Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Desarrollo de una herramienta para la Gestión y Trazabilidad de Requisitos de Software para el Centro de Desarrollo GEYSED

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

Autores

Aindamáis Téllez Pardo

Eliober Rojas Ramos

Tutores

Ing. Yusdenys Pérez Mendoza

Ing. Solangel Rodríguez Vázquez

Co-tutor

Ing. Carlos Luis Serrano Rosales

"La Habana, 21 de Junio de 2013."

"Año 55 de la Revolución."

"Tenemos que cambiar la tradicional actitud ante la construcción de software. En vez de pensar que nuestra principal tarea es indicar a un ordenador qué hacer, concentrémonos en explicar a las personas lo que queremos que el ordenador haga."

Donald E. Knuth

DECLARACIÓN DE AUTORÍA

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

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

Nombre del autor

_____

Firma del Autor

Nombre del tutor

_____

Firma del Tutor

Nombre del autor

_____

Firma del Autor

Nombre del tutor

_____

Firma del Tutor

DATOS DE CONTACTO

Autores:

- *Aindamáis Téllez Pardo*

Correo electrónico: apardo@estudiantes.uci.cu

- *Eliober Rojas Ramos*

Correo electrónico: erramos@estudiantes.uci.cu

Tutores:

- *Ing. Yusdenys Pérez Mendoza*

Correo electrónico: ypmendoza@uci.cu

- *Ing. Solangel Rodríguez Vázquez*

Correo electrónico: svazquez@uci.cu

Co-Tutor:

- *Ing. Carlos Luis Serrano Rosales*

Correo electrónico: cserrano@uci.cu

AGRADECIMIENTOS

Agradecimientos de Aindamáis:

A mis padres por traerme al mundo y ayudarme a forjarme como la mujer que soy. Mi mamá por su perseverancia, inteligencia y amor único de madre; y a mi papá por ser un ejemplo a seguir, por sus palabras de aliento y apoyo ilimitado; y a toda mi familia por el apoyo, la intransigencia e incondicionalidad en cada uno de los momentos de mi vida. A mi eterno y fiel amigo eterno Ernesto Antonio (Nestry) por ayudarme en todos estos años de universidad, por tolerar mis malcriadeces y acompañarme en los momentos más lindos así como dolorosos de mi carrera, gracias a mi otra familia, Clara, Raúl, Lissanca, Quiñones y David. A mi amiga y hermana Lisandra por ser la persona maravillosa con la que siempre he podido contar (eres parte de mi corazón); y a mis hermanitas Reina, Yaniset y Lisbet por habernos tolerado mutuamente todos nuestros caprichos, por haberme obsequiado su amistad y haber compartido cada uno de los instantes que me regalaron a su lado. A todos mis amigos por haber sabido llegar a mí de la manera más tierna y apoyándome en todo momento: Abraham, Juan Manuel, Iván, Ivanni, Yunier Alexander, Yuniel, Jorge Emilio, Carlos Luis, Darlon, Manuel Antonio, Arturo, Wilber, Roberto y Ariel. A mi primer grupo en la universidad, eternos amigos y compañeros, el 9109. A mi compañero de tesis Eliober, sin tí, de corazón no hubiese podido graduarme, has sido mi dolor de cabeza, mi paño de lágrimas, gracias por tolerarme tanto y gracias por ser además de compañero, mi amigo. A las niñas de mi apartamento, especialmente a Yanela por acompañarme en difíciles circunstancias. Gracias a todas las personas que he conocido durante los años de estudios, por ser amigos y compañeros. A mis amistades de Bayamo, por regalarme un pedacito de sus corazones. Al Decano de la Facultad 6, Reynaldo Rosado Roselló, por haberme brindado su apoyo y ayudado en momentos tan difíciles de mi carrera y vida personal, gracias profesor. A mis nuevas amistades Noe, Delmis, Fernado y Camejo, por haber llegado a mi vida en uno de los momentos más dolorosos de la misma, apoyarme incondicionalmente y regalarme su amor. Gracias a los tutores por su apoyo moral. Gracias a todos los profesores, amigos y compañeros que aportaron un pequeño grano de arena en la realización de esta tesis. Gracias a la Revolución y al Comandante Fidel Castro por darme la oportunidad de realizar mi vida profesional como Ingeniero Informático. Gracias a todas las personas que de una forma u otra han sido capaces de ayudarme con una palabra, un gesto o un abrazo, por formar parte de mí, por guiarme, levantarme el ánimo, la autoestima y creer en mí cuando yo misma no lo hacia, a todos muchísimas gracias.

AGRADECIMIENTOS

Agradecimientos de Eliober:

A Cristo redentor y la Virgen María por ser guía y mi luz, a mi mamá, por estar conmigo en todo momento, en pensamiento y físicamente dándome fuerzas, a mi padre por ser ejemplo de constancia, de humildad, de coraje y fortaleza, gracias a los dos estoy hoy aquí, gracias Dayami por estar siempre ahí cuando lo necesité y por darme lo más bello que tengo en mi vida hoy, mi hija. Gracias a Eric, Tato, Jose, Juan, Jorge y a todo el que se considere mi hermano por brindarme su amistad de forma incondicional, y ser buenos amigos, hermanos y compañeros en todo momento. A mi compañera de tesis por el apoyo y los dolores de cabeza, pero que rindió sus frutos al final. Gracias a Elizabeth por apoyarme incondicionalmente sin dar un paso atrás en ningún momento, por ser una gran compañera y ser tan comprensible. Gracias a mi hermanito Danny y mi hermanita Orquídea por estar siempre cuando necesité que alguien me escuchara, a todos mis compañeros de aula, los que están y los que no están, a mis amistades: Liz, Yami, Marcial, a los piquetes más sonados de la UCI con los que he bailado y he compartido: Los sensuales de la Pista, Abusadores de la salsa, Ilú Ashé, Los Trigueños, La Crema, a las amistades de muchos ratos en los deportes, el básquet, el balonmano y el fútbol y a los socios de cada noche en la recre y para todos los que se sienten amigos míos en general. Gracias a la revolución por darme esta oportunidad de hacerme un hombre de bien, formado bajo los conceptos de nuestra patria.

DEDICATORIA

Dedicatoria de Aindamáis:

A mis padres por apoyarme incondicionalmente.

A mis abuelos maternos y paternos por regalarme su sabiduría y ser parte de mi vida.

A mis hermanas, en especial a mi hermana e ídolo Aymét por ser mi mejor amiga y confidente.

A mis tías y tíos, principalmente a mi tía y madre María Virgen por ser un punto de apoyo y un ejemplo a seguir en mi vida.

A mis sobrinos, por ser mis primeros hijos; y a toda mi familia en general.

A mi hermana del alma Leamny por ser una de las personas especiales que han formado parte de mi vida.

A Maikel Luis por ser el angel que inspiró confianza en mí y me dio fuerzas cuando los dos más la necesitábamos para seguir adelante, por ser una de las personitas más espaciales de mi vida, por su cariño, comprensión, y su amistad verdadera.

A todas las personas que han formado, forman y formaran parte de mi vida.

Dedicatoria de Eliober:

Primero que todo a Cristo todo poderoso y la Virgen de la Caridad del Cobre.

A mi mamá, a mi papá que son una guía y punto fundamental de apoyo en mi vida, sin ustedes no fuera quien soy.

A lo más bello que es mi hija Milena Stephany, eres ahora mismo la razón de mi existir.

A Daya, por su incondicional apoyo y todos mis familiares en general.

A todos mis hermanos, los que así considero, que siempre han estado conmigo, los quiero y los llevo en mi corazón.

A Eliza por apoyarme en todo momento y fundamentalmente a todos mis amigos, en la UCI, en Santiago, donde quiera que esté alguien que se sienta mi amigo y a mi revolución por brindarme esta gran oportunidad.

RESUMEN

En los últimos años, las nuevas Tecnologías de la Información y las Comunicaciones (TIC) han revolucionado a las comunidades científicas de todo el mundo. Cuba no está exenta y por ello se ha visto inmersa en un proceso de cambio, transformación e informatización de la sociedad.

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el centro de desarrollo Geoinformática y Señales Digitales (GEYSED), se manipula una gran cantidad de información respecto a los productos que son desarrollados en el mismo, la cual necesita ser procesada para facilitar su manejo. En el centro no se cuenta con una herramienta factible que ayude a procesar la información, lo que impide agilizar el trabajo de los analistas para mantener la información accesible.

El presente trabajo de diploma contiene un estudio de las principales tendencias, tecnologías y metodologías actuales relacionadas con la gestión de requisitos. Igualmente se definen los principales requisitos del sistema y la arquitectura de software obteniéndose el diseño de la herramienta, la cual agilizará el proceso de gestión y trazabilidad de requisitos de software. La solución propuesta delimita el estilo arquitectónico, patrones de arquitectura y lenguaje de programación, que deben ser usados en la futura implementación de la herramienta.

Como resultado de la investigación se obtuvo la herramienta Gestor de Requisitos de Software (GRES). Con esta se logra mejorar en gran medida el control de la información referente a los proyectos productivos y ofrece un punto de apoyo para los analistas del centro GEYSED.

PALABRAS CLAVE

Gestión de requisitos y Trazabilidad de requisitos.

ABSTRACT

In the last years, the new Information and Communication Technololologies have revolutionized the scientific communities around the world. Cuba is not exempt and therefore it has been involved in a process of change, transformation and automation of the society.

In the University of Informatics Sciences, specifically in the development center Geoinformatics and Digital Signals (GEYSED), it is handled a great amount of information according to the products developed in it, which needs to be processed to facilitate its management. The center does not have a feasible tool to help process the information, which make difficult the quick work of analysts to keep information accessible.

This diploma thesis contains a study of the major trends, technologies and today's methodologies related to the management of requirements. It also identifies key system requirements and software architecture obtaining the tool design, which will streamline the management process and software requirements traceability. The proposed solution defines the architectural style, architectural patterns and programming language, to be used in the future implementation of the tool.

As a result of the investigation it was obtained the Requirements Manager Software tool (GRES). With this it is achieved in a great way the control of information concerning productive projects and provides a foothold for GEYSED center analysts.

KEY WORD

Requirement management, Requirement traceability.

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE DIAGRAMAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la informática, el avance tecnológico es cada vez más progresivo y la creación del mejor producto hace la competencia entre las empresas o instituciones encargadas de la producción de software. La elevada capacidad de los software, proporcionan encontrarlos en el mercado mundial con excelente calidad, para lograrlo se ha de contar con herramientas que ayuden a proveerla.

Con la finalidad de mejorar la productividad y calidad en el desarrollo de los proyectos de software, la gestión de requisitos es el proceso encargado de la identificación, asignación y seguimiento de requisitos de software para la creación de un producto. Existen herramientas que proporcionan funcionalidades enfatizadas al proceso de gestión de requisitos, siendo estos los definidos por el usuario para el logro del producto final.

Entre las necesidades a cubrir por estas herramientas está la disponibilidad de la información de los distintos proyectos en un repositorio, siendo este una localización o lugar de almacenamiento, generalmente colección de datos e información que funciona como salva de información digital, dígase: base de datos, archivos informáticos, documentos, código, recursos, al cual se pueda acceder por los distintos usuarios con el fin de ser reutilizada (Ecured, 2010).

Con el creciente avance en la producción de software se necesita de herramientas que vayan a su paso facilitando la creación de productos capaces de cumplir con los requisitos del cliente. Estas herramientas han de tener como objetivo facilitar la gestión, el control de cada uno de los requisitos que son solicitados por el cliente y la trazabilidad de los mismos, además de reducir el tiempo y esfuerzo invertidos en la creación de un producto.

En cada país existen empresas encargadas de la producción de software y cada una de ellas cuenta con herramientas orientadas a la gestión de requisitos. En Cuba, específicamente en la Universidad de las Ciencias Informáticas (UCI), se cuenta con un desarrollo de software naciente que trata de cumplir con las expectativas actuales de la producción de software.

En las minutas de reunión del Grupo de Trabajo Técnico de Administración de Requisitos o TWG (Technical Working Group) por sus siglas en inglés, quedó determinado el uso de Rational RequisitePro, Enterprise Architect y Open Source Requirement Management Tool como herramientas para la gestión de requisitos de software, decisión que fue aprobada a nivel del Grupo de Ingeniería de Procesos o EPG (Engineering Process Group) y Dirección del Grupo

Directivo o MSG (Management Steering Group), donde se encontraban todos los directores de los centros de desarrollo.

En el centro de desarrollo Geoinformática y Señales Digitales (GEYSED) perteneciente a la Facultad 6, en los proyectos productivos, con la migración a software libre, se utiliza la herramienta OSRMT, diseñada para dar cobertura a todo el ciclo de vida de desarrollo del software. Dispone de control de versiones, permite definir requisitos derivados y es posible definir tanto casos de uso como casos de prueba.

Según los analistas del centro el uso de esta herramienta no le es de utilidad pues no permite exportar un fichero que contenga toda la información, de manera que la información que es creada en una estación de trabajo, no puede ser reutilizada en otras estaciones donde se encuentra instalada esta herramienta. Al mismo tiempo el trabajo con la trazabilidad no es factible, debido a que el proceso de crear dependencias entre cada uno de los artefactos, se hace solo de manera unidireccional, lo que es ineficiente debido a que provoca demoras innecesarias y gran esfuerzo por parte del usuario cuando se tienen muchos artefactos a relacionar.

Esta herramienta no presenta un soporte empresarial, es decir, la empresa productora de la misma no presta servicios de mantenimiento y soporte ni capacitación al personal. La interfaz de usuario no es amigable y ocasionalmente lenta, lo cual causa demora en el tiempo de desarrollo del producto provocando retraso en el mismo. Además no posee la generación de un documento para presentarle al cliente el informe completo del análisis y la gestión y trazabilidad de requisitos que se ha realizado.

Por todo lo antes expuesto queda planteado como **problema a resolver:** ¿Cómo contribuir a la gestión y trazabilidad de requisitos de software, para agilizar el trabajo de los analistas del centro de desarrollo GEYSED? Según el problema antes descrito se plantea como **objeto de estudio:** Los procesos de gestión y trazabilidad de requisitos de software, delimitando como **campo de acción:** Los procesos de gestión y trazabilidad de requisitos de software en el centro de desarrollo GEYSED.

Teniendo como **objetivo general:** Desarrollar una herramienta para la gestión y trazabilidad de requisitos de software para el centro de desarrollo GEYSED.

Todo lo antes definido conduce a plantear la siguiente **idea a defender:** La implementación de una herramienta para la gestión y trazabilidad de requisitos de software permitirá que se pueda

gestionar cada uno de los requisitos especificados por el cliente llevando a cabo la trazabilidad de los mismos además de agilizar el trabajo de los analistas del centro GEYSED.

Para cumplir con el objetivo de esta investigación y resolver la situación problémica planteada, se proponen las siguientes **tareas de la investigación**:

1- Caracterizar los procesos relacionados a la gestión y trazabilidad de los requisitos de software.
2- Describir el estado del arte a nivel nacional e internacional de las herramientas existentes para la gestión y trazabilidad de los requisitos de software.
3- Caracterizar las herramientas y tecnologías a utilizar en el desarrollo de la herramienta de gestión y trazabilidad de requisitos de software del centro GEYSED.
4- Realizar el análisis y diseño de los artefactos que se involucran en la construcción de la herramienta.
5- Implementar la herramienta que posibilite la gestión y trazabilidad de requisitos de software.
6- Diseñar y Aplicar las pruebas al sistema.

En el transcurso de la investigación se utilizan diferentes métodos de investigación científica, siendo estos los siguientes:

- **Métodos teóricos:**
  - ✓ **Método Analítico - Sintético:** este método permitió la búsqueda de información relacionada con el proceso de gestión y trazabilidad de requisitos de software, sintetizando esta para desarrollar el proceso investigativo del presente trabajo dando paso al desarrollo de la herramienta según las necesidades generadas.
  - ✓ **Método Análisis histórico lógico:** mediante este método se realizó un estudio profundo de la herramienta existente que se relaciona con el objetivo de la investigación en cuanto a usabilidad, soporte y funcionalidades, obteniendo un punto de referencia y comparar los resultados entre la herramienta existente y la que se desarrollará.
- **Métodos empíricos:**
  - ✓ **Método Entrevista:** con la utilización de este método se pudo tener una comunicación con los analistas del centro de desarrollo GEYSED permitiendo recopilar información necesaria que brinda una visión para fundar la problemática planteada. La cual se puede ver en el Anexo1.

