UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Implementación de funcionalidades que faciliten la realización de las actividades del proceso de Administración de Requisitos en la herramienta OSRMT.

Autores: Anamaris Sandamil Marcelo

Dulce Genisel Silverio Torres

Tutores: Ing. Kariné Ramos Blanco

Ing. Dennis Neuland Agüero

Ciudad de La Habana, junio del 2010

…Nadie puede asegurar nunca que sus puntos de vista carecen de subjetivismo; siempre se puede correr el riesgo de parecer injusto.
Quien afirma algo, debe estar dispuesto a demostrar lo que dice y por qué lo dice…”

Fidel Castro Ruz

Declaración de Autoría

Declaramos que _____ y _____ somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad (2) para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de julio del 2009.

_____ _____

Firma del Autor Firma del Autor

<Autor 1> <Autor 2>

_____

Firma del Tutor
<Tutor>

A nuestros padres…

De Anamaris:
A mis padres:
Gracias por convertirme en la persona que soy, por educarme bajo principios que no todos saben enseñar.
A mi papá por ser ese hombre que ha depositado en mí toda su confianza, gracias por enseñarme lo que es ser un buen padre, un amigo y un mejor hijo, a ti te debo todo lo que soy y por eso el estar aquí es mi regalo.
A mi madre la amiga que siempre quise tener, la que me hace ver el camino correcto y quien a pesar de equivocarme y cometer errores me apoya a seguir adelante.
Mi hermana pequeña, mi mayor orgullo, las cosas que siempre te digo no son solamente para que sigas mis pasos, sino que vallas adelante.
A mi cosilla por todo este tiempo junto a mí, por soportar mis malcriadeces y mis celos, todo es porque te quiero mucho.
A mi extensa familia, gracias por siempre ayudarme en todo y enseñarme que siempre que me proponga algo lo puedo alcanzar. A mi tío Carlos que ha sido como un padre todo este tiempo, a mis tías Eloilda y Mariela mis otras mamás, mis primas casi hermanas Damaris y Aurora, y a todos mis vecinos.
A los nuevos que serán los viejos amigos de universidad, mi incansable compañera de tesis quien desde el primer día en la UCI fue mi amiga. A mi buen amigo Javier que siempre que he necesitado de él, ha estado dispuesto a ayudarme, a las fieras, por todos estos años de aguante, a las chicas del 208, y a todos

mis compañeros de aula del 2102, siempre será mi verdadero grupo.
A todos les debo un poquito de lo que soy, por eso y mucho más…
Gracias.

De Genisel:
A mi madre linda y querida que siempre está a mi lado incondicionalmente dando todo de ella, la que siempre me espera no importa la hora, la que conoce muy bien mis defectos y jamás me ha faltado, por el contrario me ha llenado de afecto, esa mujer, amiga, hermana y madre que siempre me ha guiado por el buen camino. Gracias mamita por darme la vida y por estar siempre junto a mi todos esos lindos y malos momentos, eres la luz que ilumina mi vida y que siempre estará ahí, te adoro mamita linda.
A ti mi tía querida por ser esa personita que me apoya y me ayuda siempre que lo necesito, gracias por estar a mi lado y escuchar mis problemas y cuentos, por reír y llorar conmigo, eres mi segunda madre espero sepas que te quiero mucho.
A mis abuelitos y tío que son una parte importante en mi vida, gracias por preocuparse por mí y quererme tanto.
A ti Anailys muchas gracias por estar ahí siempre que te he necesitado dándome apoyo, confianza y enseñándome a luchar y tener fe cuando todo está perdido, gracias por soportar todas mis malcriadeces en estos últimos días y por darme ese apoyo y cariño que he necesitado, gracias por existir y estar a mi lado te debo mucho en la vida, te quiero.
A mis amigos de la universidad muchas gracias por estos años tan lindos que hemos pasado juntos, las fiestas, las tertulias de café, historias que nunca olvidare, a las chicas del apartamento gracias por estar ahí siempre y compartir 5 años de vida juntas.

A mi compañera de tesis que fue a primera persona que conocí cuando entré y desde ese día es muy amiga mía, gracias por siempre estar ahí soportándome y compartiendo en los buenos y malos momentos.

RESUMEN

La presente tesis es un estudio sobre las herramientas existentes para los procesos de Administración de Requisitos y la propuesta de una nueva solución más funcional y aplicable en nuestro país, toda vez que constituye un componente vital en el desarrollo de un producto de software.

El uso de estas herramientas facilita el Proceso de Administración de Requisitos y posibilitan a las empresas desarrolladoras de software elaborar un producto que defina en menor tiempo y con el ahorro de todo tipo de recursos la calidad solicitada por el cliente, por lo que su empleo es insoslayable.

Muchas empresas y compañías desarrolladoras de software son ineficientes en este aspecto, lo que hace que persistan problemas en el desarrollo y elaboración de sus productos informáticos, entre ellos es sistemático la inadecuada comprensión de las necesidades de los usuarios, lo que se traduce en incapacidad de absorber los cambios en los requisitos y las insatisfacciones de los clientes por el inaceptable o bajo desempeño del software.

El objetivo final de este trabajo es brindar una herramienta para el Proceso de Administración de Requisitos en los proyectos involucrados dentro del alcance de mejora y ofrece nuevas funcionalidades agregadas a la herramienta seleccionada para la Administración de Requisitos teniendo en cuenta las necesidades de la Universidad de las Ciencias Informáticas (UCI).

La herramienta, aunque fue elaborada sobre un soporte de software libre, tiene la ventaja de ser multiplataforma y fue desarrollada en lenguaje Java y permitirá prescindir de un importante volumen de archivos en formato duro (impreso) con el consiguiente ahorro de recursos y espacio.

Palabras claves: Requisitos, Administración de Requisitos, Herramienta.

ÍNDICE

ÍNDICE DE TABLA

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En los últimos tiempos se ha impuesto en todas las ramas de la sociedad el empleo de ordenadores para la realización óptima de todos los procesos cotidianos, esto presupuso el diseño de tecnologías y software para su aplicación y a su vez se hizo necesario desarrollar procesos tecnológicos y herramientas para el sostenimiento de estas tecnologías. Uno de los procesos más importantes es el de Administración de Requisitos (REQM).

La correcta Administración de Requisitos en un software constituye la célula fundamental para muchos de los proyectos productivos, las empresas procuran adquirir un software que les facilite elaborar sus productos en menos tiempo y a menor costo.

El auge en la creación de software y los avances en la producción y desarrollo de tecnologías más sofisticadas no han sido suficientes para dar cumplimiento a las exigencias que usuarios y clientes demandan de los desarrolladores, identificándose que una de las causas que provoca que no se cumpla en tiempo con la entrega del producto es la inadecuada Administración de los Requisitos.

La Universidad de las Ciencias Informáticas (UCI), es la mayor industria productora de software del país y constituye un importante eslabón que facilita la informatización de todos los procesos de la economía y los servicios de la sociedad cubana, al tiempo que propicia el avance de la industria del software nacional para su comercialización dentro y fuera de nuestras fronteras.

En sus inicios la UCI no contaba con un Proceso de Desarrollo de Software bien definido, que determinara los pasos a seguir a la hora de desarrollar el mismo. Para ello el Centro de Calidad para Soluciones Tecnológicas (CALISOFT) de la UCI, viene acometiendo un programa de mejora, proponiéndose inicialmente que la universidad alcance en el 2010 la certificación del nivel 2 de Modelo Integrado de Madurez de las Capacidades (CMMI) y la consecuente solidez en los procesos de desarrollo de software legitimando la calidad del producto acabado.

Dentro del nivel 2 del modelo de CMMI se encuentran varias áreas de proceso, la Administración de Requisitos es una de las áreas de proceso sobre la cual se realizó un estudio de las diferentes herramientas aplicables en el mundo para una óptima Administración de Requisitos en los productos

informáticos. En el marco del programa de mejora se optó entonces para apoyar dicho proceso la herramienta Open Source Requirements Management Tool (OSRMT).

Esta es una herramienta de software libre que fue pensada para asistir todo el ciclo de vida del desarrollo del producto. Permite la descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre todos los documentos relacionados con la ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba). [1]

Como propuesta para mejorar el desempeño de dicha herramienta se propone incluir nuevas funcionalidades que hasta el momento se llevan a cabo de forma manual y provocan aglomeración de archivos en formato duro trayendo consigo una innecesaria pérdida de tiempo y otras ineficiencias en el proceso de Administración de Requisitos.

A partir de la problemática anteriormente planteada se puede definir entonces el siguiente **problema científico** de la investigación:

¿Cómo contribuir al perfeccionamiento del Proceso de Administración de Requisitos de manera que disminuya el esfuerzo y tiempo de desarrollo de los proyectos?

Donde el **objeto de estudio** lo constituyen**:** Las Herramientas que permiten realizar el proceso de Administración de Requisitos y su **campo de acción** se centra en El Proceso Administración de Requisitos en la herramienta OSRMT.

Para dar solución al problema planteado se trazó el siguiente **Objetivo General:**

Implementar nuevas funcionalidades a la herramienta OSRMT que perfeccionen la realización de las actividades del proceso de Administración de Requisitos.

La investigación se basa en la siguiente **Idea a Defender:** Con la implementación de las nuevas funcionalidades agregadas a OSRMT, se logrará perfeccionar el proceso de Administración de Requisitos.

Para dar cumplimiento al objetivo planteado se trazaron las siguientes **Tareas de la Investigación:**

- Estudio del estado del arte sobre los temas referentes a la Administración de Requisitos, proceso de mejora y Modelo Integrado de Madurez de las Capacidades.

- Estudio de las herramientas Cliber, GatherSpace, Doors, Requisite Management y OSRMT.

- Estudio de las características y funcionalidades de la herramienta escogida para la Administración de Requisitos.

- Estudio del proceso de Administración de Requisitos definido en la UCI para insertar en el funcionamiento de la herramienta definida para ello.

- Estudio de las distintas herramientas, técnicas y buenas prácticas que existen para modelar el análisis y el diseño.

Para lograr complementar estos resultados se definen los **métodos científicos**, estos se dividen en dos categorías los métodos teóricos y los métodos empíricos, enmarcando específicamente como **métodos teóricos** usados:

- **Analítico–Sintético:** Se realizó un análisis detallado de las aplicaciones y los elementos teóricos que aparecen descritos en las diferentes fuentes bibliográficas extrayéndose las experiencias más adecuadas para su aplicación en las nuevas funcionalidades de la herramienta OSRMT.

Métodos empíricos

- **Entrevista**: Se consultaron especialistas en el tema para adquirir información adicional del proceso de Administración de Requisitos en los proyectos, así como criterios sobre otras experiencias en el empleo de las herramientas existentes para la Administración de Requisitos.

