasco Ramírez, Yudelkys .***Análisis y Diseño del módulo de Fichas Planes de Personal del Sistema Nacional Público para el Seguimiento de las Inversiones y Sectores.* Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2010.

34. **Letelier , Patricio y Penadés, María Carmen .***Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia.

35. **Trilles, Juan José .** Aura Portal. *Reglas de Negocio (BR) y Gestión por Procesos de Negocio (BPM) .* [En línea] www.auraportal.com.

36. **Sommerville, Ian.***Ingeniería del Software. Séptima edición.* Madrid : Pearson Educación. SA, 2005. ISBN 84-7829-074-5.

37. **Jiménez, Beatriz Bernárdez y Durán Toro, Amador.** Una Aproximación Empírica a la Verificación de Especificaciones de Requisitos para Sistemas de Información. [En línea] Mayo de 2003. http://www.lsi.us.es/docs/doctorado/memorias/Memoria_ProyectoInvestigador.pdf.

38. La tecla de Escape. [En línea] 17 de 10 de 2010. [Citado el: 17 de 3 de 2011.] http://latecladeescape.com/ingenieria-del-software/metodologias-de-desarrollo-del-software.html.

39. **Tamayo Carvajal, Arelis y Hernández Rosario, Elisabeth.***Ingeniería de requisitos de una herramienta para la gestión de requisitos en los proyectos productivos que se desarrollan en la Universidad de las Ciencias Informáticas.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2009-2010.

40. **Pérez, Juan Diego.** *Notaciones y lenguajes de procesos. Una visión global.* Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla.

Glosario de Términos

CASE: acrónimo de ComputerAided Software Engineering (Ingeniería de Software Asistida por Ordenador), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Caso de uso:fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

Medida: indicación cuantitativa de la extensión,la cantidad, la dimensión o el tamaño de algúnatributo de producto o proceso. Tomado en unpunto del tiempo o lugar.

Métrica: medida cuantitativa del grado en queun sistema, componente o proceso posee unatributo determinado. Reúne y relacionamedidas en varios puntos (promedio, mediana, tasa, razón).

Requisito: condición necesaria para algo.

Trazabilidad:conjunto de medidas, acciones y procedimientos que permiten registrar e identificar un determinado producto desde su nacimiento hasta su destino final.

Anexos

1- Documento Especificación de Requisitos.

2- Documento Especificación de CU.

3- Documento Matriz de Trazabilidad de Requisitos.

4- Lista de Chequeo Especificación de Requisitos.

5- Lista de Chequeo Especificación de CU.

6- Diagrama de Procesos de Negocio.

7- Documento Reglas del Negocio.