El contenido de este trabajo se distribuye de la forma siguiente:

INTRODUCCIÓN

Capítulo 1. Fundamentación Teórica: En este capítulo se realiza una introducción a los conceptos y definiciones de gestión y trazabilidad de los requisitos de software. Se realiza un estudio y caracterización de las herramientas propuestas para su uso en la Universidad de las Ciencias Informáticas, las cuales se nombran a continuación: Enterprise Architect, Rational Requisite Pro y Open Source Requirement Management Tool (OSRMT); además de la propuesta de la creación de una nueva herramienta.

Capítulo 2 Características del sistema: En este capítulo se analizan las funcionalidades del sistema orientadas a la identificación de las necesidades del usuario final. Se fundamenta el uso de la metodología y tecnologías sobre la cual se apoya la propuesta, además de las herramientas a utilizarse las cuales se verán reflejadas en la solución.

Capítulo 3 Análisis y Diseño del Sistema: En este capítulo se definen los conceptos asociados al modelo de dominio del problema. Se especifican los requisitos funcionales y no funcionales de la herramienta. Se describen los patrones y estilos arquitectónicos utilizados, así como la descripción de los casos de usos de la aplicación.

Capítulo 4 Implementación y Prueba: Este capítulo está encaminado a describir el proceso de implementación de la herramienta, con el objetivo de darle solución a los requisitos funcionales y no funcionales establecidos. En el mismo se describe el modelo de implementación que incluye los diagramas de despliegue y componente de la herramienta. Se realiza una descripción de los principales algoritmos utilizados en la implementación de la misma. Finalmente se realizan las pruebas a la herramienta para verificar el su correcto funcionamiento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo aborda, entre otros aspectos, temas sobre los conceptos asociados al dominio del problema para un mejor entendimiento del contenido de la presente investigación. El estado del arte de las herramientas para la gestión y trazabilidad de requisitos de software. Además, se realizará una breve caracterización de las herramientas Enterprise Architect, Rational Requisite Pro y Open Source Requirement Management Tool.

1.2 Conceptos asociados al dominio del problema

1.2.1 Requisito de software

Un requisito es una "condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado".

También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación (Laguna, 2004).

Se entiende por consiguiente que requisito no es más que las restricciones y propiedades que debe tener un software con el objetivo de satisfacer las necesidades y expectativas de un usuario.

1.2.2 Gestión

Acción o trámite que hay que llevar a cabo para conseguir o resolver alguna situación. Conjunto de operaciones que se realizan para dirigir y administrar un negocio o una empresa (Farlex, 2013).

Se entiende por consiguiente que *gestión* es un conjunto de actividades o elementos ligados que están encaminados a resolver un problema, funcionando como guía para encontrar una solución final.

1.2.3 Gestión de requisitos de software

Se utiliza para almacenar, examinar, revisar, hacer el seguimiento y navegar por los diferentes requisitos de un proyecto de software (Jacobson, 2000).

CMM (Capability Maturity Model), plantea que el propósito de la gestión de requisitos es establecer un entendimiento común entre el cliente y el proyecto de software de los requisitos del usuario que serán abordados por el proyecto de software (Álvarez, 2001).

1.2.4 Trazabilidad

Trazabilidad es la capacidad que tiene una organización o sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos (Canoy, 2005).

A consideración de los autores de la presente investigación se considera que la *trazabilidad* es un procedimiento que se realiza con el objetivo de dar seguimiento a los requisitos de un proyecto durante todo el ciclo de vida del mismo, de manera tal que se garantice el cumplimiento de los objetivos trazados para el proyecto.

1.2.5 Trazabilidad de requisitos de software

Según el estándar IEEE 830-1998 (*Institute of Electrical and Electronics Engineers, Inc. (Instituto de Ingenieros Eléctricos y Electrónicos)*), la trazabilidad es la habilidad para seguir la vida de un requerimiento en ambos sentidos, hacia sus orígenes o hacia su implementación a través de las especificaciones generadas durante el proceso de desarrollo. Es un factor de calidad (Ferraro, 2012).

La trazabilidad de requisitos es una actividad importante en el desarrollo de software, tanto para las posteriores etapas de mantenimiento como para analizar el impacto de cambios de requisitos por parte del cliente. Además constituye un elemento importante en el proceso de mejora continua de la organización y es vital para la gestión del conocimiento acumulado durante la ejecución del proyecto (González, 2011).

1.2.6 Matriz de trazabilidad

La matriz de trazabilidad es una herramienta que se utiliza para saber qué requisitos quedan cubiertos por una prueba (Softqatest, 2012).

La matriz de trazabilidad brinda información acerca de cuál de los involucrados del sistema, hizo la solicitud de un determinado requisito, lo que permite saber a quién dirigirse en caso de tener alguna duda, problema o quién debe de estar presente en caso de llevar a cabo alguna prueba de un requerimiento en particular (Sandoval, 2012).

De acuerdo con los autores del presente trabajo se entiende por *matriz de trazabilidad* como una tabla que relaciona cada requisito con su origen, permitiendo el seguimiento de cada requisito durante todo el ciclo de desarrollo de un proyecto.

1.3 Descripción general del objeto de estudio

Los procesos de gestión y trazabilidad de requisitos de software son los encargados de la identificación, asignación, verificación y modificación de los mismos, además de llevar el control o historial de todas las transformaciones que se realizan desde el levantamiento de los requisitos hasta el logro del prototipo funcional. Estos son esenciales en el ciclo de vida de desarrollo de un producto pues facilitan la dirección y alcance del mismo.

El uso de herramientas de gestión y trazabilidad de requisitos de software ha valido de apoyo en el trabajo con los requisitos de software y ha sido un puntal en la Ingeniería de Software, siendo importante en la fase de construcción de un producto. La trazabilidad se realiza con el fin de lograr un producto de máxima calidad exigida por el cliente.

Estas herramientas proporcionan beneficios considerables en la reducción de errores tales como: ambigüedad de requisitos, la que proporciona que los desarrolladores programen funcionalidades que no se ajustan a los requisitos especificados por el usuario, dando paso a la re-programación lo cual ocasiona pérdida de tiempo a causa de recodificación. Por otra parte pueden aparecer requisitos incompletos, estos son definidos parcialmente, es decir, que no logran un objetivo concreto; y por último los contradictorios, estos no son específicos por parte de los clientes causando incomprensión de las necesidades exactas de los usuarios.

La gestión de requisitos de software consiste en gestionar los cambios de los requisitos, las relaciones entre ellos, las dependencias entre la especificación de requisitos y otros documentos producidos por el proceso de desarrollo de software. De esta forma se asegura la consistencia entre los requisitos y el sistema construido, teniendo como objetivos principales: gestionar la recogida de requisitos, obtener la aprobación de los participantes del proyecto y gestionar los cambios, es decir, trazabilidad (INTECO, 2008).

La trazabilidad es el mecanismo que permite lograr un software concreto, fiable y mantenible. Esta práctica es la base de la gestión de los requisitos, puesto que brinda la información necesaria para su control y soporte a lo largo del proceso de desarrollo de software (Tabares, 2010).

En la Universidad de las Ciencias Informáticas se tienen como herramientas para la gestión y trazabilidad de requisitos de software: Rational RequisitePro, Enterprise Architect y Open Source Requirement Management Tool. En el centro GEYSED, con la migración a software libre, se estableció el uso de la herramienta OSRMT, que trabaja en arquitectura cliente/servidor, está

desarrollada bajo Java[1] siendo así multiplataforma, además de contener un módulo para trabajar la trazabilidad de requisitos de software.

OSRMT garantiza la trazabilidad entre todos los documentos relacionados con la Ingeniería de Requisitos, como son: funcionalidades, requisitos, casos de uso y casos de prueba. La gestión y trazabilidad de requisitos de software no es factible a través de la herramienta OSRMT pues el proceso de crear dependencias entre cada uno de los elementos es engorroso debido a que no permite que las mismas puedan hacerse de manera bidireccional, lo que provoca que al tener mayor cantidad de elementos a relacionar, el proceso sea ineficiente.

Esta herramienta no presenta un soporte empresarial, es decir, la empresa productora de la misma no presta servicios de mantenimiento y soporte, ni capacitación al personal; la interfaz de usuario no es amigable y ocasionalmente lenta, lo cual causa demora en el tiempo de desarrollo del producto provocando retraso en el mismo y no posibilita la generación de un documento para presentarle al cliente el informe completo del análisis y la gestión y trazabilidad de requisitos que se ha realizado.

Según los analistas del centro el uso de esta herramienta no es útil pues no permite exportar un fichero que contenga toda la información, de manera que la información que es creada en una estación de trabajo, no puede ser reutilizada en otras estaciones donde se encuentra instalada esta herramienta.

Además, presenta como desventaja fundamental que no exporta un fichero con un formato tipo[2], imposibilitando que el mismo pueda ser usado en otras estaciones de trabajo donde se encuentre instalado OSRMT.

Las características expuestas anteriormente, imposibilitan el uso de OSRMT por los analistas del centro de desarrollo GEYSED, lo que provoca que el proceso de gestión de requisitos de software en los proyectos productivos se realice de forma manual, siendo la actualización de la información compleja, cometiéndose errores que pueden evitarse empleando una herramienta capaz de realizar esta labor. Por tanto, es una necesidad para el centro de desarrollo GEYSED tener una herramienta para la gestión y trazabilidad de requisitos de software, brindando un aporte desde el punto de vista de utilidad y usabilidad, siendo esta capaz de cumplir con las expectativas y metas de los analistas del centro.

[1] Java: Lenguaje de programación orientado a Objetos (POO). Contiene tipo de datos, clases, objetos, variables, métodos, constructores y paquetes. Es utilizado para crear todo tipo de aplicaciones informáticas.

[2] Formato tipo: formato o extensión de un archivo generado por un sistema.

1.4 Sistemas actuales de Gestión/Trazabilidad de Requisitos de software

Actualmente la creación de un software es un factor primordial en el desarrollo de la economía. La Ingeniería de Requisitos juega un papel fundamental en el proceso de desarrollo de software, debido a que ofrece un componente vital a través de la gestión de requisitos. Observando la convergencia tecnológica para la cual tiende el mercado de software, se destaca el avance en la implementación de herramientas de gestión de requisitos. Estas herramientas han disminuido el esfuerzo de trabajo en el mantenimiento de requisitos, añadiendo beneficios considerables en la reducción de errores. Es indispensable para las compañías desarrolladoras, conocer ventajas, desventajas, impacto y visión de las herramientas de gestión y trazabilidad de requisitos en la actualidad, para de esta manera obtener productos de mayor calidad.

1.4.1 Funcionalidades generales de las Herramientas de Gestión de requisitos

Anteriormente las actividades implicadas en la captura, análisis, documentación y validación de requisitos eran realizadas manualmente, incrementando no solo el tiempo de entrega sino también el aumento de errores. Es por esto que las herramientas de gestión de requisitos son vitales, pero es indispensable conocer cuáles son las características y funcionalidades de las mismas.

- **Funciones Básicas**

Según un informe de INCOSE (Instituto de la Construcción en Seco que fue fundado en el año 1993 por las empresas líderes del mercado de la construcción en seco), para que una herramienta sea considerada como una herramienta de gestión de requisitos debe contener las siguientes características básicas:

- ✓ Identificación de requisitos.
- ✓ Clasificación de requisitos.
- ✓ Grupo de requisitos, revisión, identificación y punto de arranque.
- ✓ Proveer un interfaz de datos básicos.

- **Funciones Comunes**

Las funcionalidades comunes realizadas por las herramientas de gestión, consisten en la identificación de requisitos, revisión y edición, rastreo de requisitos, y generación de informes.

- **Funciones Técnicas**

Las funciones técnicas requeridas por las herramientas, incluye análisis del impacto del cambio; cuando un requisito es alterado, se deben identificar todos los requisitos afectados. Otra función beneficiosa a ser utilizada es la verificación de la integridad y la consistencia.

- **Función de Gestión**

Requerida por las herramientas consistes en la recopilación de métricas y supervisión de la estabilidad de los requisitos a través del control de cambio. Control de cambio, consiste en mantener la pista de añadir, borrar o cambiar cualquier requisito existente.

- **Funciones Auxiliares**

Algunas herramientas comerciales de gestión de requisitos incluyen como funciones auxiliares, análisis funcional y capacidades de simulación, cualquier construcción en la herramienta o en la forma de un producto compañero estrechamente integrado con el cual la herramienta hace la interfaz (Irotes, 2010).

1.4.2 Análisis de las propuestas de herramientas de gestión y trazabilidad de requisitos de software en la UCI

De un estudio realizado en la UCI, para la obtención de una herramienta capaz de proporcionar de manera satisfactoria la gestión y trazabilidad de requisitos de software, se encontraban las siguientes propuestas: Rational RequisitePro, Enterprise Architect y Open Source Requirement Management Tool, a continuación se muestra la caracterización de cada una de ellas.

1.4.2.1 Rational RequisitePro

Figura 1: Rational RequisitePro (EuRailCheck, 2010)

Esta herramienta permite que los miembros del equipo colaboren con los requisitos del proyecto. La interfaz es sencilla e incluye una aplicación de escritorio propietaria (plataforma Windows) y una aplicación Web. Los cambios en tiempo real que impactan el análisis permiten que cada miembro del equipo comprenda como afecta otras partes del proyecto (Rational, 2007).

Es fácil de usar ya que permite al autor del proyecto compartir sus requisitos, utilizando métodos familiares de documentación mientras mejora las capacidades de la trazabilidad y el análisis de impacto con el uso de bases de datos. El resultado es una mejor comunicación y administración de requisitos para lograr completar proyectos a tiempo.

Permite también la gestión de requisitos y casos de uso que propicia una mejor comunicación, mejoras en el trabajo de equipo y reduce el riesgo de los proyectos.

Presenta además las siguientes ventajas:

- Ofrece avanzada integración con Microsoft Word.
- Esta construido sobre una infraestructura de base de datos robusta.
- Provee trazabilidad en profundidad y análisis de alcance.
- Facilita el análisis de impacto de cambios detallados con trazas de auditoría y notificaciones por correo electrónico.
- Ayuda en la creación y comparación de líneas base de proyectos.
- Provee una interfaz Web escalable y rápida para equipos distribuidos.
- Ofrece opciones flexibles de reportes y crea la documentación requerida para cumplir con los estándares organizacionales, institucionales y gubernamentales.
- Integrable con múltiples herramientas de la Plataforma de Entrega de Software de IBM Rational (Sybven, 2011).

1.4.2.2 Enterprise Architect

Figura 2: Enterprise Architect (deviantart, 2013)

Es una herramienta comprensible de diseño y análisis UML[3], que cubre el desarrollo de software desde el paso de los requisitos a través de las etapas del análisis, modelos de diseño, pruebas y

[3] Lenguaje Unificado de Modelado es una herramienta para el análisis y diseño para sistemas de software, ofrece un lenguaje común para todos los desarrolladores.

mantenimiento. Además es una herramienta multi-usuario, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.

Esta herramienta provee trazabilidad completa desde el análisis de requisitos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue. Genera reportes detallados y complejos con la información que es necesitada en el formato que una compañía o cliente demande.

Soporta generación e ingeniería inversa de código fuente para muchos lenguajes como: C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP. Posee un editor de código fuente con "resaltador de sintaxis" incorporado que le permite navegar y explorar su modelo de código fuente en el mismo ambiente (Sparxsystems, 2010).

1.4.2.3 Open Source Requirement Management Tool (OSRMT)

Figura 3: Open Source Requirement Management Tool (OSRMT)

Es una herramienta de software libre pensada para asistir en todo el ciclo de vida del desarrollo del software. 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).

La herramienta integra módulos de Administración y Configuración, Gestión de Documentos de la Ingeniería de Requisitos, Trazabilidad entre documentos de trabajo e Informes y estadísticas.

Además de las funcionalidades ya mencionadas, este sistema provee:

- Gestión de la configuración: versionado y registro de los cambios realizados en los diferentes elementos.
- Gestión de usuarios y permisos.
- Herramientas de migración para los diversos cambios de versiones.
- Múltiples idiomas (importación y exportación para dar soporte a diversos idiomas).
- Importar y exportar información en XML y mediante línea de comandos.
- Exportar información en HTML mediante línea de comandos.