El documento está compuesto por 5 capítulos que recogen el contenido del trabajo, distribuidos de la siguiente manera:

- **Capítulo I:** En este capítulo se realiza la fundamentación teórica del tema donde se exponen algunos conceptos generales acerca del mismo, las tecnologías y tendencias actuales en

la informática que han sido tomadas en consideración, además de explicar las herramientas, las metodologías y los lenguajes empleados en el desarrollo de la aplicación unido al análisis de las soluciones existentes.

- **Capítulo II:** Se muestran las características del sistema y una propuesta del mismo así como su diagrama de casos de uso. Se comenzará con la especificación en detalles del contenido que tendrá el sistema y se realiza además un levantamiento de los requisitos, tanto en lo funcional como en lo no funcional.

- **Capítulo III:** En este capítulo se realiza la construcción de la solución propuesta en el flujo de trabajo de Diseño, predominante en la fase de Elaboración del ciclo de vida del Proceso Unificado Racional (RUP). De los mismos se generan una serie de artefactos, como son los diagramas de clases del análisis, diagrama de secuencia y diagrama de clases del diseño.

- **Capítulo IV**: En este capítulo se muestra cómo los elementos del modelo de diseño se implementan en término de componentes, así como la distribución física del sistema que se muestra a través del diagrama de despliegue. Se exponen los resultados de las pruebas realizadas al software mostrando los datos de las pruebas de caja negra realizadas a la interfaz.

- **Capítulo V:** En este capítulo se realiza la estimación del esfuerzo, es decir, si es factible o no agregar nuevas funcionalidades a la herramienta OSRMT, analizando los costos y beneficios a través del método de estimación Análisis de Punto de Casos de Usos.

Capítulo 1 FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se tratan los temas que responden a la problemática para dar respuesta a las principales interrogantes tratadas en la investigación. Se enuncian los conceptos fundamentales asociados con el proceso de Administración de Requisitos. Al mismo tiempo, se realiza un estudio de la herramienta presentada para gestionar el proceso en la universidad.

1.2 Conceptos básicos

1.2.1 Introducción al modelo CMMI

En la actualidad la universidad está acometiendo un programa de mejoras de sus procesos basado en el modelo de CMMI.

CMMI es un Modelo de Madurez de Mejora de los Procesos para el desarrollo de productos y servicios. Consiste en las mejores prácticas que tratan las actividades de desarrollo y de mantenimiento que cubren el ciclo de vida de un producto, desde su concepción hasta la entrega. [2]

De los siete procesos del nivel 2 del modelo de CMMI, el área de proceso sobre la cual se basa este trabajo es la de la Administración de Requisitos, donde su propósito es gestionar los requisitos de los productos y de los componentes del proyecto e identificar las inconsistencias entre esos requisitos y los planes de trabajo de los proyectos.

1.2.2 Proceso de Administración de Requisitos

La Administración de Requisitos es el proceso de establecer las capacidades básicas de un proyecto. Esto no es más que una manifestación del propósito para una aplicación donde se debe cumplir con objetivos específicos como son investigar y registrar requisitos, paso importante en cualquier proyecto.

La Administración de Requisitos es efectiva para mejorar los flujos de trabajo a través del ciclo de vida del proyecto, desde el diseño de software hasta la estimación de costo y aún creando documentación del usuario. Un rango de técnicas de investigación se debería usar para establecer los requisitos para un

proyecto en particular. Hablar con usuarios prospectivos, conducir investigaciones y examinar sistemas anteriores son todos buenos métodos. Es difícil que requisitos adicionales no sean descubiertos durante el ciclo de vida en un proyecto, y es así que la investigación continuada y documentación de requisitos es esencial para la administración efectiva de requisitos. [3]

1.2.2.1 Criterios para validar requisitos del cliente

Como criterios de aceptación tenemos a los criterios que un producto o componente de producto deben satisfacer para ser aceptado por un usuario, un cliente u otra entidad autorizada. [4]

Los requisitos son analizados para determinar si son necesarios y suficientes para cumplir con los objetivos de niveles más altos de la jerarquía del producto. Los requisitos analizados proporcionan que sean más detallados y precisos para los niveles más bajos de la jerarquía del producto.

Para validar los requisitos del cliente ya están definidos los criterios que determinan si el requisito es seleccionado o no. Estos criterios son:

- ¿El proveedor del Requisitos (REQ) es un proveedor válido?
- ¿El REQ está identificado como único?
- ¿El REQ es modificable?
- ¿El REQ no es ambiguo?
- ¿El REQ está completo?
- ¿El REQ es congruente con otros REQ relacionados?
- ¿El REQ puede ser implementado?
- ¿El resultado de la evaluación de impacto es positivo?
- ¿El REQ está correcto?

- ¿El REQ es traceable?

1.2.2.2 Criterios para validar requisitos del producto

Los requisitos del producto son el refinamiento de los requisitos del cliente en el lenguaje de los desarrolladores.

Para validar los requisitos del producto ya están definidos los criterios que determinan si el requisito es seleccionado o no. Estos criterios son:

- Están identificados los elementos de entrada.
- Están identificados los elementos de salida.
- El requisito es dado por el superior determinado en el organigrama del proyecto.
- El requisito no es ambiguo.
- Es técnicamente factible.
- Puede ser verificado.
- Está correcto.
- El resultado de la evaluación de impacto es positivo.
- El requisito es traceable.

1.2.2.3 Proveedores válidos de requisitos

Definimos como proveedor a:

Una entidad que entrega productos o realiza servicios que serán adquiridos. Un individuo, sociedad, empresa, corporación, asociación u otros servicios que tienen un acuerdo (contrato) con un comprador para el diseño, el desarrollo, la fabricación, el mantenimiento, la modificación o el suministro de elementos bajo los términos de un acuerdo (contrato).[5]

A medida que el proyecto madura y se derivan sus requisitos, todas las actividades o disciplinas estarán encaminadas a satisfacerlos. Por tanto, para evitar el flujo continuo de los mismos se establecen criterios de aceptación que nos dirán qué tanto conoce el proveedor del negocio y se le dará un nivel de importancia que consistirá en una evaluación de 1 a 5 puntos según el nivel de cumplimiento que tenga cada uno de los criterios definidos.

Entre los criterios que se definen para validar un proveedor de requisitos están:

- Conocimiento de mercado.
- Conocimiento de negocio.
- Disponibilidad de tiempo.
- Habilidades de comunicación.
- Interacción con el sistema.
- Grado de autoridad.
- Conocimientos de informática.
- Compromiso con el proyecto.

1.2.2.4 Proveedores válidos de producto y servicio

Se define como proveedor de producto y servicio a terceros que entregan productos o realizan servicios al proyecto; una persona, sociedad, empresa, corporación, asociación u otros servicios que tienen un acuerdo (contrato) con un comprador (el proyecto) para el diseño, el desarrollo, el mantenimiento, la modificación o el suministro de elementos bajo términos de un acuerdo.

Entre los criterios para validar un proveedor de producto y servicio están:

- Manejo de Lenguaje

- Experiencia con Frameworks
- Capacidad de Suministro
- Capacidad de Documentación
- Costo
- Conveniencia de Garantías
- Tipo de Licencia
- Años de experiencia
- Desempeño con nosotros
- Personal disponible
- Ubicación
- Tiempos de entrega
- Tiempos de respuesta

1.2.3 Herramientas para la Administración de Requisitos

El uso de herramientas para la gestión de requisitos mejora la productividad y la calidad en el desarrollo de un proyecto de software. Un inadecuado entendimiento de las necesidades de los usuarios hace que la mayoría de los proyectos fallen, por lo que se hace necesario crear un hilo continuo entre requisitos, diseño e implementación.

Inicialmente se partió de un conjunto de necesidades y limitaciones que fueron observadas por los expertos en la herramienta sobre la cual se venían realizando las tareas de administración de proyectos previos. Básicamente se requería de:

- Una herramienta Open Source (libre) para la gestión de requisitos.

- Que logre llevar el ciclo de vida completo del desarrollo de software.
- Configurable, que pudiese ser adaptada a las necesidades del proyecto.
- Manejo de trazabilidad bidireccional entre los documentos.

1.2.3.1 Herramientas evaluadas

La mayoría de las herramientas de gestión de requisitos que existen en mundo realizan prácticamente los mismos servicios y tienen las mismas limitaciones, una parte de ellas también, descansa sobre un soporte de software propietario.

En el marco del programa de mejora los especialistas evaluaron un grupo de herramientas que proporcionan las necesidades básicas para la Administración de Requisitos tanto en Open Source (Código Abierto) como entre las privativas, teniendo en cuenta en cada caso los beneficios e inconvenientes que presentaban; se tomó como más factible la herramienta OSRMT en su versión original a partir de las posibilidades que brindaba para incorporarle nuevas funcionalidades.

Herramientas Privativas

Caliber

Es una herramienta creada con el objetivo de facilitar la definición, el análisis y la gestión de los requisitos de software, mejorar su calidad y reducir disfuncionalidades. Uno de los aspectos positivos que se ponderó fue su facilidad de uso, que permite un rápido aprendizaje por parte de personal no especializado en informática.

Dentro de la herramienta, los requisitos se pueden agrupar en varios paquetes como, negocio, requisitos de usuario, funcionales, de diseño y escenarios de prueba, entre otros. Esto la hace factible puesto que otras herramientas no permiten la flexibilidad en cuanto a la clasificación de requisitos.

No fue sin embargo seleccionada como la herramienta para gestionar el proceso de Administración de Requisitos en la universidad debido al alto precio de su licencia. Se evaluó no obstante como positivo y distintivo de las demás, que permite crear un glosario de términos específicos que ayuda a que los miembros del proyecto entiendan los requisitos de la misma forma.

GatherSpace

Es una herramienta del entorno la Web capaz de editar, compartir y distribuir requisitos en una red. Esta herramienta es independiente de la plataforma pero necesita de un navegador Web, no precisa ser instalada en los ordenadores para su uso y es 100% compatible con cualquier sistema operativo.

Al ser una herramienta Web el trabajo de la misma puede tornarse lento ya que depende de la disponibilidad del servidor y queda a expensas de fenómenos ajenos al trabajo como la pérdida de la conexión, la sobrecarga del mismo que ralentizará o restringirá el trabajo en el proyecto. Estas limitaciones en cuanto a funcionalidades llevaron a descartarla.

Doors

Esta herramienta forma parte de una familia de soluciones que mejoran la calidad y optimizan la comunicación y la colaboración. Encargada de gestionar los requisitos, enlazar, analizar y manejar un amplio rango de información para asegurar la adecuación del proyecto. Dentro de las funcionalidades que se tuvieron en cuenta a la hora de la selección de la herramienta a utilizar en la universidad esta cumple con varias de ellas, dentro de las que se encuentra importar y exportar información mediante Microsoft Word, Excel, PowerPoint, Outlook, textos planos (ASCII), hojas de cálculo, entre otros.