- Informes: (Básicos; Específicos creados por el usuario; A partir de los resultados de búsquedas avanzadasy Exportados a HTMLo PDF)

También es posible personalizar los atributos de las funcionalidades, requisitos, casos de prueba, se pueden configurar valores por defecto para los atributos, y personalizar las vistas (IPCorp, 2010).

Presenta además las siguientes ventajas:

- Permite el trabajo en equipo, donde cada integrante del grupo de desarrollo de proyectos puede formar parte y trabajar en la herramienta.
- Posee la generación de informes a PDF o HTML según como quiera el usuario.

1.4.3 Deficiencias de las herramientas de acuerdo al centro de desarrollo GEYSED que imposibilita su uso adecuado

Enterprise Architect y Rational RequisitePro son herramientas potentes, con múltiples funcionalidades, trazabilidad en profundidad y buen rendimiento, pero son herramientas propietarias, por lo que con el proceso de migración a software libre que se lleva a cabo en la universidad, el centro de desarrollo GEYSED estableció el uso de OSRMT, ya que es una herramienta multiplataforma diseñada para software libre (Ver Anexo 2). A partir de entrevistas realizadas a los analistas del centro de desarrollo GEYSED sobre el uso de esta herramienta, se arrojaron las siguientes conclusiones:

No es factible la gestión de requisitos a través de la herramienta OSRMT pues el proceso es engorroso debido a que:

- No se puede realizar la trazabilidad de forma viable pues no se relacionan fácilmente todos los elementos implicados en el proceso.
- Falta de generación de un documento para presentarle al cliente el informe completo del análisis y especificación de requisitos que se ha realizado.
- Su uso significa duplicar información existente en otro formato además de presentar problemas funcionales.
- La información creada en esta herramienta no puede ser reutilizada en distintas estaciones de trabajo.

1.5 Conclusiones del capítulo

Muchos proyectos fracasan por el incorrecto trabajo con los conceptos asociados a la trazabilidad de requisitos de software y la importancia e influencia de la misma en la calidad de los productos

finales, los cuales se derivan del proceso de desarrollo de software de dichos proyectos. Para darle solución a estos problemas se han creado herramientas para la gestión y trazabilidad de requisitos de software, haciendo que el uso de las mismas aporte beneficios considerables debido a las múltiples funcionalidades que brindan, dígase identificación y trazabilidad de requisitos de software y generación de reportes. La UCI es una de las entidades que hace uso de dichas herramientas; en el caso del centro GEYSED se concluye que la herramienta OSRMT, establecida y utilizada en dicha entidad, no es factible debido a que no cumple con las expectativas de los analistas del centro tales como: mejor organización y agilidad en el trabajo con la gestión de requisitos, por lo que para lograr lo antes mencionado, se considera favorable el desarrollo de una nueva herramienta que sea capaz de suplir las necesidades que OSRMT no satisface.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se abordan las tendencias y tecnologías que se utilizan en la realización de sistemas informáticos, siendo definidas las utilizadas en el presente trabajo de diploma. Además, de la explicación del uso de las herramientas y del lenguaje de programación, así como el lenguaje modelado y las metodologías a utilizar.

2.2 Metodología de desarrollo de software

Todo desarrollo de un software es complejo e inseguro a menos que tenga definido una metodología que ofrezca apoyo y guía para el logro de un producto final. Una metodología de desarrollo de software es un paradigma de trabajo (o entorno de desarrollo) que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información, siendo la misma definida por el equipo de desarrollo y utilizada en todo el ciclo de vida de desarrollo de un software.

2.2.1 Rational Unified Process (RUP)

El Proceso Unificado Rational (RUP) es un proceso de desarrollo de software que se ha ido desarrollando con el tiempo, el mismo que se maneja conjuntamente con el Lenguaje Unificado de Modelado UML para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP, es una metodología pesada, se desenvuelve en unos principios básicos que se deben cumplir para que el proyecto en desarrollo tenga éxito. Estos principios son: adaptar el proceso, balancear prioridades, demostrar valor iterativamente, elevar el nivel de abstracción y enfocarse en la calidad del producto (scruz334, 2007).

Esta metodología es más cómoda para utilizarla en grandes proyectos ya que necesita un grupo de trabajo el cual tiene que ser apto para dirigir un proceso complicado en varias fases. RUP corresponde a una metodología de trabajo intensiva en recursos, su aproximación al problema no sólo garantiza que los proyectos abordados sean ejecutados íntegramente, sino que además evita desviaciones importantes respecto de los plazos y también permite una definición acertada del sistema en un inicio para hacer innecesarias las reconstrucciones parciales posteriores (Zapata, 2012).

RUP ostenta tres características fundamentales *está dirigido por casos de uso*: orientan al proyecto a la importancia de lo que quiere el usuario y se pretende lograr, *está centrado en la arquitectura*: esta surge de las necesidades de la empresa además se realiza la toma de decisiones qué indica cómo y en qué orden debe ser construido el software, y por último *es iterativo e incremental*: donde se divide el proyecto en miniproyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera perfeccionada.

La metodología RUP está constituida por cuatro fases de desarrollo del producto:

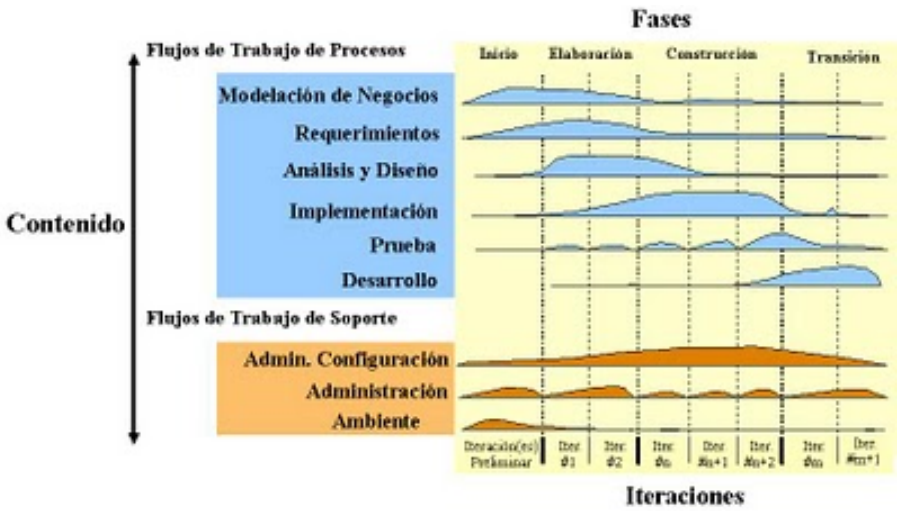

Figura 4: Fases de RUP (Jaldín, 2010)

- *Fase de inicio:* en esta fase se concibe la idea central del producto, se arma el documento de visión. El objetivo general de esta fase es establecer un acuerdo entre los interesados acerca de los objetivos del producto, se revisan y confirman estableciendo los objetivos centrales del negocio. Se asegura de identificar los riesgos relacionados con el negocio y requisitos. Para proyectos de mejora de software existente, esta fase es más breve y se centra en asegurar la viabilidad de desarrollar el proyecto.
- *Fase de elaboración:* el objetivo de esta fase es establecer la arquitectura base del sistema para proveer bases estables para el esfuerzo de diseño e implementación en la siguiente fase. La arquitectura debe abarcar todas las consideraciones de mayor importancia de los requisitos y una evaluación del riesgo.
- *Fase de construcción:* durante esta fase de construcción, el foco del producto se mueve de la arquitectura de base a un sistema lo suficientemente completo como para llevarlo al usuario, es decir, es la creación del producto.

➢ *Fase de Transición:* en la fase de transición el objetivos es garantizar que los requisitos se han cumplido, con la satisfacción de las partes interesadas. Incluye la preparación del ambiente, se completan, se identifican y corrigen defectos. Esta fase se enfoca en asegurar que el software esté disponible para sus usuarios. Se puede subdividir en varias iteraciones, además incluye pruebas del producto para poder hacer el entregable del mismo, así como realizar cambios menores de acuerdo a cambios menores propuestos por el usuario. En este punto, la retroalimentación de los usuarios se centra en depurar el producto, configuraciones, instalación y aspectos sobre utilización.

Está basado en 5 principales claves:

➢ *Adaptar el proceso:* el proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

➢ *Equilibrar prioridades:* los requisitos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. *Deben encontrar* un equilibrio que *satisfaga los deseos de todos*. Gracias a este desequilibrio se podrán corregir los desacuerdos que surjan en el futuro.

➢ *Demostrar valor iterativamente:* los proyectos se entregan, aunque sea de un modo interno, en estas etapas. En cada iteración se analiza la opinión de los inversores, la estabilidad y la calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

➢ *Elevar el nivel de abstracción:* este principio dominante motiva el uso de conceptos reutilizables tales como patrón de software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requisitos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Estas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.

➢ *Enfocarse en la calidad:* el control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente (Zapata, 2012).

Una vez finalizado el estudio se decide escoger la metodología RUP, debido a que permite la generación de gran cantidad de documentación y artefactos tales como: requisitos, casos de uso, diagramas y casos de prueba que pueden ser utilizados durante todo el ciclo de vida de desarrollo

de un producto. Además estos artefactos son los implicados en la trazabilidad de requisitos de software y de gran ventaja pues pueden ser útiles para futuras versiones de la herramienta a desarrollar.

2.3 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

UML es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo, posibilitando la captura, comunicación y nivelación de conocimiento estratégico, táctico y operacional para facilitar el incremento de valor, aumentando la calidad, reduciendo costos y reduciendo el tiempo de presentación al mercado; manejando riesgos y siendo proactivo para el posible aumento de complejidad o cambio (SlideShare, 2010).

Características de UML:

- *Ofrece vocabulario y reglas:* para crear y leer modelos bien formados que constituyen los planos de un sistema de software.
- *Es independiente del proceso de desarrollo:* un uso óptimo se consigue en procesos dirigidos por casos de uso, centrados en la arquitectura, iterativos e incrementales.
- Cubre las diferentes vistas de la arquitectura de un sistema mientras evoluciona a través del ciclo de vida del desarrollo de software (Vistas Software: estáticas, dinámicas).

Los beneficios de UML son:

- Mejores tiempos de desarrollo.
- Modelar sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos (Sánchez, 2010).

Luego del estudio realizado, se determinó escoger UML como lenguaje de modelado ya que es útil para el desarrollo del modelo visual de cualquier proyecto y mediante el mismo es posible establecer los requisitos y estructuras necesarias para modelar un sistema de software previo al proceso de implementación. Además es el lenguaje de modelado que propone la metodología propuesta, la misma permite que todos los integrantes de un equipo de trabajo, conozcan y

compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software, además de utilizar este lenguaje de modelado para el trabajo.

2.4 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática así como documentación o detección de errores (FreeDownloadManager, 2008).

2.4.1 Visual Paradigm

Visual Paradigm (VP) 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 (Giraldo, 2005).

Posee licencia gratuita y Comercial, además de poseer las siguientes características:

- Producto de calidad.
- Soporta aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones (Targetware, 2010).

VP para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.

También ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.

- Potente generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado (López, 2010).

Brinda una serie de facilidades para generar informes que permiten documentar el proyecto:

- Generación de informes en PDF.
- Generación de informes en HTML.
- Generación de informes en Word.
- Publicando el proyecto.
- Escribiendo un informe.
- Ordenando los elementos de un informe.

Visual Paradigm admite la Exportación/Importación permitiendo interoperabilidad ofreciendo una serie de facilidades para Importar/Exportar modelos en diferentes formatos ya sean: XMI, XML, casos de Uso a/desde Word e importar desde Rational Rose – Erwin (Rouse, 2005).

Después del estudio realizado de la herramienta Visual Paradigm, se determina que presenta un alto potencial en la realización del proceso ingenieril en los productos de software. Es conveniente el uso de la misma para el trabajo con la ingeniería de software, ya que cuenta con versiones multiplataforma, además es enteramente compatible con la metodología escogida, lo que proporciona generar mayor documentación para el producto que se desarrolla y es por la que se rigen los analistas del centro de desarrollo. Todo lo antes expuesto favorece obtener un producto con la calidad requerida.

2.5 Lenguajes de programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes (Zator Systems, 2013).

2.5.1 C++

C++ es un lenguaje imperativo orientado a objetos derivado del C[4]. En realidad es un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. No es un lenguaje orientado a objetos puros, pero se ha de reconocer que su sistema de detección de errores es robusto, por lo que algunos errores de éste serán rápidamente detectados (Tallercmasmas, 2012).

C++ tiene ciertas características que lo distinguen de los demás lenguajes. Las más sobresalientes son:

- *Programación Orientada a Objetos (POO):* la posibilidad de este tipo de programación permite al programador diseñar aplicaciones desde el punto de vista más como una comunicación entre objetos, que solo secuencia de código estructurado. También permite una gran reutilización de código de una manera más lógica y productiva.
- *Portabilidad:* prácticamente se puede compilar el mismo código C++ en casi cualquier tipo de computadora y sistema operativo sin hacer ningún cambio.
- *Brevedad:* el código escrito en C++ es muy corto en comparación con otros lenguajes, dado que el uso de caracteres especiales es utilizado en palabras claves (keywords), haciendo menos el esfuerzo del programador.
- *Programación Modular:* el cuerpo de una aplicación en C++ puede estar hecha de muchas fuentes (sources) que son compiladas por separado y luego unidas, ahorrando tiempo dado que no es necesario recompilar toda la aplicación al hacer pequeños cambios.
- *Velocidad (Speed):* el código resultante de una compilación en C++ es muy eficiente, debido a su dualidad como lenguaje de alto y bajo nivel y el reducido tamaño del lenguaje en si mismo (Delgado, 2011).

Otras características del lenguaje C++:

- Tiene un conjunto completo de instrucciones de control.
- Permite la agrupación de instrucciones.
- Incluye el concepto de puntero (variable que contiene la dirección de otra variable).
- Los argumentos de las funciones se transfieren por su valor.

[4] C: lenguaje de programación, de nivel medio, orientado a la implementación de sistemas y aplicaciones informáticas.

- Entradas y salidas (E/S) no forma parte del lenguaje, sino que se proporciona a través de una biblioteca de funciones.

Las diversas razones por la cual se ha convertido en un lenguaje de uso profesional son:

- El uso de constructores de alto nivel.
- El poder manejar actividades de bajo nivel.
- El generar programas eficientes.
- La posibilidad de poder ser compilado en una variedad de computadoras, con pocos cambios (portabilidad) (Pergamino Virtual, 2012).

2.5.2 Java

Java es un lenguaje de programación simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. Presenta muchas características que lo diferencian de lenguajes similares como C++, empezando por las posibilidades de ejecución.

Básicamente un programa en Java puede ejecutarse como:

- *Stand Alone:* aplicación independiente.
- *Applet:* una aplicación especial que se ejecuta en el navegador del cliente.
- *Servlet:* una aplicación especial sin Interfaz que se ejecuta en servidor.

Por otro lado un desarrollo en Java nunca empieza desde cero. Este implementa consigo un gran número de clases, incluidas gratuitamente en su entorno de desarrollo, para realizar muy variadas tareas que permiten al desarrollador centrarse en las características más inherentes a su proyecto (Globedia, 2012).

Las características principales que ofrece Java son:

- *Lenguaje Simple:* se lo conoce como lenguaje simple porque viene de la misma estructura de C y C++; ya que C++ fue un referente para la creación de Java por eso utiliza determinadas características de C++.
- *Orientado a Objeto:* toda la programación, en su mayoría, está orientada a objeto, ya que al estar agrupados en estructuras encapsuladas es más fácil su manipulación.
- *Distribuido:* permite abrir sockets, establecer y aceptar conexiones con los servidores o clientes remotos; facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red.

- *Indiferente a la arquitectura:* Java es compatible con los más variados entornos de red, cualesquiera sean estos desde Windows 95, Unix a Windows Nt y Mac, para poder trabajar con diferentes sistemas operativos. Java es muy versátil ya que utiliza byte-codes que es un formato intermedio que sirve para transportar el código eficientemente o de diferentes plataformas (Hardware - Software).
- *Portable:* por ser indiferente a la arquitectura sobre la cual está trabajando, esto hace que su portabilidad sea muy eficiente, sus programas son iguales en cualquiera de las plataformas, ya que java especifica tamaños básicos, esto se conoce como la máquina virtual de Java.
- *Interpretado y compilado a la vez:* puede ser compilado e interpretado en tiempo real, ya que cuando se construye el código fuente este se transforma en una especie de código de máquina.
- *Dinámico:* este lenguaje es muy dinámico en la fase de enlazado, sus clases solamente actuarán en medida en que sean requeridas o necesitadas, con esto permite que los enlaces se puedan incluir desde fuentes muy variadas o desde la red.
- *Produce Applets:* se pueden crear aplicaciones independientes y applets. Independientes porque se pueden comportar como cualquier programa escrito en cualquier lenguaje.
- *Alto rendimiento:* este lenguaje es considerado de alto rendimiento por ser tan veloz en el momento de correr los programas y por ahorrarse muchas líneas de código (Globedia, 2012).

2.5.3 Fundamento del lenguaje seleccionado

Después de haber realizada un estudio detallado de las características que poseen cada uno de los lenguajes de programación explicados anteriormente, se concluye que java es un lenguaje con potencial y que muestra un elevado nivel de rendimiento en cuanto a velocidad de ejecución, pero presenta el inconveniente de ser dependiente de la máquina virtual mientras que C++ se puede ejecutar sin aplicaciones externas. Además C++ es un lenguaje flexible y poderoso, permite la obtención de buenos resultados con su uso y tiene la ventaja con respecto a Java de presentar un soporte directo para todas las construcciones que se realicen en la implementación de la herramienta, trabaja a más bajo nivel y realiza una gestión de memoria más profunda que propicia que tenga un mayor nivel de rendimiento, lo que representa una ventaja ya que se busca lograr que la herramienta a desarrollar cuente con un mejor rendimiento y menor tiempo de respuestas antes las solicitudes realizadas por el usuario.

2.6 Entorno de desarrollo

Un entorno de desarrollo integrado (en inglés Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

Un IDE es un programa informático compuesto por un conjunto de herramientas para programar en un lenguaje de programación o varios, donde se puede encontrar como mínimo un editor, compilador, intérprete y depurador de uno o varios lenguajes de programación (Alemany, 2009).

2.6.1 Qt

Qt es un framework de desarrollo de aplicaciones multiplataforma. Viene acompañado de un conjunto de herramientas para facilitar su uso. Incluye una serie de funcionalidades que no están normalmente en C++. Para ello utiliza su propio preprocesador llamado MOC (Meta Object Compiler) (zonaqt, 2010).

Qt es un framework muy poderoso; la función más conocida es la creación de interfaces gráficas o de usuarios, escrita en un lenguaje particular; además, ofrece una suite de aplicaciones para facilitar y agilizar las tareas de desarrollo, siendo las mismas las siguientes:

- *Qt Assistant:* herramienta para visualizar la documentación oficial de Qt.
- *Qt Designer:* herramienta WYSIWYG para crear interfaces de usuario.
- *Qt Linguist:* herramienta para la traducción de aplicaciones.
- *Qt Creator:* IDE para el lenguaje C++, pero especialmente diseñado para Qt, integra las primeras dos herramientas mencionadas (qt-project.org, 2012).

Una de las mayores ventajas de Qt Creator es que permite que un equipo de desarrolladores comparta un proyecto a través de diferentes plataformas de desarrollo (Microsoft Windows®, Mac OS X®, and Linux®) con una herramienta común para desarrollo y depurado.

Su objetivo principal es conocer las necesidades de desarrollo de los programadores que buscan la simplicidad, facilidad de uso, productividad, extensibilidad y apertura, mientras se baja la barrera de entrada para los recién llegados a Qt. Las características clave de Qt Creator permiten a los programadores realizar las siguientes tareas:

- Empezar a desarrollar aplicaciones con Qt rápida y fácilmente con el asistente de proyectos, y acceder velozmente a proyectos y sesiones recientes.
- Diseñar aplicaciones de interfaz de usuario basadas en widgets Qt con el editor integrado, Qt Designer.

- Desarrollar aplicaciones con el editor de código avanzado de C++ que provee nuevas y poderosas características para completar recortes de código, remanufactura de código, y viendo el esquema de archivos (que es, la jerarquía de símbolos de un archivo).
- Compilar, correr y desarrollar proyectos Qt que se dirigen a múltiples plataformas de escritorio y móviles, como Microsoft Windows, Mac OS X, Linux, Symbian, MeeGo, y Maemo.
- Depurar con el depurador GNU y el CDB usando una interfaz gráfica de usuario con una mayor conciencia de las estructuras de clases Qt.
- Usar herramientas de análisis de código para verificar la administración de memoria en sus aplicaciones.
- Desarrollar aplicaciones para dispositivos móviles y crear paquetes de instalación para dispositivos Symbian, MeeGo, y Maemo que pueden ser publicadas en la tienda Ovi y otros canales.
- Acceder fácilmente a información con el módulo del sistema de ayuda contextual de Qt (qt-project.org, 2012).

Luego del estudio realizado del IDE de desarrollo, se determina que Qt presenta las características necesarias para dar cumplimiento al objetivo general trazado, ya que está diseñado para desarrollar aplicaciones e interfaces de usuario y desplegarlas a múltiples sistemas operativos, tanto móviles como de escritorio. Además, contiene un gran número de librerías que facilitan su uso, es personalizable y libre, y es una herramienta con un alto rendimiento y velocidad, permite el trabajo con interfaces de manera sencilla y utiliza como lenguaje nativo a C++, que es el lenguaje de programación seleccionado para el desarrollo de la herramienta.

2.7 Framework de desarrollo

Un Framework (marco de trabajo o marco de desarrollo) es un conjunto de librerías y componentes de probada solvencia, junto con una documentación y metodología de uso, que permite diseñar, construir e implantar aplicaciones corporativas de forma más uniforme, rápida, y con mayor calidad (Casas, 2012).

Para la realización del presente trabajo se utiliza como framework de desarrollo Qt, ya que es un framework multiplataforma para el desarrollo de aplicaciones con interfaz gráfica de usuario o de consola y utiliza C++ como lenguaje de programación. Incorpora un estilo de programación a C++, que le hace menos vulnerable a los fallos de programación, sobre todo en el manejo de memoria, y que lo hace muy agradable de cara al desarrollador. Cuenta además con un conjunto de librerías

que son clases escritas en C++ para facilitar el desarrollo y posee un editor de código sofisticado el cual ofrece soporte para la edición de C++, ayuda sensible al contexto, completamiento de código y buena navegación. Permite también el diseño de interfaces de usuario integradas además de proporcionar editores visuales integrados. Contiene herramientas y entornos de desarrollo multiplataforma con un alto rendimiento en todas las plataformas y presenta componentes que facilitan el trabajo a los desarrolladores.

2.8 Conclusiones del capítulo

Luego de realizar un análisis de las principales características acerca de las tecnologías y herramientas existentes que puedan ser factibles en la construcción y desarrollo de la herramienta propuesta, se concluye que las más apropiadas para el desarrollo del presente trabajo por sus peculiaridades, ventajas y desventajas son las que se mencionan a continuación: como metodología de desarrollo RUP; UML como lenguaje de modelado; Visual Paradigm en su versión 8.0 como herramienta CASE; para un desarrollo eficaz y efectivo de la herramienta se define C++ como lenguaje de programación; como marco de trabajo, Qt en su versión 4.8 por ser un framework de desarrollo multiplataforma y como IDE de desarrollo Qt Creator en su versión 2.4.

Con las tendencias actuales y las políticas de migración hacia el software libre en el país y la universidad, se determinó que las herramientas seleccionadas son las más factibles para la realización de la solución que se propone en el trabajo, por lo que una vez concluido el capítulo quedan sentadas las bases para iniciar el desarrollo de la herramienta para la gestión y trazabilidad de requisitos de software para el centro GEYSED.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se realiza la descripción y el diseño del prototipo funcional que se presenta en este trabajo. Se modela el dominio de problema para una mejor comprensión del medio donde se ubica. Se describen una serie de conceptos que son agrupados en el modelo. Se realiza una especificación de casos de uso del sistema, además de mostrar el diagrama de los mismos, los diagramas de clases del análisis y los diagramas de interacción de cada caso de uso.

3.2 Modelo de dominio

El modelo de dominio es un punto de partida para lograr un mejor diseño del sistema a desarrollar y una comprensión más clara del problema a resolver. En el se muestran las clases conceptuales más significativas asociadas al dominio del problema. Está considerado como parte del modelo de negocio, pues se centra en una parte del mismo relacionada con el ambiente del proyecto (Sosa, 2013).

A partir de un estudio realizado sobre las diferencias existentes entre el modelo de dominio y modelo del negocio, en el libro titulado Ingeniería de Software del autor Benet Campderrich Falgueras que se encuentra disponible en la bibliografía del presente trabajo, se determina utilizar modelo del dominio ya que para el desarrollo de la herramienta no se cuenta con procesos bien definidos en el entorno del negocio, además de que los requisitos definidos son cambiantes. Esto, unido a que no se cuenta con un cliente real, provoca que se dificulte determinar los elementos más importantes a tener en cuenta en el desarrollo de la herramienta. Por ello se hace necesario un modelado del dominio, donde se puedan identificar entidades, objetos y eventos involucrados en el entorno del problema. A continuación se muestra una descripción de los conceptos fundamentales asociados al dominio del problema.

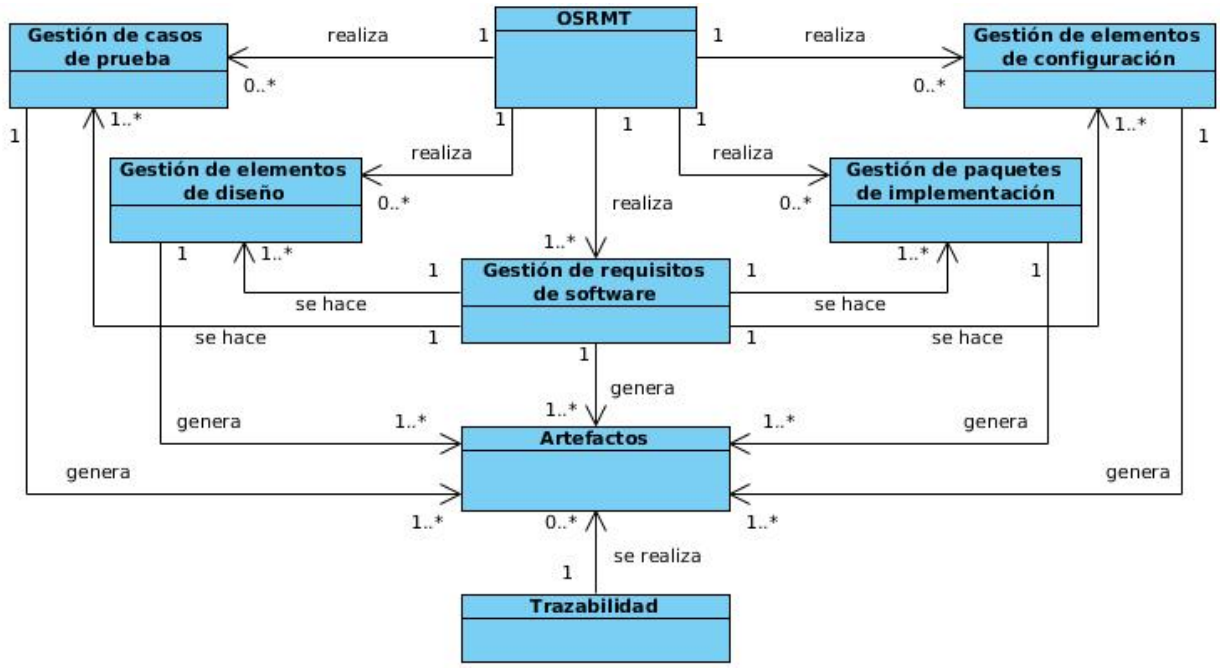

Diagrama 1: Diagrama de clases del dominio

3.2.1 Descripción de los conceptos fundamentales

Open Source Requirement Management Tool (OSRMT): es una herramienta de código abierto que realiza la Gestión de requisitos, Gestión de casos de prueba, Gestión de elementos de configuración, Gestión de elementos de diseño, Gestión de paquetes de implementación. Por cada uno de estos procesos se generan un grupo de artefactos sobre los cuales se realiza la trazabilidad.

La **Gestión de requisitos de software:** es gestionar un conjunto de requisitos propuestos por un cliente.

La **Gestión de casos de prueba:** es la posibilidad de gestionar los casos de prueba que se ha de realizar en un proyecto.

La Gestión de elementos de configuración: son todos los posibles documentos que se pueden gestionar.

La **Gestión de elementos de diseño:** es la gestión de todos los elementos asociados al diseño.

La **Gestión de paquetes de implementación:** es la gestión de los paquetes de implementación.

Un **artefacto** son los elementos que se generan durante el ciclo de desarrollo de un proyecto.

La **trazabilidad** es la habilidad de describir y seguir la vida de los artefactos generados.

3.3 Especificación de los requisitos del prototipo funcional

La captura de requisitos es uno de los procesos críticos en la ingeniería de software. Es una descripción completa del comportamiento del sistema que se va a desarrollar. En esta se incluye un conjunto de casos de uso que describen todas las interacciones que tendrán los usuarios con el software.

3.3.1 Requisitos funcionales del sistema (RF)

Son los servicios que proporcionan la herramienta, condición o capacidad que cumplen un objetivo determinado.

No.	Requisito	Descripción
RF1	Crear_proyecto	La herramienta debe permitir crear un nuevo proyecto.
RF2	Cargar_proyecto_existente	La herramienta debe permitir cargar un proyecto previamente creado.
RF3	Eliminar_ proyecto	La herramienta debe permitir eliminar un proyecto existente.
RF4	Crear_CU_del_negocio	La herramienta debe permitir crear casos de uso del negocio.
RF5	Editar_CU_del_negocio	La herramienta debe permitir editar un caso de uso del negocio que da la posibilidad de modificar los datos implícitos en el caso de uso.
RF6	Eliminar_CU_del_negocio	La herramienta debe permitir eliminar un caso de uso del negocio del proyecto.
RF7	Crear_CU_del_sistema	La herramienta debe permitir crear casos de uso del sistema.
RF8	Editar_CU_del_sistema	La herramienta debe permitir editar un caso de uso del sistema que da la posibilidad de modificar los datos implícitos en el caso de uso.
RF9	Eliminar_CU_del_sistema	La herramienta debe permitir eliminar un caso de uso del sistema del proyecto.

RF10	Crear_procesos	La herramienta debe permitir crear procesos.
RF11	Editar_procesos	La herramienta debe permitir editar un proceso que da la posibilidad de modificar los datos implícitos en el proceso.
RF12	Eliminar_procesos	La herramienta debe permitir eliminar un proceso del proyecto.
RF13	Crear_entidad_del_negocio	La herramienta debe permitir crear entidades del negocio.
RF14	Editar_entidad_del_negocio	La herramienta debe permitir editar una entidad del negocio que da la posibilidad de modificar los datos implícitos en la entidad.
RF15	Eliminar_entidad_del_negocio	La herramienta debe permitir eliminar una entidad del negocio del proyecto.
RF16	Crear_requisito_de_software	La herramienta debe permitir crear requisitos de software.
RF17	Editar_requisito_de_software	La herramienta debe permitir editar un requisito de software que da la posibilidad de modificar los datos implícitos en el requisito.
RF18	Eliminar_requisito_de_software	La herramienta debe permitir eliminar un requisito de software del proyecto.
RF19	Crear_reglas_del_negocio	La herramienta debe permitir crear reglas del negocio
RF20	Editar_reglas_del_negocio	La herramienta debe permitir editar una regla del negocio que da la posibilidad de modificar los datos implícitos en la regla.
RF21	Eliminar_reglas_del_negocio	La herramienta debe permitir eliminar una regla del negocio del proyecto.
RF22	Crear_clases	La herramienta debe permitir crear clases
RF23	Editar_clases	La herramienta debe permitir editar una clase que da la posibilidad de modificar los datos implícitos en la clase.
RF24	Eliminar_clases	La herramienta debe permitir eliminar una clase del proyecto.