Su limitación fundamental para la universidad radica en el carácter de herramienta sujeta a un software propietario con sus consiguientes complicaciones.

Herramientas de Código Abierto

Requisite Management (REM)

Es una herramienta experimental gratuita desarrollada por la Universidad de Sevilla, pero a pesar de ser una herramienta de código libre solamente funciona sobre el sistema operativo de Windows lo que hace de esto un inconveniente puesto que se pretende que la universidad, migrando completamente al software libre logre alcanzar la independencia tecnológica.

Esta herramienta tampoco permite la creación de gráficos ni diagramas y solamente ofrece la posibilidad de crear documentos.

Open Source Requirement Management Tool (OSRMT)

Es una herramienta libre que permite una descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre los documentos relacionados con la ingeniería de requisitos. Dispone de un entorno multiusuario. Ante cualquier duda o problema con alguna funcionalidad de la herramienta se dispone de una ayuda en línea y un manual de usuario. Dentro de las herramientas libres que garantizan la ingeniería de requisitos es sin duda una de las que mejores funcionalidades ofrece.

1.2.3.2. Matriz de selección

	OSRMT	REM	Doors	Caliber	GatherSpace
Manejo de varios proyectos	Sí	-	Sí	Sí	Si
Open Source	Sí	No	No	No	No
Manejo de trazabilidad	Sí	Sí	Sí	Sí	Sí
Generación de reportes	Sí	Sí	Sí	Sí	Sí
Documentación	Sí	-	No	Sí	Sí
Soporte	No	No	Sí	Sí	Sí
Incluye interfaz web	Sí	No	No	Sí	Sí
Control de versiones	Sí	Sí	Sí	Sí	-
Manejador de base de datos	Sí	Sí	Sí	Sí	Sí
Plataforma	Independiente	Windows	Independiente	Windows	Independiente
Niveles de seguridad	Configurables	Configurables	-	-	Predeterminados
Control de cambios	Sí	No	-	-	Sí

Tabla 1 Matriz de selección.

De acuerdo con los datos expuestos en la tabla anterior, ninguna de las herramientas analizadas cubre las funcionalidades que se desean incorporar al proceso de Administración de Requisitos, por lo que la herramienta que se propuso por los especialistas como base para sobre ella incorporar las nuevas funcionalidades necesarias al proceso resultó OSRMT a la que las autoras incorporaron, como se expone en este trabajo, los requisitos para su empleo óptimo.

OSRMT es una de las herramientas de código abierto más moldeable a las necesidades particulares de distintos usuarios, aun los más exigentes, puesto que posee las características y funcionalidades básicas de cualquiera de las herramientas de Administración de Requisitos, pero al propio tiempo permite el monitoreo de varios proyectos, maneja la trazabilidad entre los requisitos y permite la generación de reportes, además consta con niveles de seguridad configurables.

Es una herramienta multiplataforma, desarrollada en Java y la misma cumple en términos generales con las características que en principio fueron tomadas como necesarias.

1.3 Herramientas de desarrollo

1.3.1 Entorno Integrado de Desarrollo

El Ambiente de Desarrollo Integrado que se utilizó en la elaboración de OSRMT es Eclipse, el mismo incorpora un conjunto de plugins para trabajar con parte de las tecnologías usadas en el desarrollo de la aplicación.

1.3.1.1 Eclipse

Entorno de desarrollo libre para crear aplicaciones de cualquier tipo. Es un proyecto de desarrollo de software de código abierto dedicado a proporcionar una plataforma industrial robusta, con amplias características y con calidad comercial para el desarrollo de herramientas altamente integradas.

Se utilizó Eclipse como entorno de desarrollo debido a que dentro de sus funcionalidades permite usar y añadir plugins que proveen nuevas utilidades al programa. Utilizar Eclipse para programar en Java facilitó ahorrar gran cantidad de tiempo y esfuerzo en el desarrollo del sistema ya que dentro de sus funcionalidades brinda la muestra de los errores de compilación en tiempo real y la compilación automática de los archivos salvados, evitando el tedioso y lento proceso de compilar, observar y corregir los errores.

1.3.1.2 NetBeans

NetBeans es una plataforma para el desarrollo de aplicaciones de escritorio usando Java. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de

software llamados módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. [6

Las nuevas funcionalidades agregadas a OSRMT están desarrollas en NetBeans, debido a la complejidad que presentaba el código de la herramienta original, para ello se desarrolló este módulo independiente que finalmente queda integrado a la versión original.

1.4 Lenguaje de programación

1.4.1 Java

Java es un lenguaje de alto nivel, orientado a objetos, multiplataforma que permite la ejecución de un mismo programa en múltiples sistemas operativos. Es distribuido ya que contiene un conjunto de clases que permiten la comunicación a través de la red facilitando así la creación de aplicaciones distribuidas. Fue diseñado para crear software altamente fiable. Tiene una plataforma, que a diferencia de las demás son una combinación de hardware y software, está basada en software, que corre sobre cualquier hardware. Consta de 2 componentes: la máquina virtual de Java (JVM) y la interfaz de programación de aplicaciones (API). Este lenguaje nos permitió implementar el software en una plataforma y así ejecutarlo prácticamente en cualquier otra y combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados. El ser libre nos proporcionó una mayor flexibilidad, ahorro en el costo de desarrollo, independencia y compatibilidad con otros sistemas.

1.5 Gestor de base de datos

1.5.1 PostgreSQL

PostgreSQL es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Ofrece control de concurrencia multiversión, soportando casi todas las sintaxis SQL (incluyendo subconsultas, transacciones, y tipos de funciones definidas por el usuario).

Es altamente escalable, tanto en la gran cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede acomodar. [7]

A ello se agrega que su código fuente está disponible bajo los más liberales términos de licencia de código abierto, lo que permite hacer uso de sus funcionalidades y desarrollo del producto sin incurrir en gasto alguno. Debido a sus características y técnicas sobresalientes, se utilizó para almacenar toda la información del sistema ya que es fiable y presenta gran capacidad de almacenamiento de datos.

1.6 Herramientas y metodologías de modelado visual

1.6.1 Metodología de desarrollo

Proceso Unificado de Desarrollo de Software (RUP)

El Proceso Unificado de Desarrollo de Software tiene un carácter que está regido por los casos de uso, centrado en la arquitectura, iterativo e incremental. El proceso utiliza el Lenguaje Unificado de Modelado (UML), un lenguaje que produce dibujos comparables en sus objetivos a los que se utilizan desde hace mucho tiempo en otras disciplinas técnicas. El proceso pone en práctica el basar gran parte del proyecto de desarrollo en componentes reutilizables, es decir, en piezas de software con una interfaz bien definidas para su posterior utilización en versiones distintas si son necesarias. [8]

Dentro de las metodologías existentes RUP fue la escogida ya que es una metodología completa y documentada hasta el detalle, que permite avanzar en el desarrollo del software sin contratiempos adicionales, ya que es muy expedita manejo.

Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software, permite la modelación de sistemas con tecnología Orientado a Objetos. En UML podemos encontrar elementos (abstracciones que constituyen los bloques básicos de construcción), relaciones (Ligan los elementos) y diagramas (Es la representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas). Esto beneficia a que a lo largo de todo el proceso se creen modelos del sistema a desarrollar.

1.6.2 Herramienta de modelado visual

Visual Paradigm

Es una herramienta que se utiliza para modelar mediante UML profesional, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue.

Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación minimizando el tiempo invertido y aumentando la eficiencia al modelar. El mismo puede ejecutarse de diferentes formas, puede ser instalado, o portable, es multiplataforma y posee varios ambientes, es capaz de generar código para Java y NET, de manera que proyectos escritos en estos lenguajes puedan desarrollarse fácilmente o hacer la ingeniería inversa a partir del código fuente generar diagramas.[9]

1.7 Conclusiones

En este capítulo se exponen los principales conceptos empleados en la investigación y se explica el análisis previo realizado de las herramientas existentes para llevar a cabo la Administración de Requisitos así como del proceso en general.

Se realiza también un estudio comparativo sobre las posibilidades y funcionalidades del grupo seleccionado de herramientas para demostrar sobre que cualidades se hizo énfasis para la mejor selección de la herramienta base, la metodología de trabajo, el lenguaje a emplear y el gestor de base de datos en el desarrollo de nuevas funcionalidades para OSRMT.

Capítulo 2 CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se abordan las características y organización de los procesos del sistema a informatizar, así como la propuesta de cómo quedaría el mismo una vez incorporadas las nuevas funcionalidades que serán agregadas a la herramienta.

Se realiza el modelo de dominio y el levantamiento de los requisitos funcionales y no funcionales para demostrar las nuevas posibilidades que tendrá el sistema.

2.2 Descripción de los procesos del negocio propuestos

La UCI desarrolla disímiles proyectos productivos, donde la gestión de requisitos, en su mayoría, no cumple con las especificaciones necesarias al no tener definida una herramienta que permita gestionar todos estos procesos se presentan problemas en el tiempo de entrega de los proyectos.

Para lograr un mejor intercambio entre desarrolladores y clientes, y de paso crear un estándar en el proceso de Administración de Requisitos en la universidad se decidió agregar nuevas funcionalidades a la herramienta OSRMT.

El sistema desarrollado es un módulo incorporado a la herramienta OSRMT que incorpora funcionalidades que otras herramientas de gestión de requisitos no realizan y estará destinada a administrar los requisitos en los proyectos de la UCI.

2.2.2 Objeto de informatización

Fueron objeto de informatización en la herramienta los siguientes procesos:

- Validación: se validaron las nuevas funcionalidades del paquete de mejoras: requisitos del cliente, requisitos del producto, proveedores de requisitos y proveedores de productos y servicios.
- Gestión de Inconsistencias: se agregarán, modificarán y eliminarán inconsistencias encontradas entre los requisitos, planes del proyecto y productos de trabajo.

- Evaluación de la complejidad y prioridad de casos de uso: se evaluará la complejidad y la prioridad de los casos de uso.

2.3 Propuesta de sistema

Después de identificar las principales funcionalidades a implementar, la propuesta del sistema se obtuvo una herramienta que permitirá administrar los requisitos en un software, multiplataforma y desarrollada en Java. Brindará al usuario además de las funcionalidades básicas que se realizan en la herramienta original, otras nuevas que han sido identificadas como importantes y que fueron agregadas al proceso.

Dentro de estas nuevas funcionalidades la validación de requisitos del producto y del cliente se realizará en la versión original de OSRMT, así como la evaluación de casos de uso. En el módulo extendido se realizarán las funcionalidades de gestión de proveedores de requisitos, proveedores de producto y servicio y la validación de los mismos.

Al finalizar cada sesión de trabajo posibilitará la generación de reportes sobre el estado de los requisitos y proveedores en formato PDF.