RF25	Crear_objeto_del_dominio	La herramienta debe permitir crear objetos del dominio
RF26	Editar_objeto_del_dominio	La herramienta debe permitir editar un objeto del dominio que da la posibilidad de modificar los datos implícitos en el objeto.
RF27	Eliminar_objeto_del_dominio	La herramienta debe permitir eliminar un objeto del dominio del proyecto.
RF28	Crear_casos_de_prueba	La herramienta debe permitir crear casos de prueba
RF29	Editar_casos_de_prueba	La herramienta debe permitir editar un caso de prueba que da la posibilidad de modificar los datos implícitos en el caso de prueba.
RF30	Eliminar_casos_de_prueba	La herramienta debe permitir eliminar un caso de prueba del proyecto.
RF31	Crear_dependencias	La herramienta debe permitir crear dependencias entre los artefactos del proyecto.
RF32	Eliminar_dependencias	La herramienta debe permitir eliminar dependencias entre los artefactos del proyecto.
RF33	Mostrar_matriz_de_trazabilidad	La herramienta debe permitir mostrar la matriz de trazabilidad entre los artefactos.
RF34	Mostrar_impacto	La herramienta debe permitir mostrar el impacto del cambio de un elemento.
RF35	Exportar_informe_completo_del_proyecto	La herramienta debe permitir exportar a PDF el informe completo del proyecto.
RF36	Exportar_informe_de_un_elemento	La herramienta debe permitir exportar el informe de un elemento en particular.
RF37	Leer_archivo_XML_de_Visual_Paradigm	La herramienta debe permitir leer un archivo XML de Visual Paradigm posibilitando el trabajo con los elementos que son generados desde ese archivo.
RF38	Crear_nueva_estructura_proyecto	La herramienta debe permitir crear nuevas

		estructura de proyectos posibilitando que se organicen estructuras propias para cada proyecto.
RF39	Crear_elementos_configuración	La herramienta debe permitir crear elementos de configuración.
RF40	Editar_elementos_configuracion	La herramienta debe permitir editar un elemento de configuración que da la posibilidad de modificar los datos implícitos en el elemento.
RF41	Eliminar_elementos_configuracion	La herramienta debe permitir eliminar un elemento de configuración del proyecto.
RF42	Guardar_proyecto	La herramienta debe permitir guardar un proyecto posibilitando que el mismo pueda ser accedido posteriormente.

Tabla 1: Tabla de descripción de requisitos funcionales de software

3.3.2 Requisitos no funcionales del sistema (RNF)

Son las restricciones que afectan a los servicios o funciones del sistema, los cuales pueden ser: la usabilidad, rendimiento y escalabilidad, restricciones de diseño e implementación, software, hardware así como de soporte y mantenimiento. Los requisitos no funcionales definidos para la creación de la herramienta para agilizar el trabajo con la gestión y trazabilidad de requisitos de software se muestran a continuación:

RNF 1 Usabilidad

- La herramienta deberá permitir al usuario gestionar proyectos, artefactos y sus dependencias de manera rápida, sencilla e intuitiva.
- La herramienta deberá permitir visualizar la matriz de trazabilidad y un gráfico con el impacto del cambio de un artefacto en específico.
- La herramienta deberá salvar toda la información en un fichero con extensión (.grs), y posteriormente debe permitir abrir este fichero en la herramienta nuevamente.

RNF 2 Software

- La herramienta deberá correr sobre el sistema operativo Windows, en su versión 7 o superior.

➢ La herramienta deberá correr sobre el sistema operativo Ubuntu, en su versión 11.04 o superior.

RNF3 Hardware

➢ Procesador Intel Core 2 Duo o Dual Core de 1 GHz o superior.

➢ 256 MG de memoria RAM o superior.

➢ Espacio en disco duro de 10 GB o superior.

RNF4 Soporte y mantenimiento

➢ El proyecto encargado del desarrollo de la herramienta deberá brindar servicios de mantenimiento y soporte en caso de ser necesario, además de ser capaz de realizar futuras versiones de la herramienta y en caso de que el cliente lo requiera deberá brindar capacitación al personal.

Una vez realizado el modelo de dominio y determinados lo requisitos del prototipo funcional es posible modelar y desarrollar la herramienta que se desea obtener.

3.4 Definición de los casos de uso del sistema

3.4.1 Actores del sistema

Actor	Descripción
Analista	Representa al usuario que hará uso de la herramienta y tendrá la oportunidad de interactuar con todas las funcionalidades de la aplicación.

Tabla 2: Descripción de actores del sistema

3.4.2 Diagrama de casos de uso del sistema

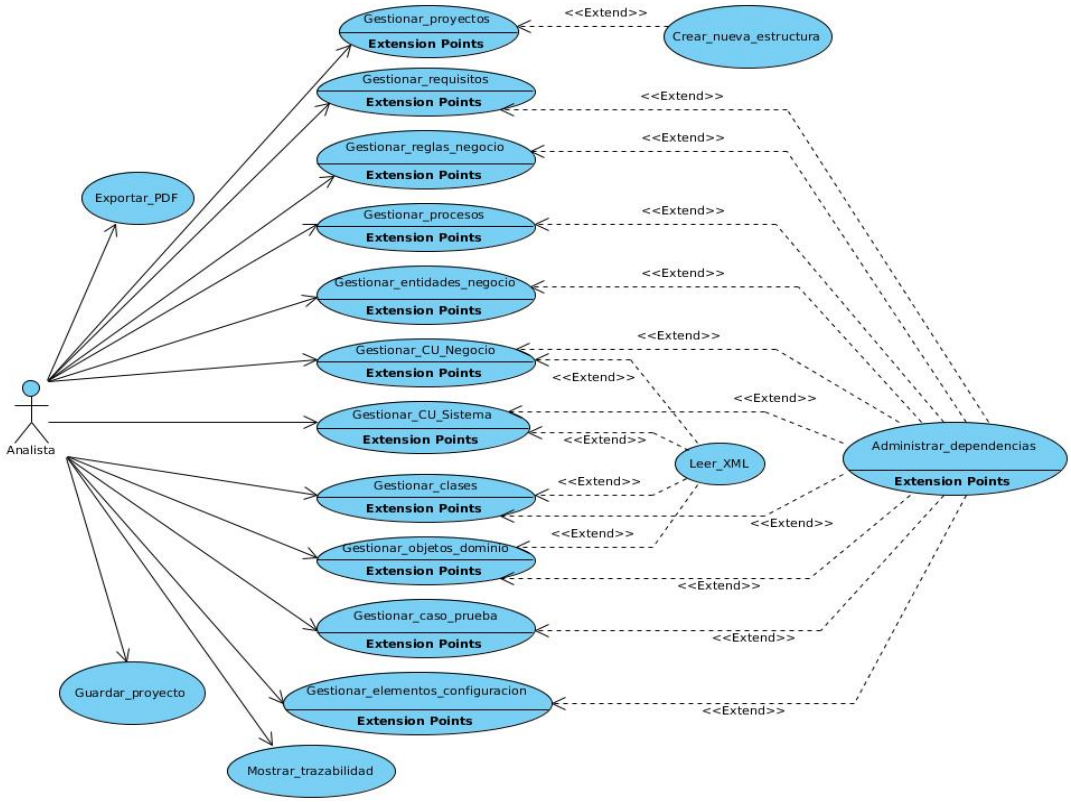

Diagrama 2: Diagrama de casos de uso del sistema

3.4.3 Descripción de los casos de uso del sistema

Descripción del caso de uso Gestionar_proyecto

Objetivo	Gestionar un proyecto determinado.
Actores	Analista: (Inicia) Crea, modifica, carga un proyecto en el sistema y lo cierra.
Resumen	El caso de uso se inicia cuando el actor crea un nuevo proyecto o carga uno existente permitiendo eliminar alguno existente.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Para cargar o actualizar un proyecto, este debe de existir.
Postcondiciones	Queda creado el proyecto.

Flujo de eventos		
Flujo básico: Gestionar un proyecto		
	Actor	**Sistema**
1.	Puede realizar cualquiera de las siguientes acciones: Crear, Abrir o Cerrar un proyecto. Selecciona crear un nuevo proyecto en el menú archivo(A) o en la barra de herramientas superior (Ver Figura 7). Si selecciona Abrir un proyecto. Ver Sección 1: "Abrir proyecto". Si selecciona Cerrar un proyecto. Ver Sección 2: "Cerrar proyecto".	
2.		Muestra un formulario en el cual aparece el campo nombre del proyecto, las opciones: orientado a casos de uso, orientados a procesos o crear una nueva estructura (Ver Figura 8).
3.	Introduce el nombre del proyecto, selecciona la opción orientado a casos de uso u orientado a procesos y presiona el botón "Aceptar".	
4.		Valida los datos y muestra la interfaz principal con la estructura de carpetas orientada a casos de uso, orientado a procesos o con una nueva estructura. (Ver Figura 11)
5.		Termina el caso de uso.
Flujo alterno		
4: Validar datos.		
	Actor	**Sistema**
4.a	No introduce un nombre para el proyecto, introduce un nombre comenzando con	

	números, no selecciona ninguna de las estructuras o introduce un nombre existente para un proyecto.	
5.a.		Muestra una alerta en dependencia del error generado (Ver Figura 9)
Flujo alterno		
3: Cancelar operación.		
	Actor	**Sistema**
3.a	Presiona el botón “Cancelar”.	
4.a.		El formulario se cierra y se muestra la interfaz principal.
Sección 1: “Abrir proyecto”		
Flujo básico: Abrir un proyecto		
	Actor	**Sistema**
1.	Selecciona la opción Abrir proyecto en el menú archivo o en la barra de herramientas superior.	
2.		Muestra una ventana para escoger el archivo en una ubicación determinada (Ver Figura 10).
3.	Escoge el archivo y presiona el botón “Aceptar”.	
4.		Muestra la interfaz principal con la estructura de carpetas y los datos del proyecto. (Ver Figura 11)
Flujo alterno		
3: Existe un proyecto con el mismo nombre.		
	Actor	**Sistema**

3.a.	.	Muestra una alerta si existe un proyecto con el mismo nombre. (Ver Figura 9)
Flujo alterno		
3: Cancelar operación.		
	Actor	**Sistema**
3.a.	Presiona el botón cancelar.	
4.a.		La ventana se cierra y se muestra la interfaz principal.
Sección 2: "Cerrar proyecto"		
Flujo básico: Cerrar un proyecto		
	Actor	**Sistema**
1.	Selecciona un proyecto y selecciona la opción Cerrar proyecto en el menú archivo o en el menú contextual que se muestra al dar click derecho sobre un proyecto. (Ver Figura 12)	
2.		Muestra un mensaje indicando si desea guardar los cambios realizados en el proyecto (Ver Figura 13).
3.	Presiona el botón "Sí".	
4.		Guarda los datos, elimina el proyecto del sistema y muestra la interfaz principal.
Flujo alterno		
3: Selecciona "No".		
	Actor	**Sistema**
3.a	Presiona el botón "No".	

4.a		Elimina el proyecto del sistema y muestra la interfaz principal.
Flujo alterno		
3: Cancelar operación.		
	Actor	**Sistema**
3.a	Presiona el botón cancelar.	
4.a		El mensaje se cierra y se muestra la interfaz principal.

Relaciones	**CU Incluidos**	No existen
	CU Extendidos	Crear_nueva_estructura_proyecto: paso 3 del flujo básico.
Requisitos no funcionales	No existen	
Asuntos pendientes	No existen	

Tabla 3: Descripción del caso de uso Gestionar_proyecto

PROTOTIPO ELEMENTAL DE INTERFAZ GRÁFICA DE USUARIO

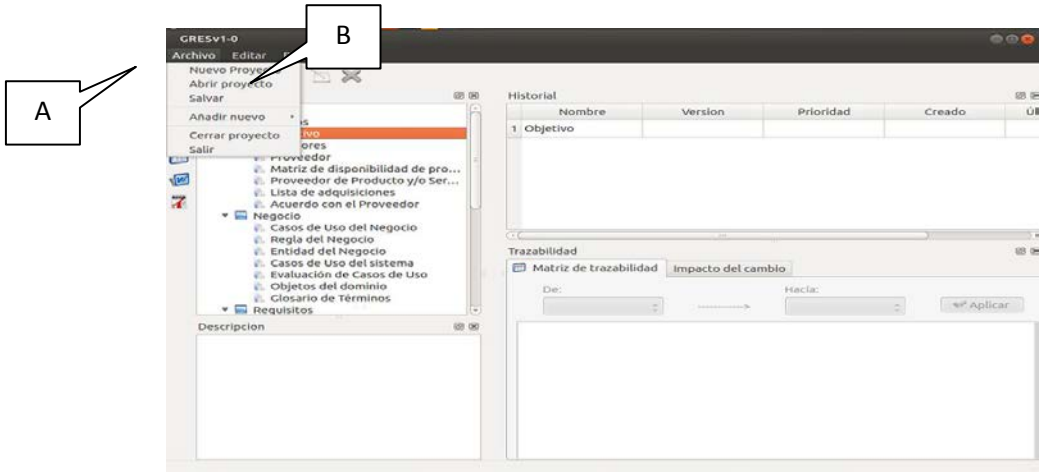

Figura 5: Prototipo de interfaz de usuario para crear un nuevo proyecto

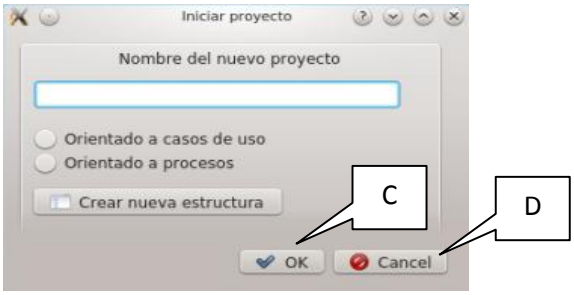

Figura 6: Formulario para iniciar un proyecto

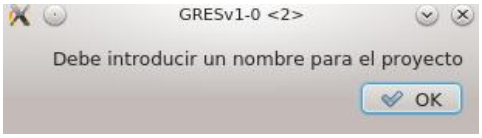

Figura 7: Ejemplo de alerta: Dejar el campo de nombre en blanco

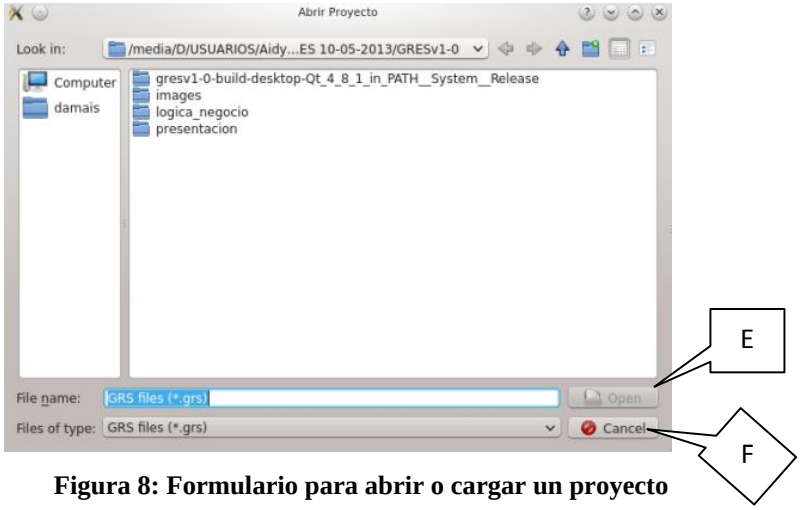

Figura 8: Formulario para abrir o cargar un proyecto

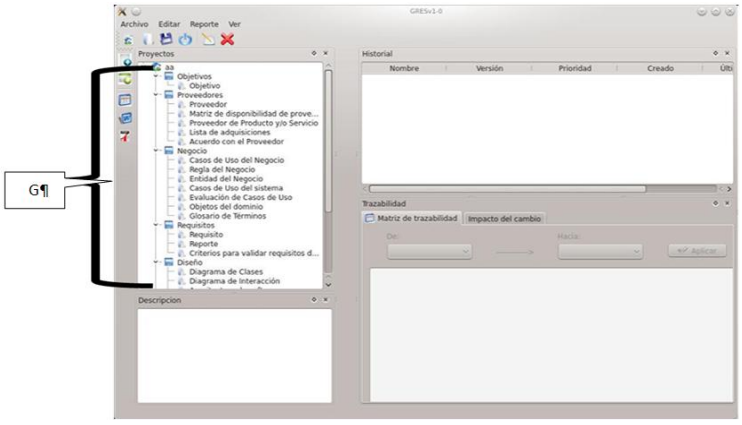

Figura 9: Estructura de las carpetas del nuevo proyecto

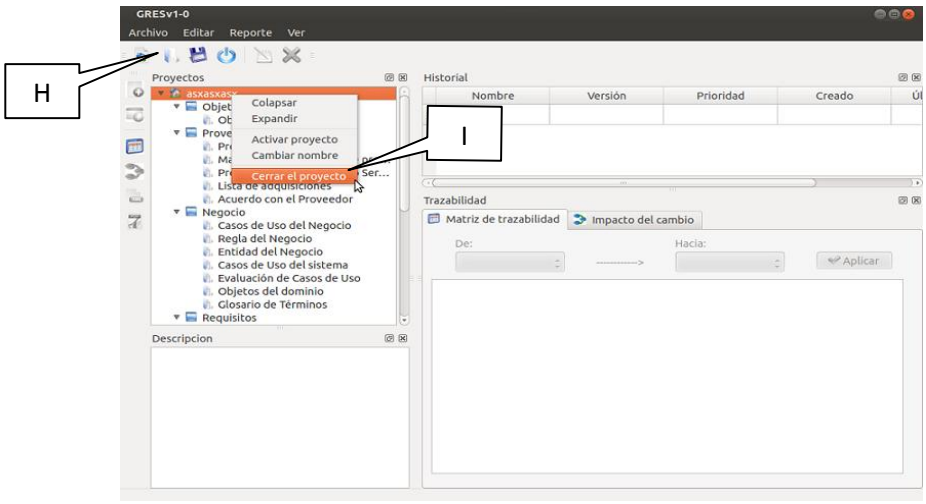

Figura 10: Prototipo de interfaz para cerrar un proyecto

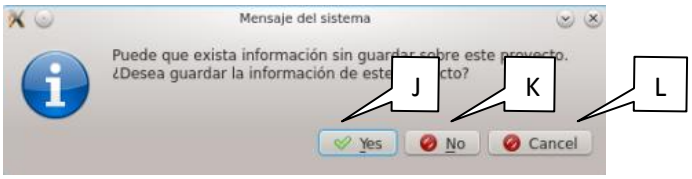

Figura 11: Mensaje del sistema de confirmación para guardar el proyecto

El resto de las descripciones de los casos de uso pueden ser consultadas en el **Anexo 3: Descripciones de los Casos de Uso**.