2.4 Modelo de dominio

Se propone un modelo del dominio, ya que en el sistema los procesos del negocio no están bien definidos, no son visibles y las fronteras no están bien establecidas. Además, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los usuarios, clientes, desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se desarrolla el sistema. Este modelo va a contribuir a identificar algunas clases que se utilizarán en el sistema.

2.4.1 Diagrama de clases del dominio

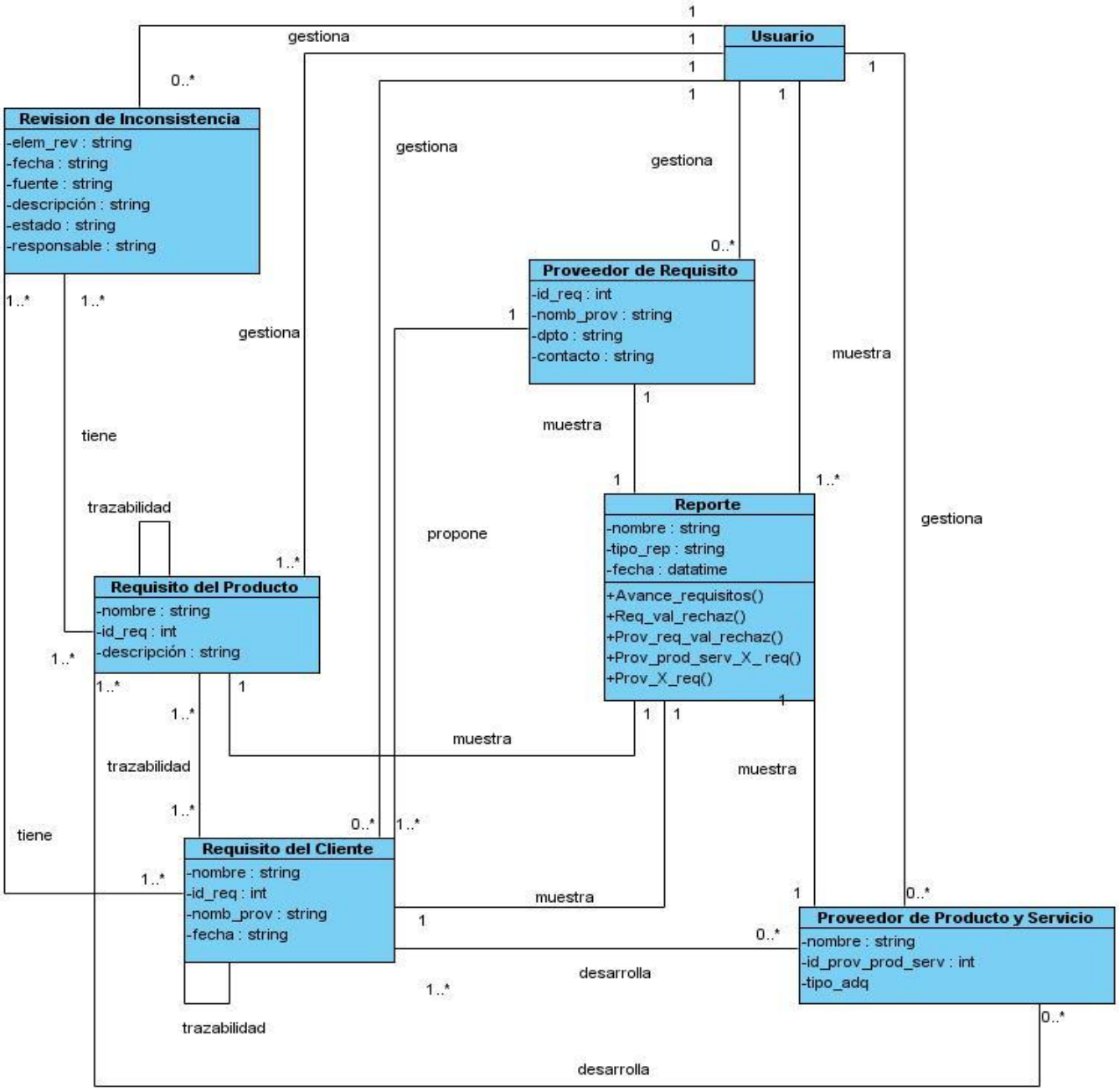

Figura 1 Diagrama de clases del dominio

2.4.2 Descripción de los objetos del dominio

Usuario: Representa la persona que interactúa con la herramienta.

Reporte: Muestra el estado de los requisitos del cliente y del producto, proveedores de requisitos y proveedores de producto y servicio.

Requisitos del producto: Refinamiento de los requisitos del cliente en el lenguaje de los desarrolladores.

Requisitos del cliente: Resultado de consolidar y resolver los conflictos entre las necesidades, expectativas, limitaciones y las interfaces de las partes interesadas en el producto de manera que sea adaptable para el cliente.

Proveedores de requisitos: Representan personas que proveen de requisitos a algún miembro del proyecto, (para los requisitos del cliente el proveedor es el cliente, para los requisitos del producto es algún miembro del proyecto autorizado para ello).

Proveedores de productos y servicios: Terceros que entregan productos o realizan servicios al proyecto; una persona, sociedad, empresa, corporación, asociación u otros servicios que tienen un acuerdo (contrato) con un comprador (el proyecto) para el diseño, el desarrollo, el mantenimiento, la modificación o el suministro de elementos bajo términos de un acuerdo.

Revisión de Inconsistencias: Inconsistencias encontradas entre los requisitos, planes del proyecto y productos de trabajo.

2.5 Relación de los requisitos

El propósito general de la captura de requisitos es tener una descripción de lo que el sistema debe hacer y delimitar su alcance, es decir, qué debe y qué no debe hacer.

Los requisitos son las condiciones o capacidades que tienen que ser alcanzadas por un sistema para satisfacer al cliente o usuario final. Se clasifican en dos tipos: funcionales y no funcionales.

2.5.1 Listado de requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir.

R1. Gestionar requisitos del cliente

- 1.1 Insertar requisitos del cliente
- 1.2 Modificar requisitos del cliente
- 1.3 Eliminar requisitos del cliente

R2. Validar requisitos del cliente

R3. Mostrar requisitos del cliente aprobados

R4. Gestionar proveedores de requisitos

- 4.1 Insertar proveedores de requisitos
- 4.2 Modificar proveedores de requisitos
- 4.3 Eliminar proveedores de requisitos

R5. Validar proveedores de requisitos

R6. Mostrar proveedores de requisitos validados

R7. Gestionar requisitos del producto

- 7.1 Insertar requisitos del producto
- 7.2 Modificar requisitos del producto
- 7.3 Eliminar requisitos del producto

R8. Validar requisitos del producto

R9. Mostrar requisitos del producto aprobados

R10. Evaluar complejidad de CU

R11. Evaluar prioridad de CU

R12. Importar requisitos del producto desde un fichero XMI

R13. Gestionar Inconsistencias

- 13.1 Insertar inconsistencia
- 13.2 Modificar inconsistencia
- 13.3 Eliminar inconsistencia

R14. Mostrar reportes

- 14.1 Avance de los requisitos
- 14.2 Requisitos validados y rechazados
- 14.3 Proveedores de requisitos validados y rechazados
- 14.4 Proveedores de producto y servicio por requisitos
- 14.5 Proveedores por requisitos

R15. Gestionar proveedores de productos y servicios

- 15.1 Insertar proveedores de productos y servicios
- 15.2 Modificar proveedores de productos y servicios
- 15.3 Eliminar proveedores de productos y servicios

R16. Validar proveedores de productos y servicios

R17. Mostrar proveedores de producto y servicio validados

2.5.2 Definición de los requerimientos no funcionales

Los requisitos no funcionales son características o propiedades que el sistema debe tener.

Apariencia o interfaz externa

- La interfaz debe ser agradable para el usuario, combinando los colores grises de las interfaces del OSRMT con las del módulo Extend. El texto de las funcionalidades a mostrar debe estar en Arial tamaño 12 y las mismas no deben contener imágenes que demoren la respuesta al usuario.

Usabilidad

- Los usuarios deben tener un conocimiento básico del Proceso de Administración de Requisitos para poder trabajar con la herramienta.

- El sistema debe permitir el acceso concurrente de diferentes usuarios.

Seguridad

- El acceso a cualquier manipulación del sistema, tanto como la entrada o análisis de los datos estará sometido a un proceso de autenticación.

Software

- La computadora que haga función de servidor, independientemente del sistema operativo que tenga, Windows o Linux necesita como gestor de base de datos PostgreSQL.

- Las computadoras de los usuarios solo requieren de la máquina virtual de Java JDK 5 o una superior.

Hardware

- Los requerimientos mínimos del hardware son: Una PC cliente con 128 MB y un servidor de base dato de 256 MB, velocidad mínima de 1 MHz para cliente y servidor y la capacidad en disco duro varía en dependencia de los documentos que se deseen guardar.

2.6 Modelo de Caso de Uso del Sistema

El modelo de casos de uso del sistema comprende actores del sistema, casos de uso que estos inicializan, el diagrama de casos de uso del sistema y una descripción detallada de estos casos de uso.

2.6.1 Definición de los actores del sistema

Actores del sistema	Justificación
Usuario	Representa a la persona que interactúa con la herramienta.

2.6.2 Diagrama de Casos de Uso del Sistema

Los casos de uso son artefactos narrativos que describen el comportamiento del sistema desde el punto de vista del usuario. Establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibles requisitos que debe cumplir el sistema.

Casos de uso del sistema identificados teniendo en cuenta los requisitos funcionales.