3.5 Arquitectura del software

La Arquitectura de Software, según la IEEE 1471-2000, define que es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Reynoso, 2013).

La arquitectura de software definida para la realización de la herramienta es N capas, debido a que se enfoca en la distribución de roles y responsabilidades de forma jerárquica, proporcionando una forma más efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. Son los componentes de cada capa los que se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.

A continuación se describen los patrones utilizados en el diseño de la herramienta.

3.5.1 Patrones

Diversas son las definiciones que se brindan en las bibliografías de qué es un patrón de diseño, sin embargo, la mayoría coincide en la definición brindada por Christopher Alexander[5], la cual se toma como término en este trabajo. Él plantea que:

"*cada **patrón** describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez.*"

Una vez expuesto el criterio de Christopher Alexander se considera que un patrón de diseño se define de la siguiente manera:

Un patrón describe un problema que ocurre con reiteración, y a continuación describe la solución y aplicación del mismo, consiguiendo utilizarse en un mismo contexto varias veces sin repetirse ninguna solución. Presenta como objetivos la reducción de tiempo, disminución del esfuerzo de mantenimiento además de aumentar la eficiencia.

3.5.2 Patrones de arquitectura

Los patrones arquitectónicos o patrones de arquitectura ofrecen soluciones a problemas de arquitectura de software en la ingeniería de software. Estos brindan una descripción de los elementos y el tipo de relación que tienen un conjunto de restricciones sobre cómo pueden ser usados. Expresan un esquema de organización estructural esencial para un sistema de software que consta de subsistemas, sus responsabilidades e interrelaciones.

Para el desarrollo de la aplicación se definen dos capas, que tienen como objetivo fundamental separar el diseño de la lógica del negocio. Esto permite una amplia reutilización del código generado, lo que facilita el fraccionamiento de problemas complejos y la localización de errores. A continuación se muestran las capas definidas:

- **Capa de presentación o interfaz de usuario:** esta capa representa la interfaz con que interactuará el usuario y está formada por los formularios y los controles.
- **Capa lógica del negocio:** en esta capa se concentra el código implementado, donde se responde a las solicitudes del usuario, permitiendo automatizar los procesos de negocio que lleva a cabo el usuario.

Dentro del modelo en capas se utilizan solo dos, porque en la herramienta no se almacenan grandes volúmenes de información y no es necesario acceder a datos previamente definidos. El

[5] **Destacado arquitecto reconocido internacionalmente por sus numerosos aportes en la Teoría de Patrones.**

acceso y modificación de la información no se realiza de forma dinámica, ya que ésta se guarda en un fichero y de ser necesaria una modificación; se abre el fichero, se modifica y luego se cierra el mismo donde queda guardado el cambio.

3.5.3 Patrones de diseño

Patrones de software para la asignación de responsabilidades o General Responsibility Assignment Software Patterns (GRASP) que son utilizados en el desarrollo de la herramienta.

- **Experto:** este patrón está diseñado para que la responsabilidad de realizar una función determinada sea de la clase que tiene o puede tener la información requerida. Está presente en la clase controladora CC_Gres y en la clase CE_Proyecto de manera que cada responsabilidad está asignada a la clase que cuenta con la información necesaria para realizarla.
- **Creador:** está diseñado para asignar responsabilidades relacionadas con la creación de objetos. Su intención es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Está presente en la clase CC_Gres que es el encargado de crear instancias de la clase CE_Proyecto y CAD_Acceso_Fichero, igualmente está presente en la clase CE_Proyecto la cual es la encargada de crear instancias de la clase CE_Artefacto.
- **Controlador:** está diseñado para asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades y se encuentra presente en la clase CC_Gres, que es la encargada de atender los eventos del sistema que son generados por un actor externo y se asocian a operaciones del sistema.
- **Alta Cohesión:** se encarga de que las clases del diseño cumplan con las tareas que tienen definidas, aplicando atributos y métodos de manera sencilla para implementar dichas tareas.
- **Bajo Acoplamiento:** se encarga de que las clases del diseño colaboren unas con otras, siempre y cuando esta colaboración se mantenga en un mínimo aceptable, reduciendo el impacto de posibles cambios en la herramienta.
- **Fabricación Pura:** asigna un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa un objeto en el dominio del problema. Se ve reflejado en la clase CE_Aux_Datos, pues esta no representa un objeto asociado al dominio del problema, sino que es ideada por los diseñadores con el fin de dar soporte a una alta cohesión, un bajo acoplamiento y la reutilización de un conjunto de operaciones y atributos que posibilite lograr un mejor diseño y organización de la herramienta a desarrollar.

Patrones Gang of Four (GoF) o La Banda de los cuatro, en el caso particular de la herramienta que se desarrolla es utilizado el Singleton o instancia única.

- **Singleton (Instancia única):** garantiza la existencia de una única instancia para una clase.

En este caso se puede identificar este patrón en la clase controladora CC_Gres cuando crea una única instancia de la clase CAD_Acceso_Fichero.

3.6 Diagramas de interacción

Los diagramas de interacción describen el comportamiento de un sistema, para demostrar cómo los objetos interactúan dinámicamente en diferentes momentos durante la ejecución del sistema. Los objetos dentro de un sistema se comunican unos con otros, enviándose mensajes. Un mensaje es justo una operación donde un objeto llama a otro objeto. Así pues la dinámica de un sistema se refiere a cómo los objetos dentro del sistema cambian de estado durante el ciclo de vida del mismo y también a cómo dichos objetos colaboran a través de la comunicación (Otero, 2007).

Los diagramas de interaccion generados durante el desarrollo de la herramienta se pueden observar en el **Anexo 4: Diagramas de interacción.**

3.7 Diseño

El diseño es técnicamente la parte central de la ingeniería de software. Durante el diseño se desarrollan, revisan y documentan los refinamientos progresivos de las estructuras de datos, de la estructura del programa y los detalles procedimentales. Da como resultado representaciones cuya calidad puede ser evaluada.

Durante el diseño, se usa el diagrama de clases del análisis y se modifica para satisfacer los detalles de las implementaciones.

3.7.1 Diagrama de clases del diseño

Los diagramas de clases del diseño son una asignación de responsabilidades inicial. Se utilizan para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, composición, agregación y herencia. Es la creación de la primera imagen ingenieril de cómo quedará la implementación del prototipo funcional. A continuación se muestran los diagramas de clases del diseño correspondiente a la herramienta.

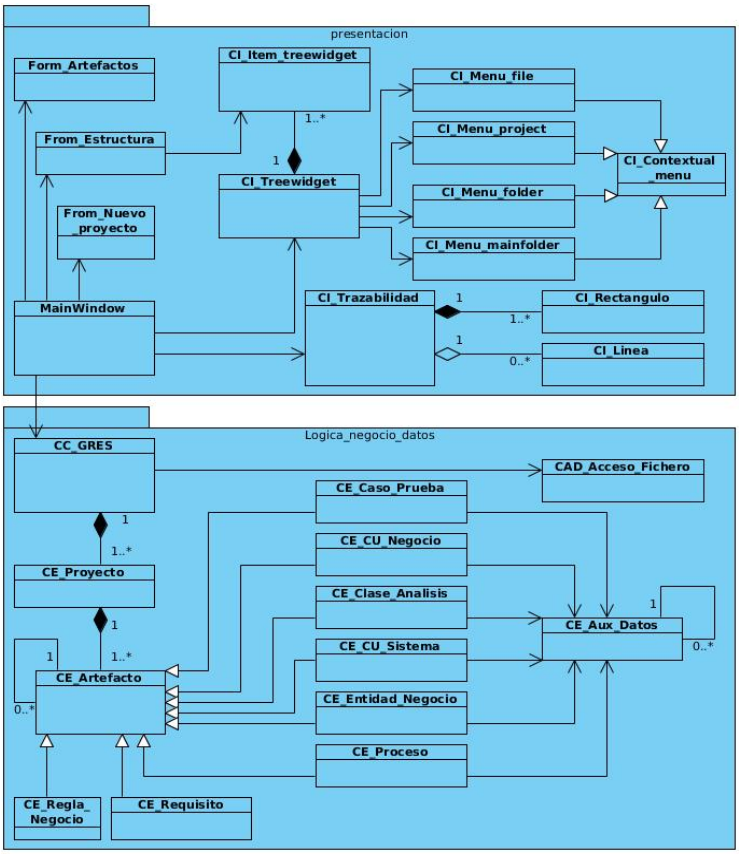

Diagrama 3: Diagrama de clases del diseño

La descripción de cada una de las clases con sus atributos y operaciones se pueden observar en el **Anexo 5: Clases del diseño**.

Una vez mostrados los diagramas de clases del diseño, se muestra a continuación una descripción general de cada una de las clases presentes en el diseño.

Nombre de la clase	**Tipo de clase**	**Descripción**
MainWindow	Interfaz	Se encarga de conectar la capa de presentación y la capa de lógica del negocio y controlar las acciones del usuario sobre los elementos visuales.
Form_Nuevo_proy	Interfaz	Se encarga de recopilar y validar los datos para crear un proyecto.

Form_Artefactos	Interfaz	Se encarga de gestionar los datos para crear un artefacto.
Form_Estructura	Interfaz	Se encarga de crear y visualizar estructuras de proyectos.
CI_Trazabilidad	Interfaz	Se encarga de crear una escena en la interfaz principal.
CI_Rectangulo	Interfaz	Se encarga de crear un rectángulo en la escena creada por la interfaz CI_Trazabilidad en la interfaz principal.
CI_Linea	Interfaz	Se encarga de crear líneas para conectar cuadrados creados por la interfaz CI_Rectangulo.
CI_Treewidget	Interfaz	Se encarga de crear una estructura en forma de árbol en la interfaz principal.
CI_Item_treewidget	Interfaz	Se encarga de crear los elementos de la estructura en forma de árbol de la interfaz CI_Treewidget.
CI_Contextual_menu	Interfaz	Es la clase general de la cual heredan cada uno de los menús contextuales.
CI_Menu_project	Interfaz	Se encarga de crear un menú contextual para los elementos que son proyectos en la estructura en forma de árbol de la interfaz CI_Treewidget.
CI_Menu_mainfolder	Interfaz	Se encarga de crear un menú contextual para los elementos que son carpetas principales, contenidos dentro de elementos que son proyecto en la estructura en forma de árbol de la interfaz CI_Treewidget.

CI_Menu_folder	Interfaz	Se encarga de crear un menú contextual para los elementos que son carpetas, contenidos dentro de elementos que son carpetas principales en la estructura en forma de árbol de la interfaz CI_Treewidget.
CI_Menu_file	Interfaz	Se encarga de crear un menú contextual para los elementos que son archivos, contenidos dentro de elementos que son carpetas en la estructura en forma de árbol de la interfaz CI_Treewidget.
CC_Gres	Controladora	Se encarga de controlar todas las acciones realizadas por el usuario, distribuyendo en dependencia de la funcionalidad, las obligaciones al resto de las clases del negocio.
CE_Proyecto	Controladora	Es la encargada de manejar todas las operaciones sobre el flujo de datos de los proyectos y las operaciones sobre los artefactos contenidos dentro de los mismos.
CE_Artefacto	Controladora	Esta clase se encarga de controlar el flujo de datos general de cada uno de los artefactos así como las operaciones sobre sus atributos.
CE_Requisito	Controladora	Esta clase se encarga de controlar el flujo de datos de cada uno de los requisitos definidos por el usuario, así como las operaciones sobre sus atributos.
CE_Regla_Negocio	Controladora	Esta clase se encarga de controlar el flujo de datos de cada una de las reglas del negocio definidas por el usuario, así como las operaciones sobre sus atributos.

CE_Caso_Prueba	Controladora	Esta clase se encarga de controlar el flujo de datos de cada uno de los casos de prueba definidos por el usuario, así como las operaciones sobre sus atributos.
CE_Clase_Analisis	Controladora	Esta clase se encarga de controlar el flujo de datos de cada una de las clases definidas por el usuario, así como las operaciones sobre sus atributos.
CE_Entidad_Negocio	Controladora	Esta clase se encarga de controlar el flujo de datos de cada una de las entidades del negocio definidas por el usuario, así como las operaciones sobre sus atributos.
CE_Proceso	Controladora	Esta clase se encarga de controlar el flujo de datos de cada uno de los procesos definidos por el usuario, así como las operaciones sobre sus atributos.
CE_CU_Sistema	Controladora	Esta clase se encarga de controlar el flujo de datos de cada uno de los casos de usos del sistema definidos por el usuario, así como las operaciones sobre sus atributos.
CE_CU_Negocio	Controladora	Esta clase se encarga de controlar el flujo de datos de cada uno de los casos de usos del negocio definidos por el usuario, así como las operaciones sobre sus atributos.
CE_Aux_Datos	Controladora	Esta clase es auxiliar y se utiliza para dar soporte y reutilización de atributos y funcionalidades al resto de las clases que así lo requieran.
CAD_Acceso_Fich	Controladora	Se encarga de controlar todas las acciones

ero		relacionadas con guardar un proyecto en un fichero o abrirlo desde un fichero.

Tabla 4: Descripción general de cada una de las clases presentes en el diseño

3.8 Conclusiones del capítulo

En este capítulo se describieron los principales conceptos asociados al dominio del problema y su relación entre ellos, detallando cómo se desarrollan los procesos de gestión y trazabilidad de requisitos para la herramienta a crear. Se brindó una clara definición del actor determinado que interactuará con la herramienta, así como el levantamiento de los requisitos funcionales y no funcionales con los cuales debe cumplir la misma. Se seleccionó la arquitectura N capas para una mejor distribución entre las capas delimitadas, garantizando que cada una cumpla con sus responsabilidades. Además se describieron los patrones utilizados y las clases del diseño, que permitieron una mejor comprensión de la estructura de la herramienta a desarrollar.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En este capítulo se realiza el flujo de trabajo de implementación, en los que se observan los principales artefactos como el Modelo de Implementación, que incluye los diagramas de despliegue y componentes. Se describen los estilos de codificación utilizados, así como una descripción de los principales algoritmos presentes en la implementación de la herramienta. Una vez concluida la implementación se realizarán las pruebas funcionales a la herramienta.

4.2 Modelo de implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes, y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios o ejecutables. Además se deben hacer los tests de unidad donde cada implementador es responsable de testear las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable (EcuRed, 2010).