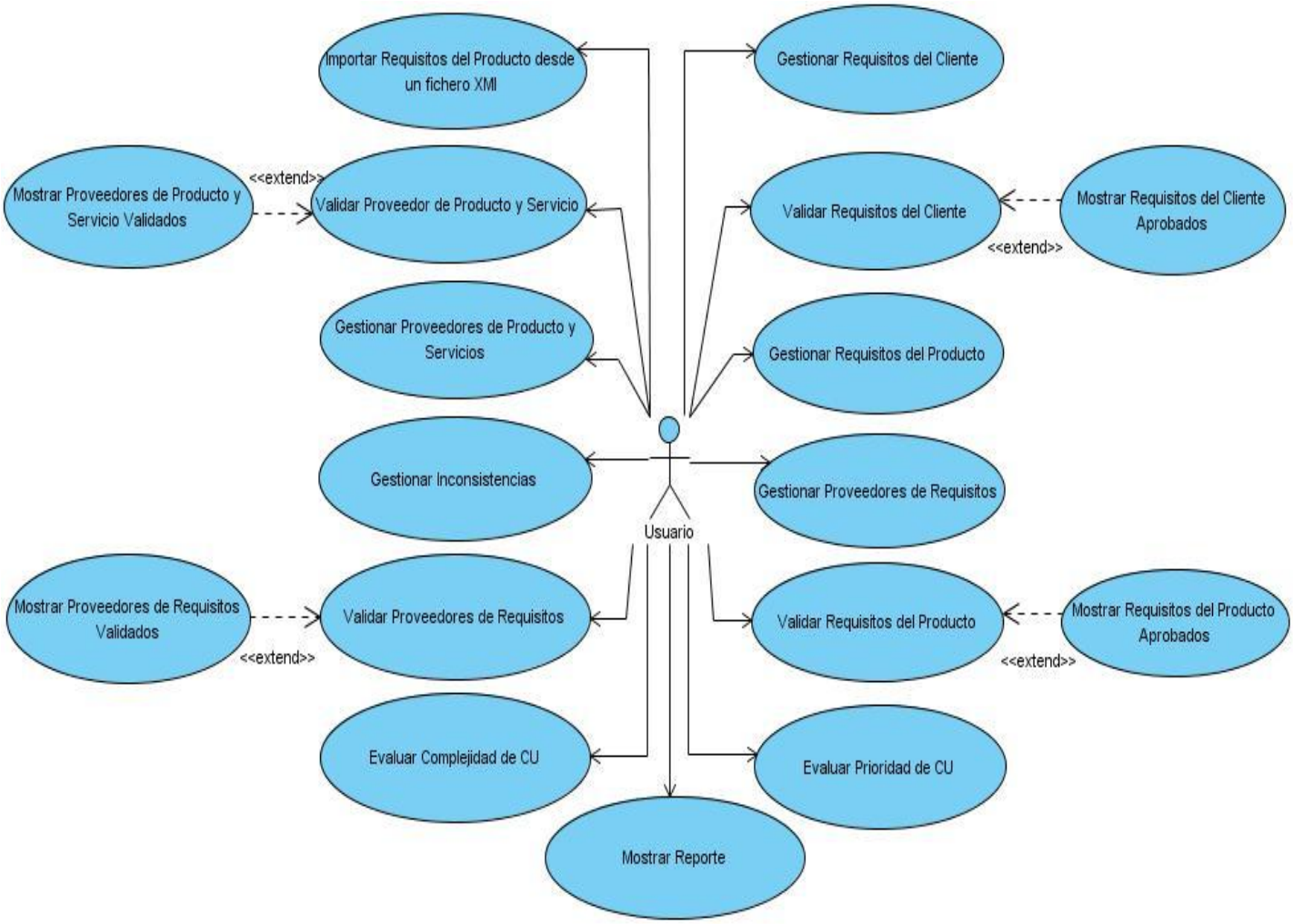

Figura 2 Diagrama de Casos de Uso del Sistema

2.6.3 Descripción textual de los Casos de Uso

2.7 Conclusiones

En este capítulo se expuso la propuesta del sistema para cada una de las funcionalidades que serán agregadas a la herramienta se presentó el modelo de dominio así como los requisitos funcionales y no funcionales. Se realizó el diagrama de casos de uso del sistema y la descripción detallada de cada uno de estos.

Capítulo 3 ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se presentan los diagramas de clases del análisis y del diseño, así como los diagramas de interacción y el modelo lógico y físico de datos. También se describe la arquitectura utilizada y los patrones empleados.

3.2 Modelo de Análisis

El modelo de análisis servirá para lograr una mayor comprensión del dominio del problema descrito en el capítulo anterior por un Modelo de Dominio que se elaboró. También será indispensable a su vez como punto de partida para realizar el diseño del sistema.

En el modelo de análisis no se toman en cuenta el lenguaje de programación que se va a utilizar en la construcción ni otros aspectos como la plataforma o los componentes reutilizables de otros sistemas, pues su objetivo principal es comprender, preparar, modificar y en general mantener exactamente los requisitos del software y no precisar cómo se realizará su implementación. Se considera que este modelo es la primera aproximación al Modelo de Diseño y se describe en un lenguaje comprensible para los desarrolladores.

3.2.1 Diagramas de clases del análisis

El diagrama de clases del análisis (DCA) es un artefacto en el que se representan los conceptos del dominio de un problema y que representa las cosas del mundo real, no de la implementación automatizada. Está compuesto por clases del análisis y sus relaciones. Las clases del análisis están centradas en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos importantes y relaciones del dominio.

Estas clases se clasifican en:

- Clases de interfaz: Se utilizan para modelar la interacción entre el sistema y sus actores.

- Clases de Control: Representan coordinación, secuencia, transacciones, y control de otros objetos.

- Clases de Entidad: Se emplean para modelar información que posee una larga vida y que es a menudo persistente.

3.3 Modelo del Diseño

Una entrada esencial en el diseño es el resultado del análisis, o sea, el modelo de análisis que proporciona una comprensión detallada de los requisitos. En el diseño se modela el sistema y se encuentra su forma, incluyendo la arquitectura, para que soporte todos los requisitos funcionales, no funcionales y las restricciones que se le suponen. Es específico para una implementación y depende del lenguaje de programación.

3.3.1 Diagramas de Secuencia

Los diagramas de secuencia destacan el orden temporal de los mensajes, o sea, la forma en que las operaciones se producen en el tiempo. Se crean durante la elaboración, cuando es útil identificar los detalles de los eventos del sistema para poner en claro cuáles son las operaciones que se deben diseñar para que el sistema gestione, escribir los contratos de las operaciones del sistema y posiblemente, dar soporte a la estimación. Para modelar los aspectos dinámicos del sistema se utilizaron los diagramas de secuencia por cada escenario de caso de uso.

3.3.2 Diagrama de clases del diseño

El diagrama de Clases de diseño (DCD) describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones que existen entre ellos. Además, muestra los atributos y operaciones de una clase y las restricciones a que se ven sujetos, según la forma en que se conecten los objetos. Un diagrama de clases está compuesto por los siguientes elementos: Clase (Atributos, métodos y visibilidad) y Relaciones (Herencia, Agregación, Asociación y Composición).

3.4 Modelo de Datos

3.4.1 Modelo Lógico de Datos

Este modelo contiene el diagrama de clases persistentes que serán las tablas que compondrán la base de datos del sistema. A partir de este modelo se genera el modelo físico que es el que representa las relaciones de las tablas de la base de datos después de haber pasado por un proceso de mapeo donde las relaciones se transforman según su naturaleza y pueden o no incrementar el número de tablas de la base de datos.

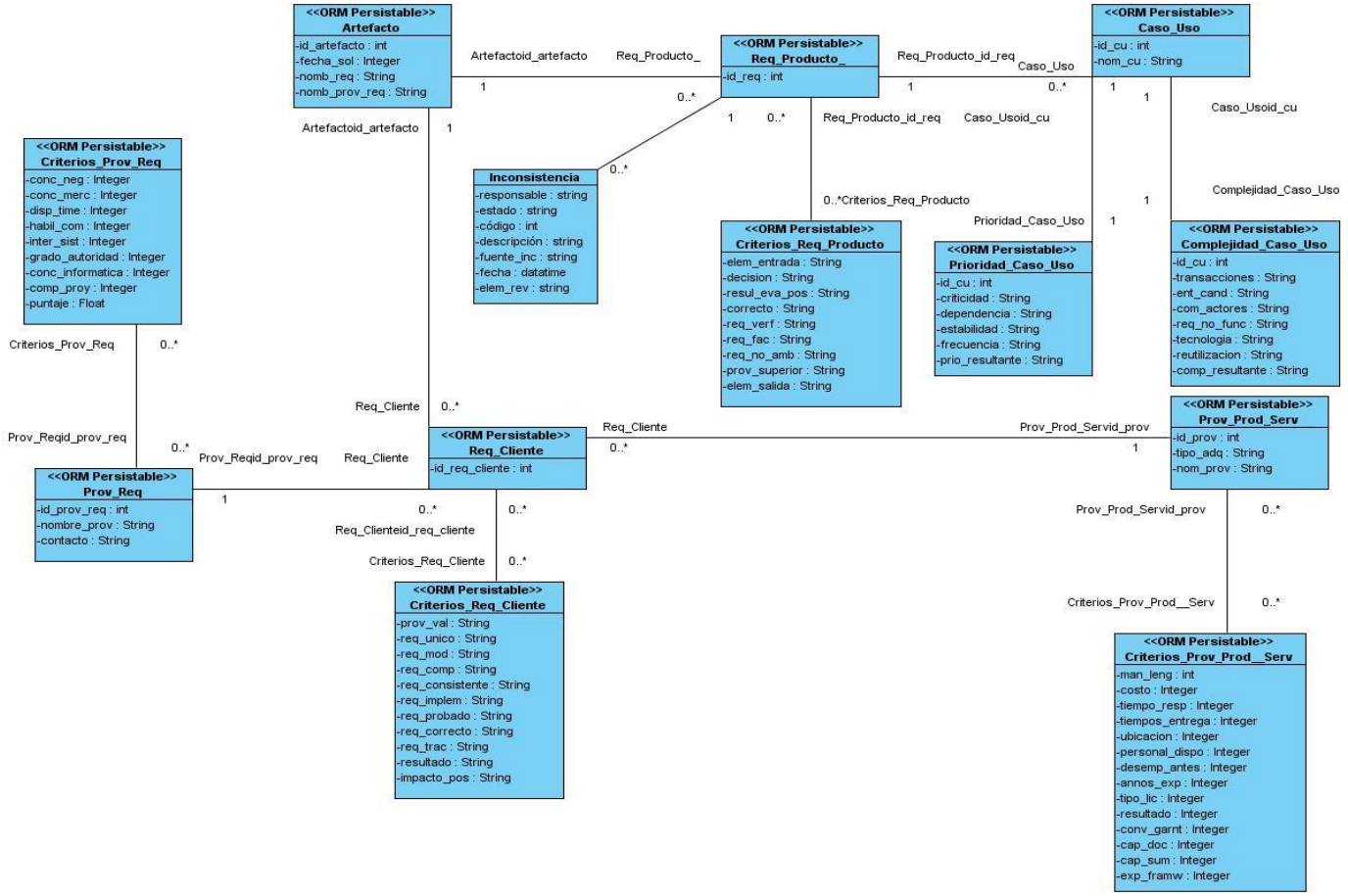

Figura 3 Modelo Lógico de Datos

3.4.2 Modelo Físico de Datos

Son las tablas que contienen la base de datos del sistema con sus atributos y relaciones entre las tablas y es donde se identifica el atributo identificador de la tabla. Este modelo de datos se realizó a partir del diagrama de clases persistentes, las cuales fueron identificadas en el diagrama de clases del diseño.