4.2.1 Implementación

Es el proceso que asegura la operatividad del sistema de información y que permite al usuario obtener beneficios por su operación. La fase de implementación es donde se define la organización del código en términos de subsistemas estructurados en capas, se definen las clases y objetivos en términos de componentes (código fuente, ejecutables, bases de datos).

4.3 Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado del Modelado (UML) que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Se modela la topología del hardware sobre el que se ejecuta el sistema y se describe la distribución física del prototipo funcional. A continuación se muestra el diagrama de despliegue de la herramienta GRES v1.0

Diagrama 4: Diagrama de despliegue

La estación de trabajo donde se desplegará la herramienta debe poseer las siguientes características:

- Sistema multiplataforma, por tanto se utilizará en cualquier sistema operativo.
- Procesador Intel Core 2 Duo o Dual Core de 1 GHz o superior.
- 256 MB de memoria RAM o superior.
- Espacio en disco duro de 40 GB.

4.4 Diagrama de componentes

Un diagrama de Componentes permite modelar la estructura del software y la dependencia entre componentes. Un componente es un grupo de clases que trabajan estrechamente relacionadas entre sí que pueden ser código fuente, binario o ejecutable. Una relación de dependencia indica que un componente utiliza otro, por lo cual depende del mismo.

A continuación se muestra el diagrama de componente de código fuente correspondiente a la herramienta GRES v1.0:

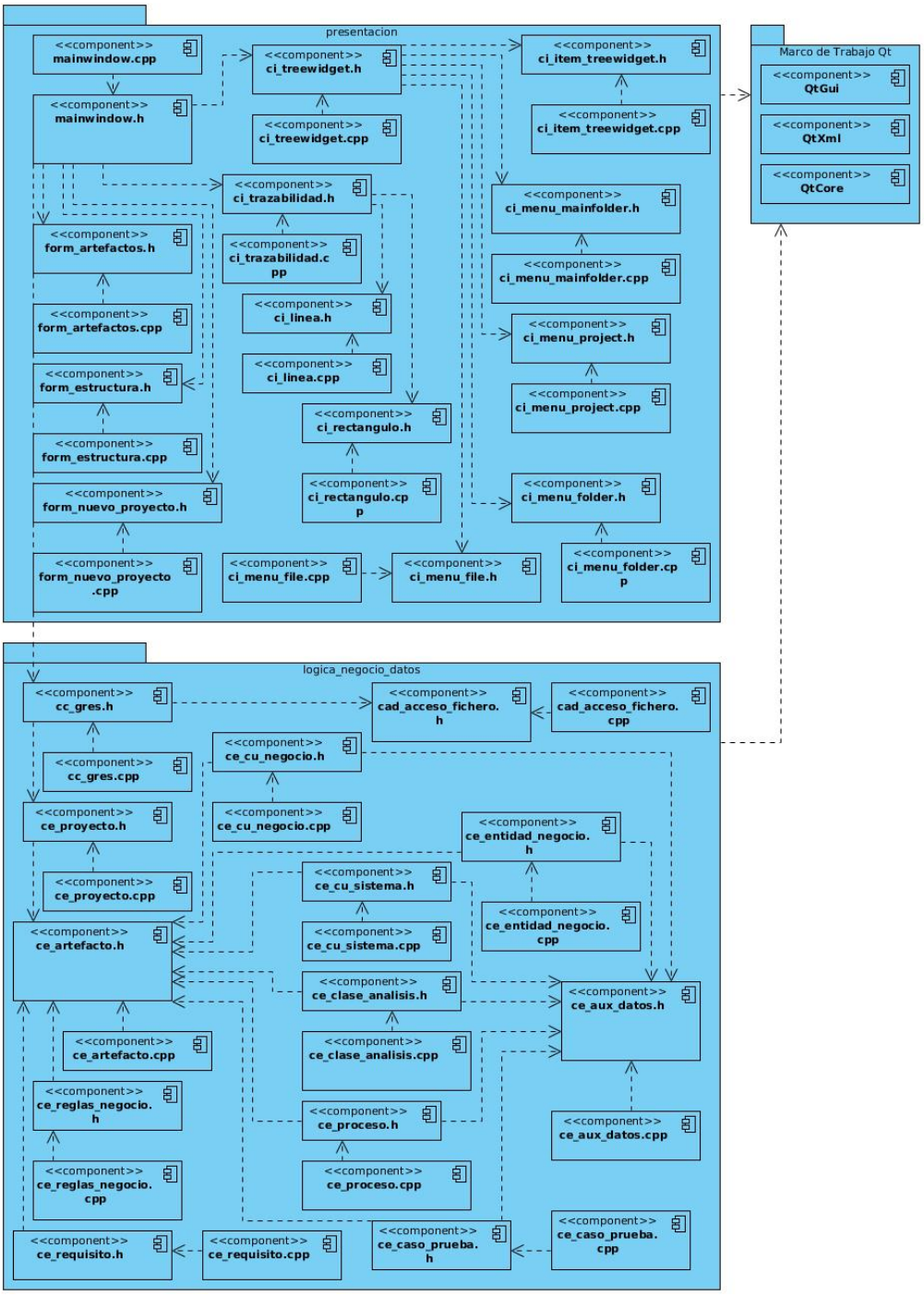

Diagrama 5: Diagrama de componentes de código fuente

4.5 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a la estructura y apariencia física del código. Un estándar de codificación completo comprende todos los aspectos de la generación de código. Usar una técnica de codificación sólida

y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para la calidad del software y para obtener un buen rendimiento ya que convierte al software en un sistema fácil de comprender y de mantener.

4.5.1 Estilo de codificación utilizado

Los estilos de codificación describen convenciones para escribir el código fuente del software. Estos se utilizan de manera uniforme durante toda la implementación de la herramienta para facilitar la lectura y comprensión del código para aquellas personas que deseen efectuar posteriores mejoras a la versión realizada.

- Las variables del mismo tipo de dato se declaran en una sola línea.
- Se añaden comentarios descriptivos en el código de tipo aclaratorios, sólo cuando es necesario.
- Todos los nombres de las variables declaradas se escriben con letra minúscula, si poseen dos o más palabras se separan mediante un guión bajo.
- Los nombres de los métodos comienzan con letra mayúscula, si existe dos o más palabras en el nombre del método se separan mediante un guión bajo y el resto de las palabras comienzan con minúscula o mayúscula según convenga.
- Los bloques de código siempre deben estar encerrado entre llaves, incluso si solo constan de una línea de código.
- Los identificadores de variables y métodos deben tener un nombre significativo para que por su simple lectura, pueda conocerse su función.
- La llave “{” que comienza un bloque de código, no debe aparecer a continuación de la declaración.
- La llave que cierra el bloque de código debe estar sola en una línea en correspondencia a la llave “{” que abre el bloque.

4.6 Descripción de los principales algoritmos implementados

Nombre de la clase	Método	Descripción
CC_Gres	Guardar_proyecto(dirección, proyecto)	Este método se mueve por todos los elementos contenidos dentro de un proyecto, recopilando información de los mismos y

		creando etiquetas con la información recopilada. Luego todas estas etiquetas con la información son almacenadas en un fichero con extensión (.grs) mediante una llamada al método que guarda el fichero contenido dentro de la clase acceso_fichero.
CC_Gres	Abrir_proyecto(dirección)	Se encarga de procesar el fichero que obtiene del método contenido en la clase CAD_Acceso_Fichero. El mismo recopila las etiquetas creadas dentro del fichero y con la información obtenida de cada una de ellas crea un proyecto con toda la información contenida dentro del mismo.
CE_Proyecto	Buscar_elemento(elemento, padre)	Realiza una búsqueda recursiva sobre los elementos de un proyecto y devuelve el elemento buscado. En la variable padre, entrada por parámetro como referencia, se guarda adicionalmente el nombre del elemento que contiene al elemento buscado.
CE_Proyecto	Cargar_Artefactos_XML(elemento, direccion)	Carga un fichero XML generado previamente por Visual Paradigm y crea una serie de elementos que posteriormente añadirá al elemento que se le pasa por parámetro.

CC_Proyecto	Add_elemento(nuevo, elemento, lista_artefactos)	Recibe como parámetros un nuevo elemento, un elemento a buscar y una lista de elementos sobre la cual realizará una búsqueda recursiva del elemento en cuestión para agregarle el nuevo elemento.

Tabla 5: Descripción de los principales algoritmos implementados

4.7 Pruebas

Consisten en una dinámica de la verificación del comportamiento de un conjunto finito de casos de prueba contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo.

La prueba de software es un elemento crítico para la garantía del correcto funcionamiento del software, presentando como objetivos: detectar defectos en el software, verificar la integración adecuada de los componentes, verificar que todos los requisitos se han implementado correctamente, identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente, diseñar casos de prueba que sistemáticamente saquen a la luz diferentes errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

4.7.1 Pruebas de Caja Negra

Para la aplicación de pruebas a la herramienta se seleccionó la prueba de caja negra. Estas se centran en los requisitos funcionales del software. Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa, según lo que se desea hallar (un error o probar una funcionalidad). Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. Muchos autores consideran que estas pruebas permiten encontrar: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a las bases de datos externas, errores de rendimiento y errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

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

- ***Técnica del Análisis de Valores Límites:*** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ***Técnica de Grafos de Causa-Efecto:*** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En el presente trabajo se utilizó específicamente, del método de Caja negra, ***la Técnica de Partición de Equivalencia,*** siendo la misma una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

Para ello se diseñaron casos de pruebas a través de la descripción de los casos de uso con el fin de descubrir errores del software. A continuación se muestran el diseño de casos de pruebas para el caso de uso Gestionar_proyecto. El resto de los diseños de casos de pruebas pueden ser consultados en el **Anexo 3: Diseños de Casos de Pruebas**.

Descripción del caso de prueba Gestionar_Proyecto

Descripción general del caso de uso

El caso de uso inicia cuando el usuario selecciona la opción crear o cargar un proyecto, inserta los datos solicitados y los guarda de ser correctos. El sistema guarda los datos insertados terminando así el caso de uso.

Condiciones de ejecución

Para cargar o eliminar un proyecto, este debe de existir.

SC1 Crear proyecto

Escenario	**Descripción**	**Nombre del nuevo proyecto**	**Estructura**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear un proyecto correctamente.	El sistema muestra un formulario en el cual aparece el campo nombre del proyecto, las opciones para la	V	N/A	Guarda los datos y crea el nuevo proyecto	Barra de menú/ submenú
		Nombre del proyecto	Estructura del proyecto		

<table>
<tr><td></td><td>estructura del proyecto: orientado a casos de uso, orientados a procesos o crear una nueva estructura. Comprueba que los datos sean correctos y crea el nuevo proyecto.</td><td></td><td></td><td>con su estructura de carpetas.</td><td>"Archivo"/ opción "Nuevo proyecto".</td></tr>
<tr><td rowspan="10">EC 1.2 Crear un proyecto incorrect a-mente.</td><td rowspan="10">Si al insertar los datos para crear el proyecto no introduce un nombre para el proyecto, introduce un nombre comenzando con números, introduce un nombre existente para un proyecto o no selecciona ninguna estructura. Se envían mensajes de alerta en dependencia del error generado. Si no introduce un nombre para el proyecto (el sistema muestra un mensaje indicando que no debe dejar el campo en blanco), si introduce un nombre comenzando con números (el sistema muestra un mensaje indicando que no debe introducir un nombre comenzando con números), si introduce</td><td>I</td><td>V</td><td rowspan="10">No se guardan los datos y no se crea el nuevo proyecto.</td><td rowspan="10">Barra de menú/ submenú "Archivo"/opción "Nuevo proyecto".</td></tr>
<tr><td>Nombre del proyecto vacío</td><td>Estructura del proyecto</td></tr>
<tr><td>I</td><td>V</td></tr>
<tr><td>Nombre del proyecto vacío.</td><td></td></tr>
<tr><td>I</td><td>V</td></tr>
<tr><td>Nombre del proyecto existente.</td><td>Estructura del proyecto.</td></tr>
<tr><td>V</td><td>I</td></tr>
<tr><td>Nombre del proyecto correcto.</td><td>Estructura del proyecto no seleccionada.</td></tr>
</table>

	un nombre existente (el sistema muestra un mensaje indicando que ya existe el nombre del proyecto) y si no selecciona ninguna estructura para el proyecto (el sistema muestra un mensaje indicando que debe seleccionar una estructura para el proyecto).				
EC 1.3 Crear proyecto. Cancelar.	El sistema cancela la operación de crear un proyecto.	N/A Nombre del proyecto.	N/A Nombre del proyecto.	Cancela la operación y cierra la ventana de crear el proyecto.	Barra de menú/ submenú "Archivo"/ opción "Nuevo proyecto"/ botón "Cancelar".
EC 1.4 Crear proyecto. Cerrar.	El sistema cierra la operación de crear un proyecto.	N/A Nombre del proyecto.	N/A Nombre del proyecto.	Cierra la ventana de crear el proyecto.	Barra de menú/ submenú "Archivo"/ opción "Nuevo

					proyecto"/ botón "Cerrar".

Tabla 6: Caso de prueba del CU Gestionar_Proyecto (SC1 Crear proyecto)

SC2 Abrir proyecto

Escenario	Descripción	Nombre del nuevo proyecto	Tipo	Respuesta del sistema	Flujo central
EC 2.1 Abrir un proyecto correctamente.	El sistema muestra un formulario que brinda la opción de buscar un archivo en cualquier dirección donde esté ubicado además del campo nombre del proyecto que contenga la extensión (.grs).	V	V	Se activa el botón "Abrir" y al dar click encima de este se carga el proyecto existente.	Barra de menú/ submenú "Archivo" / opción "Abrir proyecto".
		Nombre del proyecto	Extensión del proyecto		
EC 2.2 Abrir un proyecto incorrectamente.	El sistema muestra un formulario que brinda la opción de buscar un archivo y no se carga o no elige el archivo con extensión (.grs) entonces no se carga el proyecto.	N/A	N/A	No se activa el botón "Abrir".	Barra de menú/ submenú "Archivo" / opción "Abrir proyecto".
		Nombre del proyecto vacío	Extensión del proyecto		
		V	N/A		
		Nombre del proyecto vacío.	Extensión del proyecto		
EC 2.3 Abrir proyecto. Cancelar.	El sistema cancela la operación de abrir un proyecto.	N/A	N/A	Cancela la operación y cierra la ventana	Barra de menú/ submenú "Archivo"
		Nombre del proyecto.	Nombre del		

			proyecto.	de crear el proyecto.	/ opción "Abrir proyecto "/ botón "Cancelar".
EC 2.4 Abrir proyecto. Cerrar.	El sistema cierra la operación de abrir un proyecto.	N/A Nombre del proyecto.	N/A Nombre del proyecto.	Cierra la ventana de crear el proyecto.	Barra de menú/ submenú "Archivo" / opción "Abrir proyecto "/botón "Cerrar".

Tabla 7: Caso de prueba del CU Gestionar_Proyecto (SC2 Abrir proyecto)

SC3 Cerrar Proyecto

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Cerrar un proyecto correctamente.	Al seleccionar la opción de cerrar un proyecto en el menú archivo o en el menú contextual que se despliega al dar click derecho sobre un proyecto, el sistema muestra el siguiente mensaje (Puede que exista información sin guardar sobre este proyecto. ¿Desea guardar la información de este proyecto?, Si, No y Cancelar). Si escoge la opción "Sí" que desea guardar la información, el sistema muestra una nueva	Muestra una nueva ventana que posibilita guardar el proyecto en una dirección seleccionada por el usuario.	Barra de menú/ submenú "Archivo"/ opción "Cerrar proyecto" o en el Menú contextual/ opción "Cerrar proyecto"/ botón "Sí".

	ventana para guardar el proyecto en la dirección deseada, se guarda el proyecto y se elimina del sistema.		
EC 3.2 Cerrar un proyecto incorrectamente.	Al seleccionar la opción de cerrar un proyecto en el menú archivo o en el menú contextual que se despliega al dar click derecho sobre un proyecto, no se selecciona el proyecto a cerrar.	Muestra un mensaje indicando que debe seleccionar el proyecto que desea cerrar.	Barra de menú/ submenú "Archivo"/ opción "Cerrar proyecto" o en el Menú contextual/ opción "Cerrar proyecto".
EC 3.3 Presiona el botón "No".	Si escoge "No", se cierra el proyecto y se elimina del sistema.	Cierra el mensaje y muestra la interfaz principal.	Barra de menú/ submenú "Archivo"/ opción "Cerrar proyecto" o en el Menú contextual/ opción "Cerrar proyecto"/ botón "No".
EC 3.4 Presiona el botón "Cancelar".	El sistema cierra la operación de cerrar un proyecto.	Cierra la ventana de cerrar el proyecto y se muestra la interfaz principal.	Barra de menú/ submenú "Archivo"/ opción "Cerrar proyecto" o en el Menú contextual/ opción "Cerrar proyecto"/ botón "Cancelar".