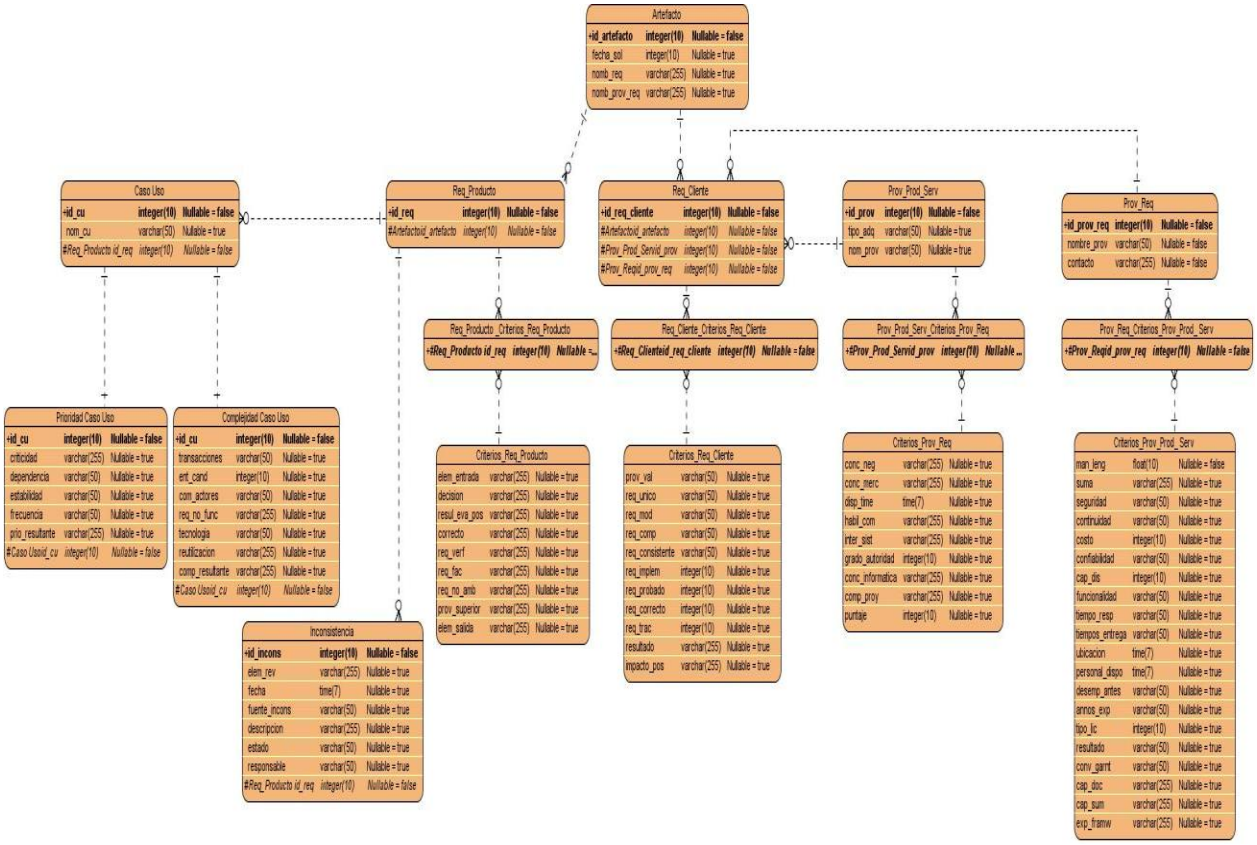

Figura 3 Modelo Físico de Datos.

3.5 Arquitectura utilizada

Arquitectura en capas:

Es un estilo de programación donde su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos. Su ventaja principal es que el desarrollo se puede llevar a cabo en varios niveles, en caso de que sobrevenga algún cambio este no afecte la capa superior.

Capa de presentación: Esta capa donde se le presenta el sistema al usuario, comunicándole y capturando la información del usuario en un mínimo proceso. Esta capa se comunica únicamente con la capa de negocio. Es conocida también como interfaz gráfica, siendo amigable al usuario donde se presentan los formularios con los que interactúa el usuario. Todas las clases referentes a esta capa en el sistema están agrupadas en un paquete nombrado Vistas.

Capa de Negocio: Aquí es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso y se establecen las reglas que debe cumplir el sistema. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. En la aplicación todas estas clases están contenidas en el paquete Modelo donde se implementa toda la lógica de la capa de Negocio.

Capa de datos: Es donde residen los datos y es la encargada de acceder a los mismos. Este paquete recibe el nombre DAO y está formulado por el gestor de base de datos PostgreSQL que es el encargado de realizar las consultas que garanticen todo el almacenamiento o recuperación de la información desde la capa de Negocio.

3.6 Patrones de diseño

Se usa un patrón de diseño con el objetivo de facilitar la reusabilidad de la arquitectura de un software y que el sistema sea de gran fortaleza, para proporcionar una interfaz unificada. El objetivo que se persigue es reducir la complejidad y minimizar las dependencias.

3.6.1 Fachada (Facade):

Patrón estructural que proporciona una interfaz unificada que es la encargada de crear instancias de las demás interfaces que ella utiliza, haciéndolo más fácil de usar. Este patrón permite reducir la complejidad y minimizar las dependencias y es muy intuitivo si se quiere proporcionar una interfaz sencilla para un sistema complejo.

La comunicación de los clientes con el subsistema es a través de la fachada, que reenvía las peticiones a los objetos del subsistema apropiados y puede realizar también algún trabajo de traducción. Los clientes que usan la fachada no necesitan acceder directamente a los objetos del sistema.

3.6.2 Patrones GRASP (General Responsibility Assignment Software Patterns)

Los patrones GRASP permiten describir los principios fundamentales de asignación de responsabilidades a objetos. Estos patrones son esenciales en el diseño eficaz de un software. Por las características de estos patrones, se consideró necesaria su utilización en este trabajo para obtener un producto de mayor calidad.

Experto

Este patrón de diseño define asignar la responsabilidad a la clase que cuente con la información necesaria para manejar esta responsabilidad. Esto proporciona que cada clase lleve a cabo la asignación de responsabilidades de forma adecuada, lo que permite al sistema la reusabilidad de sus componentes.

Alta Cohesión

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un gran trabajo. Esto permite tener identificada la labor que cada elemento del diseño debe realizar dentro del sistema, evitando la sobrecarga de funcionalidades en una clase.

Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Un alto acoplamiento tendría como consecuencia que las clases sean más difíciles de entender cuando estén aisladas, más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen. Por lo que un bajo acoplamiento plantea que debe haber pocas dependencias entre las clases, o sea, hace que en el sistema cada clase cumpla con las funciones que tiene definidas.

3.7 Conclusiones

En este capítulo se exponen los principales artefactos del análisis y diseño del sistema empleado en la solución del problema, específicamente los diagramas de clases del análisis y diseño y los diagramas de interacción. También se describieron los patrones y arquitectura utilizada en el sistema y se justifica su uso en cada caso por encima de otros existentes.

Capítulo 4 IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En este capítulo se lleva a cabo la descripción de las principales tareas del flujo de implementación, así como cada uno de los artefactos generados. Se realizan las pruebas al sistema.

4.2 Diagrama de despliegue

El modelo de despliegue representa la ubicación física que tendrá el sistema cuando se vaya a desplegar. Teniendo en cuenta que la solución propuesta está compuesta por las terminales de las áreas de trabajo, un servidor de base dato y un dispositivo de impresora, a continuación se presenta cada una de las funciones de las mismas:

Desde el nodo PC Cliente, se puede acceder al servidor de base de datos y al dispositivo de impresión. El nodo servidor de base de datos, utilizará el sistema Gestor de base de datos PostgreSQL y utilizará para la conexión al servidor de base de dato JDBC. El nodo Impresora utilizará un dispositivo de impresión al cual se podrá acceder mediante USB.

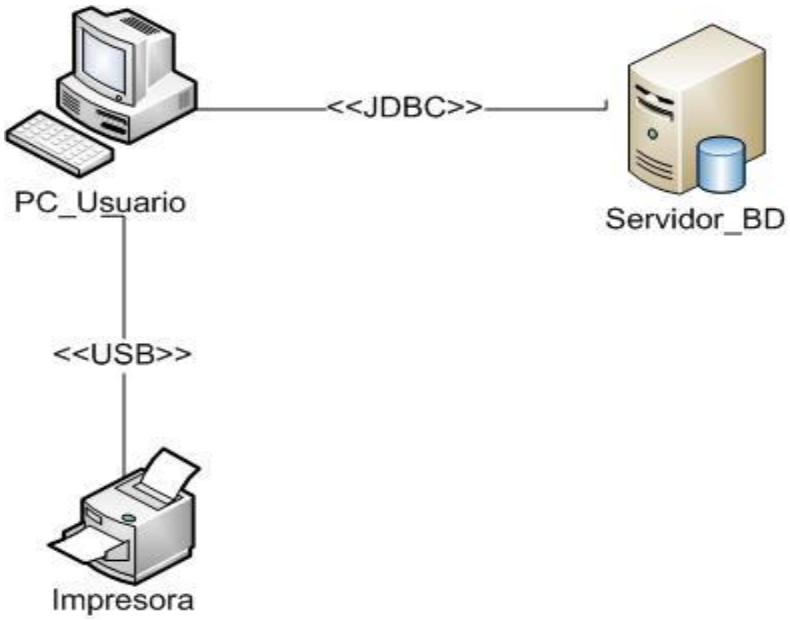

Figura 4 Diagrama de despliegue.

4.3 Diagrama de componentes

Los diagramas de componentes se utilizan para dar una visión estática de la implementación del sistema, mostrando los componentes del software (código fuente, ejecutables, etc.) y sus relaciones dentro del sistema. Estos diagramas pueden contener paquetes para agrupar los elementos del modelo.

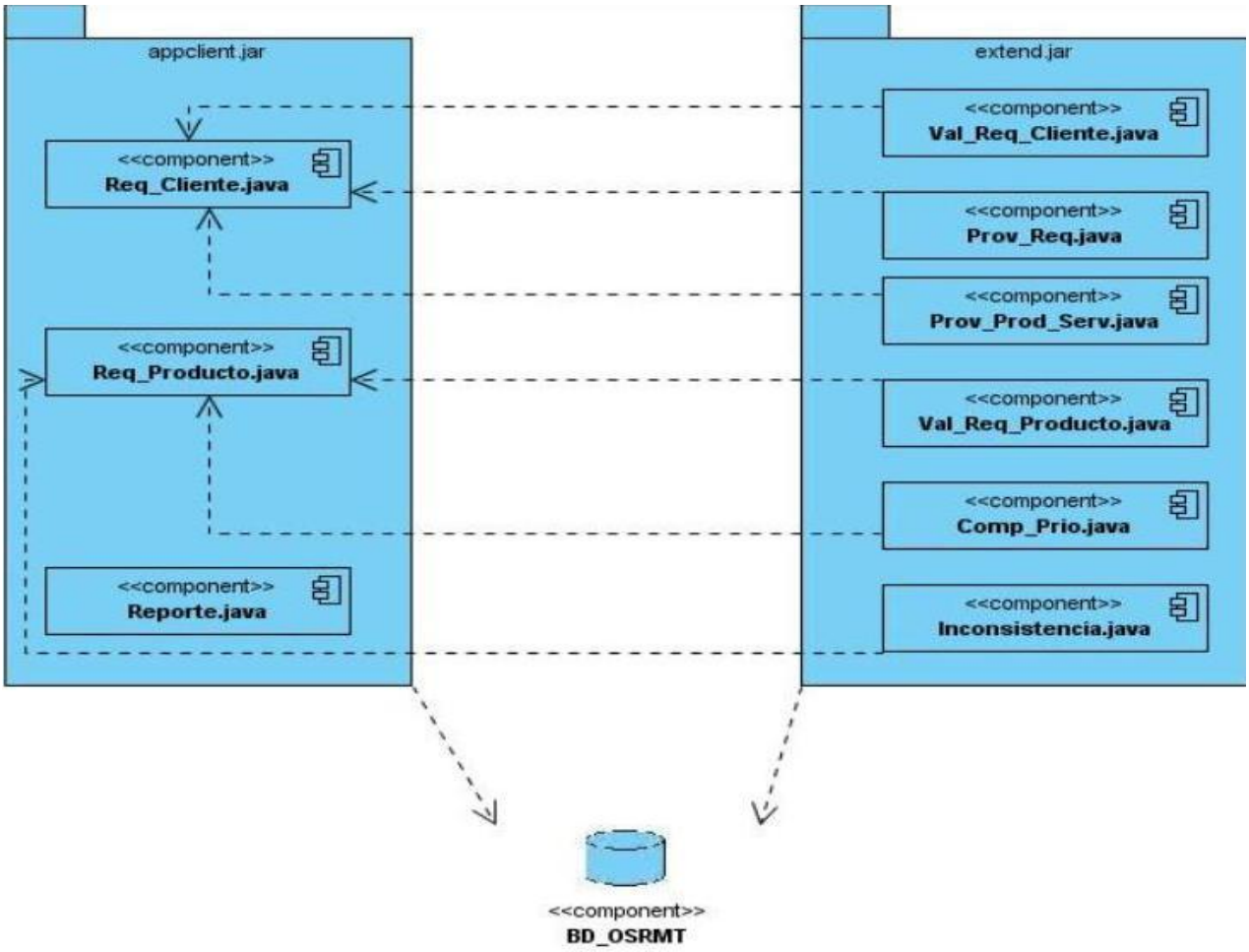

Figura 5 Diagrama de componentes.

4.4 Pruebas de Software

Desarrollar un software implica una serie de actividades de producción la cuales son realizadas por personas y éstas cometen errores y cambian de ideas. Los errores pueden empezar a darse desde el

primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea e imperfecta; así en los posteriores pasos del diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad.

Las pruebas constituyen la actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es realizada de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. (Jacobson, y otros, 2000)

Durante el proceso de implementación se fue realizando paralelamente la gestión de la calidad del producto mediante el uso y técnicas de pruebas de software. Las técnicas de prueba ayudan a definir conjuntos de casos de prueba aplicando un cierto criterio los casos de pruebas quedarán determinados por los valores a asignar en las entradas en su ejecución.

4.5 Pruebas de Caja Negra

Las pruebas de caja negra se llevan a cabo en la interfaz de la aplicación. Lo que se pretende probar con este método es que las funcionalidades del software son operativas, que los datos insertados son correctos y que se obtiene la respuesta esperada.

La realización de esta prueba se basa en los requisitos funcionales y los errores que principalmente encuentra son de interfaz, de rendimiento, de estructuras de datos y de funciones incorrectas o ausentes.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de estos casos y permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea hallar es un error o probar una funcionalidad. Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- **Técnica de la Partición de Equivalencia:** Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

- **Técnica del Análisis de Valores Límites:** Esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

- **Técnica de Grafos de Causa-Efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Una de las pruebas a desarrollar, utilizando la técnica de caja negra, es la del Sistema, que es realizada sobre la interfaz. Esta prueba se realiza luego de integrar todos los módulos que conforman el sistema para comprobar las especificaciones obtenidas durante el análisis y diseño contra el funcionamiento del sistema final. La correcta realización de estas pruebas garantiza detectar varios errores.

4.6 Diseño de Casos de Prueba

Con el diseño de casos de pruebas se obtienen un conjunto de pruebas cuya misión es encontrar defectos y errores en el sistema. Cada caso de prueba brinda un número de valores de entradas en las pruebas, condiciones de ejecución y resultados esperados de las mismas para verificar una determinada funcionalidad del sistema.

Se diseñaron casos de pruebas por cada una de las funcionalidades del sistema.

(Ver Anexo VI)

4.7 Conclusiones

En este capítulo se finaliza la etapa de implementación, donde se realizó la modelación del diagrama de despliegue y componente del sistema. Se muestran las pruebas de caja negra realizadas a la interfaz del software.

Capítulo 5 FACTIBILIDAD DEL SISTEMA

5.1 Introducción

En la planificación del proceso de desarrollo de software es de vital importancia la estimación, la cual consiste en determinar con cierto grado de certeza los recursos necesarios para el desarrollo del mismo, ya sean recursos de hardware, software, esfuerzo, tiempo y costo. En el presente capítulo se realizará un estudio de factibilidad para la realización del sistema propuesto mediante una estimación de tamaño, esfuerzo y planificación necesaria para llevar a cabo el mismo.

5.1.1 Cálculo de Puntos de Casos de Usos sin ajustar

El cálculo de los Puntos de Caso de Uso sin ajustar constituye el primer paso y se calcula a partir de la siguiente ecuación:

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

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

Para obtener el Factor de Peso de los Actores sin ajustar se debe tener en cuenta la cantidad de actores presentes en el sistema y la complejidad de los mismos. Los criterios se muestran en la siguiente tabla:

Tipo de actor	Descripción	Peso	Cantidad * Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación.	1	0*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	0*2

Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	1*3
Total			3

Tabla 2 Factor de Peso de los Actores sin ajustar

Para obtener el valor del Factor de Peso de los Casos de Uso sin ajustar se debe tener en cuenta la cantidad de casos de uso y la complejidad de cada uno de ellos. Los criterios se muestran en la siguiente tabla:

Tipo de Caso de Uso	Descripción	Peso	Cantidad * Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones.	5	12*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones.	10	5*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	0*15
Total			110

Tabla 3 Factor de Peso de los Casos de Uso sin ajustar

Por lo que los Puntos de Caso de Uso sin ajustar resultan:

UUCP= UAW + UUCW = 3+110 = 113

5.1.2 Cálculo de Puntos de Casos de Uso ajustados

Una vez obtenidos los Puntos de Caso de Uso sin ajustar se procede al cálculo de los Puntos de Casos de Uso ajustados mediante la siguiente ecuación:

UCP = UUCP * TCF * EF

Donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Factor de complejidad técnica: Este coeficiente se calcula mediante un conjunto de Factores que determinan la complejidad del sistema, cada factor se cuantifica con un valor de 0 a 5.

Significado de los valores:

0: No presente o sin influencia

1: Influencia incidental o presencia incidental

2: Influencia moderada o presencia moderada

3: Influencia media o presencia media

4: Influencia significativa o presencia significativa

5: Fuerte influencia o fuerte presencia

La ecuación para su cálculo es:

TCF = 0.6 + 0.01 * Σ (Peso i * Valor Asignado i)

Factor	Descripción	Peso	Peso * Valor
T1	Sistema distribuido.	2	2 * 4
T2	Objetivos de performance o tiempo de respuesta.	1	1 * 3
T3	Eficiencia del usuario final.	1	1 * 5
T4	Procesamiento interno complejo.	1	1 * 2

T5	El código debe ser reutilizable.	1	1 * 5
T6	Facilidad de instalación.	0.5	0.5 * 5
T7	Facilidad de uso.	0.5	0.5 * 5
T8	Portabilidad.	2	2 * 4
T9	Facilidad de cambio.	1	1 * 3
T10	Concurrencia.	1	1 * 2
T11	Incluye objetivos especiales de seguridad.	1	1 * 3
T12	Provee acceso directo a terceras partes.	1	1 * 2
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	1 * 2
Total		48	

Tabla 4 Factor de complejidad técnica

Entonces, TCF = 0.6 + 0.01 * 48 = 1.08

Factor de ambiente: Contempla las habilidades y el entrenamiento del grupo de desarrollo por su importancia en las estimaciones de tiempo. Al igual que el factor de complejidad técnica se cuantifican con valores de 0 a 5. La ecuación para su cálculo es:

EF = 1.4 - 0.03 * Σ (Peso i * Valor Asignado i)

Factor	Descripción	Peso	Peso * Valor
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	1.5 * 3
E2	Experiencia en la aplicación.	0.5	0.5 * 1
E3	Experiencia en orientación a objetos.	1	1 * 4
E4	Capacidad del analista líder.	0.5	0.5 * 4
E5	Motivación.	1	1 * 4
E6	Estabilidad de los requerimientos.	2	2 * 3
E7	Personal part - time.	-1	-1 * 3

E8	Dificultad del lenguaje de programación.	-1	-1 * 3
Total			15

Tabla 5 Factor de ambiente

Entonces, EF = 1.4 - 0.03 * 15 = 0.95

Finalmente, UCP (Puntos de Caso de Uso ajustados) = UUCP * TCF * EF = 58 * 1.08* 0.95= 59.51

5.1.3 Estimación del esfuerzo

El esfuerzo en horas - hombre viene dado por:

E = UCP * CF

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: Factor de conversión.

CF = 20 horas-hombre (si Total EF ≤ 2)

CF = 28 horas-hombre (si Total EF = 3 ó Total EF = 4)

CF = abandonar o cambiar proyecto (si Total EF ≥ 5)

Total EF = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Como Total EF = 2 + 0

Total EF = 2 CF = 20 horas-hombre (porque Total EF ≤ 2)

Luego E = 59.51 * 20 horas-hombre

E = 1190.2 horas-hombre

5.1.4 Distribución del Esfuerzo entre las diferentes actividades

Para llevar a cabo una estimación más completa de la duración total del módulo, se agrega el esfuerzo de las demás actividades. Para ello se plantea la distribución del esfuerzo entre las diferentes actividades:

Actividad	Porcentaje	Horas-Hombre
Análisis	10%	297.55
Diseño	20%	595.1
Implementación	40%	1190.2
Pruebas	15%	446.3
Otras actividades	15%	446.3
Total	100%	2975.45

Tabla 6 Distribución del Esfuerzo entre las diferentes actividades

El Esfuerzo Total sería 2975.45 horas-hombre, si estimamos teniendo en cuenta que un mes tiene 176 horas laborables, pues se trabajan 8 horas diarias 22 días al mes, entonces el Esfuerzo Total en mes-hombre sería 16.91 mes-hombre.

5.1.5 Calcular el costo de todo el proyecto

Costo = CHM * ET / CH

Donde: CH: Cantidad de hombres

CHM: Costo Hombre – Mes

ET: Esfuerzo Total

Si la Cantidad de hombres es 2 y se tiene un Salario Promedio mensual igual a $100.00.

Entonces CHM = CH * Salario Promedio

CHM = 2 * 100

CHM = 200.00

Luego Costo = CHM * ET /

CH Costo = 200.00 * 16.91 / 2

Costo = $ 1691

Hoy en día, la utilización de herramientas para la Gestión de Requisitos es muy importante ya que esto proporciona a cada empresa y compañía un avance productivo en cada software además de reducir los costos y tiempo empleando en la elaboración del producto.

La herramienta que se presenta será de gran utilidad a los proyectos que la utilicen, pues facilitará la calidad, eficiencia y el resultado en tiempo del proyecto que se realice.

Con la informatización de las nuevas funcionalidades a la herramienta se garantiza llevar a cabo un trabajo organizado y con menos posibilidades de cometer errores que impidan obtener un producto que no cumpla con las necesidades del cliente, además de que se ahorrarán grandes volúmenes de papel, tinta, presillas y el procesamiento con la documentación será menos tedioso para aquellos que lo realicen pues tendrán las herramientas necesarias para desarrollar estas actividades disponibles desde sus puestos de trabajo.

5.2 Análisis de costos y beneficios

Se analizaron todos los beneficios tanto tangibles como intangibles que brindará el uso de esta herramienta para la Administración de Requisitos en los proyectos involucrados dentro del alcance de programa de mejora. Al ya haberse usado la herramienta en varios proyectos hace que se pueda afirmar que será fácil de usar a pesar de que se trabajarán con nuevas funcionalidades que permitirán un mejor desempeño del proceso de Administración de Requisitos.

La utilización de OSRMT no aportará gastos de recursos a la universidad ya que la misma posee licencia GPL, y su base de datos e IDE de desarrollo son tecnologías libres con el objetivo de aprovechar las ventajas que proporcionan para cooperar con la migración del país al uso de software libre.

Una vez concluido el estudio de la factibilidad del sistema se puede decir que el tiempo estimado para el desarrollo del proyecto es de 5 meses aproximadamente, donde las funcionalidades que se proponen agregar a la herramienta brindan una serie de beneficios que contribuyen a minimizar los posibles errores que se puedan cometer en el proceso de Gestión de Requisitos, por lo que se puede decir que si es factible implementar dichas funcionalidades.

CONCLUSIONES

Con la realización de este trabajo se le dio cumplimiento a los objetivos que fueron propuestos, a partir de un análisis crítico de las diferentes herramientas existentes para la Administración de Requisitos, se optó por OSRMT toda vez que la misma incluía una buena parte de los requerimientos necesarios para tomarla como una base de partida.

A OSRMT se le incorporaron nuevas funcionalidades que le permiten optimizar sus propiedades y gestionar nuevos recursos para el desarrollo del proceso de Administración de Requisitos, con lo que se consigue abaratar los costos, reducir el tiempo de desarrollo de los productos, dotarlos de una mayor calidad y como valor agregado se reduce el volumen del archivo en formato duro y de fuerza de trabajo.

Un beneficio insoslayable consiste en que sus funcionalidades corren en distintos sistemas operativos, es multiplataforma, además su licencia es libre lo que la convierte en la más factible para nuestras posibilidades de entre las propuestas estudiadas. Se ha desarrollado en lenguaje Java, lo que coincide con las políticas de independencia tecnológica de la universidad.

RECOMENDACIONES

La realización de este trabajo posee gran importancia, ya que dota a la universidad de una herramienta que permite llevar a cabo la Administración de Requisitos en un software mediante funcionalidades que se realizaban anteriormente de forma manual por lo que se recomienda lo siguiente:

- Perfeccionar las funcionalidades existentes en la herramienta OSRMT en su versión original en el módulo de reportes que presenta inconsistencias a la hora de mostrar cada uno de los mismos.
- Lograr que todas las funcionalidades se realicen en un mismo módulo.
- Lograr igualdad de lenguaje en la interfaz de la herramienta original y de la nueva propuesta.
- Implementar la funcionalidad Importar requisitos del producto desde un fichero XMI que facilitará cargar al OSRMT los datos que contiene el fichero.

REFERENCIAS BIBLIOGRÁFICAS

[1] Sánchez Pescador, E. (2009). Análisis e implantación de una herramienta de gestión de requisitos para la gestión de servicios. Madrid.

[2] Presentación mundial de la versión española del Modelo de Mejora de Procesos CMMI para el desarrollo de software, Retrieved, junio 05, 2009 from http://www.alphagalileo.org/ViewItem.aspx?ItemId=58068&CultureCode=es

[3] Sparx Systems. (2007). Retrieved febrero 3, 2009

[4] Beth Chrissis, M, Konrad, M, & Shrum, S. (2009). Guía para la integración de procesos y la mejora de productos. Madrid: Pearson Educación.

[5] Carrera Carrera, S. (2007). Adquisición semi-automática del conocimiento: una arquitectura preliminar. Vigo.

[6] Guía Ubuntu. (2009, Enero 15). Retrieved mayo 8, 2010, from http://www.guia-ubuntu.org/index.php?title=PostgreSQL

[7] El Proceso Unificado de Desarrollo de Software. (2000). Madrid: Pearson Educación.

[8] Free Download Manager. (2007). Retrieved abril 15, 20, from http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

BIBLIOGRAFÍA

- (2010). 5138_Guía de Trazabilidad. Ciudad de La Habana.
- Alarcón Aldana, A. K., & Sandoval Valero, M. E. HERRAMIENTAS CASE PARA INGENIERIA DE REQUISITOS. Boyacá.
- Alvarez de Zayas, C. (1995). METODOLOGIA DE LA INVESTIGACION CIENTIFICA. Santiago de Cuba.
- Anaya, V., & Letelier, P. SmarTTrace: Una Herramienta para Trazabilidad de Requisitos en Proyectos basados en UML. Valencia.
- Beth Chrissis, M., Konrad, M., & Shrum, S. (2009). Guía para la integración de procesos y la mejora de productos. Madrid: Pearson Educación.
- Calisoft. (s.f.). Recuperado el 15 de marzo de 2010, de http://calisoft.uci.cu/
- Calisoft. (s.f.). Recuperado el 23 de marzo de 2010, de http://calisoft.uci.cu/
- Canales Mora, R. (2003-2010). adictosaltrabajo. Recuperado el 5 de febrero de 2010, de http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=RationalSoftwareArchitectyRationalRequisitePro
- Carrera Carrera, S. (2007). Adquisición semi-automática del conocimiento: una arquitectura preliminar. Vigo.
- Chrissis, M. B., Konrad, M., & Shrum, S. (2009). Guía para la integración de procesos y la mejora de productos. Madrid: Pearson Educación.
- (2003). Cómo escribir... el título y el resumen de un artículo.
- Delgado Dapena., M. D. Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos. Ciudad de La Habana.

- El Proceso Unificado de Desarrollo de Software. (2000). Madrid: Pearson Educación.
- FileHeaven.com. (2003-2007). Recuperado el 4 de febrero de 2010, de http://www.fileheaven.com/descargar/smarttrace/19320.htm
- Free Download Manager. (2007). Recuperado el 15 de abril de 20, de http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/
- GSINNOVA. (s.f.). Recuperado el 7 de Febrero de 2010, de Grupo Soluciones Innova: http://www.rational.com.ar/apertura/acerca.html
- Guía Ubuntu. (15 de enero de 2009). Recuperado el 8 de Mayo de 2010, de http://www.guia-ubuntu.org/index.php?title=PostgreSQL
- Hernández Meléndrez, E. (2006). Cómo escribir una tesis.
- ipcorp. (8 de Diciembre de 2008). Recuperado el 10 de febrero de 2010, de http://www.ipcorp.com.ar/blog/2008/12/11/osrmt-open-source-requirements-management-tool/
- Lam Díaz, R. M. (s.f.). Entorno Visual de Aprendizaje. Recuperado el 2 de junio de 2010, de http://eva.uci.cu/mod/resource/view.php?id=2740
- Larman, C. (1999). UML Y Patrones Introducción al análisis y diseño orientado a objetos. México: Pearson.
- LuAuF. (13 de marzo de 2008). Recuperado el 15 de Mayo de 2010, de http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/
- newsgrupos. (2007). Recuperado el 20 de febrero de 2010, de http://www.newsgrupos.com/es-comp-programas/35417-osrmt-sdlc-herramienta-de-gerencia-de-los-requisitos-de-la-fuente-de-la-pluma.html
- (2006). Proyecto Vulcano Forja de proyectos software de calidad. San Francisco.

- Sánchez Pescador, E. (2009). Análisis e implantación de una herramienta de gestión de requisitos para la gestión de servicios. Madrid.

- SARAVIA GALLARDO, M. A. (2006). Orientación metodológica para la elaboración de proyectos e informes de investigación. Buenos Aires.

- SERENA. (s.f.). Recuperado el 19 de noviembre de 2009, de http://www.serena.com/geo/SP/solutions/software-requirements/

- Silvia Tabares, M., Barrera, A. F., Arroyabe, J. D., & Pineda, J. D. (2007). UN MÉTODO PARA LA TRAZABILIDAD DE REQUISITOS EN EL PROCESO UNIFICADO DE DESARROLLO. EIA, 69-82.

- Sparx Systems. (2007). Recuperado el 3 de febrero de 2009

GLOSARIO DE TÉRMINO

Calisoft: Es el centro de calidad para soluciones tecnológicas en la UCI, garantiza el crecimiento continuo de una producción de software con calidad en la organización.

CMMI: Es un modelo de madurez de mejora de los procesos para el desarrollo de productos y de servicios. Consiste en las mejores prácticas que tratan las actividades de desarrollo y de mantenimiento que cubren el ciclo de vida del producto, desde la concepción a la entrega y el mantenimiento.

Fichero XMI: Es una especificación para el intercambio de diagramas, escrita para compartir modelos UML entre diferentes herramientas de modelado.

Open Source: Permite a los desarrolladores leer, redistribuir, y modificar el código fuente de una aplicación.

Plugins: Es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Proveedor: (1) Una entidad que entrega productos o realiza servicios que han sido adquiridos. (2) Un individuo, sociedad, empresa, corporación, asociación u otros servicios que tienen un acuerdo (contrato) con un comprador para el diseño, el desarrollo, la fabricación, el mantenimiento, la modificación o el suministro de elementos bajo los términos de un acuerdo (contrato).

REQM: (1) Una condición o capacidad necesitada por un usuario para solucionar un problema o lograr un objetivo. (2) Una condición o capacidad que debe cumplir o poseer un producto o componente de producto para satisfacer un contrato, un estándar, una especificación u otros documentos impuestos formalmente.