Tabla 8: Caso de prueba del CU Gestionar_Proyecto (SC3 Cerrar proyecto)

Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre del nuevo proyecto	Campo de texto	No.	Acepta cualquier carácter pero debe comenzar con letras, nunca con números.

2	Estructura	Campo de selección	No.	Debe estar seleccionada cualquier opción.

Tabla 9: Descripción de las variables del caso de prueba Gestionar_proyecto

Durante la fase de prueba en su primera iteración se encontraron un total de 45 no conformidades. En una segunda iteración se verificaron las no conformidades anteriores, que fueron corregidas por el equipo de desarrollo y se arrojaron 5 nuevas no conformidades, las cuales fueron corregidas inmediatamente, quedando así el producto listo para su liberación y posterior uso por los analistas del Centro GEYSED.

El resto de las descripciones de los casos de uso pueden ser consultadas en el **Anexo 6: Descripciones de los Casos de Prueba**.

4.7.2 Pruebas de Aceptación

Estas pruebas las realiza el cliente. Son pruebas funcionales sobre el sistema completo y buscan una cobertura de la especificación de requisitos y del manual del usuario. Las mismas no se realizan durante el desarrollo, pues sería impresentable de cara al cliente, sino una vez concluidas todas las pruebas de integración por parte del desarrollador.

Para la realización de estas pruebas se tuvieron en cuenta las técnicas denominadas "pruebas alfa" y "pruebas beta". Las pruebas alfa se pusieron en práctica con la invitación del cliente al entorno de desarrollo a probar el sistema. Se trabajó en un entorno controlado y el cliente siempre tuvo un experto a mano para ayudarle a usar el sistema y para analizar los resultados.

Las pruebas beta se realizaron después de las pruebas alfa, y se desarrollaron en el entorno del cliente, un entorno que estuvo fuera de control de los desarrolladores. El cliente se quedó a solas con el producto y trató de encontrarle fallos (reales o imaginarios) de los que informó al desarrollador.

Una vez aplicadas las pruebas de aceptación, se arrojaron un total de 10 no conformidades, las cuales fueron corregidas inmediatamente por el equipo de desarrollo.

Las pruebas alfa y beta son habituales en productos que se van a vender o ser utilizados por muchos clientes, evidenciándose en el caso particular la herramienta GRES v1.0, pues la misma será utilizada por todos los analistas del centro GEYSED. Algunos de los potenciales compradores o usuarios finales, se prestan a estas pruebas para ir entrenando a su personal con tiempo, lo cual ha sido efectivo para los analistas pues les brinda la ventaja de mediante el uso de la herramienta familiarizarse con la misma.

4.8 Comparación entre las herramientas OSRMT y GRES

A continuación se muestra una tabla comparativa entre las herramientas OSRMT y GRES en dependencia de sus características y criterios emitidos por los analistas del centro GEYSED según las funcionalidades de las mismas. Este análisis se ha realizado con el fin de comprobar si la herramienta obtenida cumple con las expectativas de dichos analistas y se acerca en la medida de lo posible a las herramientas reales existentes.

Para un mejor entendimiento de la misma se expone como leyenda la siguiente:

- Ausente (A): si algunas de las pautas para la comparación no está presente en cualquiera de las herramientas analizadas.
- Medio (M): si cumple con la pauta planteada pero no en su totalidad, ni de la manera más factible.
- Elevada (E): si cumple en su totalidad con la pauta en cuestión.

Herramientas de gestión y trazabilidad de requisitos			
Requisitos		**OSRMT**	**GRES**
	Requisitos funcionales		
	Se puede trabajar en múltiples proyectos a la vez.	A	M
	Permite creación de Informes PDF.	M	E
	Exporta documentos generados.	M	E
	Permite guardar y cargar la información generada desde un archivo.	A	E
	Se realiza el proceso de crear dependencias entre los artefactos de forma factible.	M	E
	Permite crear otras estructuras para proyectos o personalizar las existentes.	M	E
	Importa datos desde diferentes formatos de fichero.	A	E

	Requisitos no funcionales		
	Posee buen rendimiento y realiza las operaciones de manera rápida y satisfactoria.	M	E
	Es multiplataforma.	E	E
	Interfaz amigable e intuitiva.	M	E
	Genera un archivo con una extensión propia que contiene la información de proyectos y permite cargarlo posteriormente en el sistema.	A	E
	Trazabilidad		
	Gestión de la trazabilidad entre requisitos.	E	E
	Permite visualizar un gráfico de las dependencias.	E	E
	Permite generar documentos de trazabilidad del proyecto.	A	M
	Muestra la matriz de trazabilidad y el tipo de matriz.	M	E
	Gestión de requisitos		
	Permite la captura de requisitos.	E	E
	Gestiona Casos de Uso.	M	E
	Gestiona artefactos del negocio.	M	E
	Gestiona elementos de diseño.	M	E
	Gestiona casos de prueba.	M	E
	Gestiona elementos de configuración.	M	E

	Mantenimiento y soporte		
	Servicio de mantenimiento y soporte por parte del equipo de desarrollo o entidad productora.	A	M
	Manual de usuario.	A	E
	Capacitación del personal que interactúa con la herramienta.	A	E
	Reportes		
	Muestra informe con vista a presentarle al cliente el desarrollo alcanzado hasta el momento.	A	E
	Muestra informe para el usuario que interactúa con la herramienta.	M	E

Tabla 10: Tabla comparativa entre OSRMT y GRES

4.9 Conclusiones del capítulo

En este capítulo se realizó una descripción de la implementación. Luego de aplicar estándares de codificación y buenas prácticas de programación definidos por el equipo de desarrollo, se obtuvo como resultado la herramienta Gestor de Requisitos de Software (GRES). A la misma se le aplicaron las pruebas funcionales, garantizando el cumplimiento de los requisitos definidos por el cliente.

Se concluye además que la realización del diagrama de componentes, permitió comprender la interacción entre los componentes que conforman la herramienta y mostrar las dependencias que existen entre los mismos. Con la realización de las pruebas y sus resultados satisfactorios unidos a una comparación con respecto a la herramienta OSRMT, se puede concluir que el software no presenta ningún inconveniente, por lo que su etapa de desarrollo fue totalmente eficiente.

CONCLUSIONES

Durante el desarrollo de la presente investigación se ha arribado a las siguientes conclusiones:

- Se determinaron las deficiencias existentes en el proyecto a la hora de trabajar con la trazabilidad y se identificaron los principales objetivos del proyecto a los cuales no se les da solución.

- Con la utilización de la herramienta desarrollada, el tiempo y esfuerzo que conllevaba el trabajo en la gestión de requisitos de software por parte de los analistas del centro, se logrará realizar con mayor rapidez y menor esfuerzo. El trabajo con la misma posibilitará que el proceso de gestión y trazabilidad de requisitos de software se torne menos engorroso, elemento indispensable para poder obtener productos con mayor calidad. Por otra parte se logra una mayor organización de la información de los proyectos que se desarrollan dentro del centro, trayendo consigo la eliminación de información redundante. La solución informática desarrollada, hace que el trabajo de los analistas se torne más simple, y que sean más reales los resultados que se obtengan a partir de su uso.

- Por todo lo planteado con anterioridad se puede decir que con la herramienta Gestor de Requisitos de Software (GRES v1.0) se logra mejorar en gran medida el control de la información referente a los proyectos productivos en el centro de desarrollo GEYSED.

- Los artefactos generados en el proceso de desarrollo de la herramienta servirán para que otros desarrolladores obtengan un mejor entendimiento de su estructura, facilitando la realización de posibles versiones o la agregación de nuevas funcionalidades a la misma.

- Tiene gran importancia el Gestor de Requisitos de Software como una herramienta automática de gestión y trazabilidad de requisitos de software, ya que ella ofrece un punto de apoyo para los analistas del centro GEYSED y la posibilidad de realizar el proceso de gestión de requisitos con mayor facilidad y menos esfuerzo que si se realizara de manera manual.

RECOMENDACIONES

Como resultado del desarrollo y conclusiones de la investigación realizada, se hacen las siguientes recomendaciones con el fin de proveer a la herramienta de futuras mejoras.

- Estimular y promover el continuo desarrollo de la herramienta, agregándole nuevas funcionalidades y refinando las ya existentes para posteriores versiones, además de abarcar más en la recopilación de datos de manera que se pueda agregar como complemento de la trazabilidad, la funcionalidad de poder calcular el impacto de la trazabilidad.

- Se recomienda agregar la posibilidad de que la herramienta pueda cargar desde un fichero XML el formato del reporte que se desee generar, para garantizar que esta funcionalidad no quede obsoleta si en un futuro se realizan cambios en las plantillas que actualmente se utilizan en el centro.

- Reestructurar la herramienta de manera tal que toda la información pueda estar centralizada en un servidor central donde todos los analistas puedan almacenar su información sobre el trabajo realizado y puedan colaborar entre sí para un mejor desempeño del trabajo en equipo.

- Continuar el estudio del tema abordado, ampliando el uso de los elementos teóricos que sustentan la investigación y aplicándolos en la herramienta.

REFERENCIAS BIBLIOGRÁFICAS

Alemany Garrido, Salvador;. 2009. Introducción a Qt., Programación gráfica en C++ con Qt4. [aut. libro] Salvador Alemany Garrido. *Introducción a Qt., Programación gráfica en C++ con Qt4.* 2009.

Álvarez Cárdenas, Ing. Sofía. 2001. Gestión de Requerimientos para Progrmación a Distancia. [aut. libro] Ing. Arturo Aguiar Vila, Ing. Yaquelín González Padilla Ing. Sofía Álvarez Cárdenas. *Gestión de Requerimientos para Progrmación a Distancia.* 2001.

Canoy, Jeymy J.;. 2005. razabilidad de las Operaciones Electrónicas. Un reto para la Gerencia de Tecnologías de la Información. [aut. libro] Ph. D., CFA, CFE By Jeimy J. Cano. *Trazabilidad de las Operaciones Electrónicas. Un reto para la Gerencia de Tecnologías de la Información.* s.l. : s.l. : Systems Audit and Control Association Inc, 2005.

Casas Rescalvo, Esther. 2012. Framework de desarrollo de Código Abierto, (Utilización de framework de desarrollo de Código Abierto en el mundo empresarial). [aut. libro] Esther Casas Rescalvo. *Framework de desarrollo de Código Abierto, (Utilización de framework de desarrollo de Código Abierto en el mundo empresarial).* 2012.

definicion.org. 2010. [En línea] 2010. [Citado el: 26 de Febrero de 2013.] http://www.definicion.org/lenguaje-de-programacion ----definición.org.

Delgado, Jordan;. 2011. Globered. *Globered.* [En línea] 2011. [Citado el: 26 de Febrero de 2013.] http://teoria-de-programacion.globered.com/categoria.asp?idcat=34.

deviantart. 2013. deviantart. *deviantart.* [En línea] 2013. [Citado el: 09 de Junio de 2013.] http://migueltrega.deviantart.com/art/Enterprise-Architect-210914563.

Downpanda.com. 2007. Downpanda.com. *Downpanda.com.* [En línea] 2007. [Citado el: 04 de Junio de 2013.] http://www.downpanda.com/autorun-architect-download.html.

Ecured. 2010. EcuRed. *EcuRed.* [En línea] 2010. [Citado el: 25 de Octubre de 2012.] http://www.ecured.cu/index.php/Herramientas_de_Gesti%C3%B3n_de_Requisitos..

EcuRed. 2010. EcuRed. *EcuRed.* [En línea] 2010. [Citado el: 04 de Mayo de 2013.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_de_Implementaci%C3%B3n.

EuRailCheck. 2010. EuRailCheck. [En línea] 2010. [Citado el: 09 de Junio de 2013.] https://es.fbk.eu/projects/eurailcheck/index.php?n=Main.RelatedTools.

Farlex. 2013. Diccionario Manual de la Lengua Española. *Diccionario Manual de la Lengua Española.* [En línea] Larousse Editorial, S.L, 2013. [Citado el: 17 de Enero de 2013.] http://es.thefreedictionary.com/gesti%C3%B3n.

Ferraro, María de los Angeles. 2012. ESPECIFICACIÓN Y TRAZABILIDAD DE REQUERIMIENTOS EN EL DESARROLLO DE APLICACIONES WEB. [aut. libro] Yaninaa Medina, Gladysa Dapozo, Marcelob Estayno Maria de los Angeles

REFERENCIAS BIBLIOGRÁFICAS

Ferraro. *ESPECIFICACIÓN Y TRAZABILIDAD DE REQUERIMIENTOS EN EL DESARROLLO DE APLICACIONES WEB.* s.l. : Argentina : s.n, 2012.

FreeDownloadManager. 2008. Paradigma_Visual_para_UML. *Paradigma_Visual_para_UML.* [En línea] 2008. [Citado el: 11 de Noviembre de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

Giraldo, Luis. 2005. *HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX.* 2005.

Globedia. 2012. Globedia, Aspectos teóricos - Entorno de Desarrollo Integrado (IDE). *Globedia, Aspectos teóricos - Entorno de Desarrollo Integrado (IDE).* [En línea] 2012. [Citado el: 26 de Febrero de 2013.] http://cu.globedia.com/aspectos-teoricos-entorno-desarrollo-integrado-ide.

González Ferro, Osmany. 2011. *Un acercamiento a la trazabilidad en el desarrollo ágil de software.* La Habana, Cuba : Grupo Editorial "Ediciones Futuro", 2011.

INTECO. 2008. GUÍA PRÁCTICA DE GESTIÓN DE REQUISITOS. [aut. libro] Instituto Nacional de Tecnologías. *GUÍA PRÁCTICA DE GESTIÓN DE REQUISITOS.* 2008.

IPCorp. 2010. Presentación trabajo "Herramientas de Gestión de Requerimientos". *Presentación trabajo "Herramientas de Gestión de Requerimientos".* [En línea] 2010. [Citado el: 20 de Noviembre de 2012.] http://www.ipcorp.com.ar/blog/2008/12/11/osrmt-open-source-requirements-management-tool/.

Irotes;. 2010. Buenastareas. *Buenastareas.* [En línea] 2010. [Citado el: 14 de Noviembre de 2012.] http://www.buenastareas.com/ensayos/An%C3%A1lisis-Situacional-De-Las-Herramientas-De/398835.html.

Iván Ayala Catari y otros. 2010. Estudio de herramientas CASE de soporte UML y UML2. *Estudio de herramientas CASE de soporte UML y UML2.* [En línea] 2010. [Citado el: 25 de Febrero de 2013.] http://es.scribd.com/doc/25374125/Estudio-de-Herramientas-CASE-de-Soporte-a-UML-y-UML2.

Jacobson, Ivar. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid, España : PEARSON EDUCACION S.A, 2000.

Jaldín, Rolando. 2010. Sistema de Información Herramientas y Metodologías. *Sistema de Información Herramientas y Metodologías.* [En línea] 05 de Octubre de 2010. [Citado el: Junio de 06 de 2013.] http://rolandojaldin.blogspot.com/2010/10/introduccion-la-metodologia-rup-proceso.html.

López, Patricia;. 2010. INGENIERÍA DEL SOFTWARE I, Herramienta CASE Visual Paradigm. *INGENIERÍA DEL SOFTWARE I, Herramienta CASE Visual Paradigm.* [En línea] 2010. [Citado el: 10 de Abril de 2013.] ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/practicas-1/is1-p01-trans.pdf.

Miguel A. Laguna. 2004. Departamento de Informática, Universidad de Valladolid. *Departamento de Informática, Universidad de Valladolid.* [En línea] 2004. [Citado el: 10 de Diciembre de 2012.] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.

—. Ingeniería del Software I, 3º I.T.I.Gestión.

Otero Vidal, Mari Carmen. 2007. Universidad del País Vasco. *Universidad del País Vasco.* [En línea] 2007. [Citado el: 10 de Junio de 2013.] http://www.vc.ehu.es/jiwotvim/IngenieriaSoftware/Teoria/BloqueII/UML-4.pdf.

Pergamino Virtual. 2012. Pergamino Virtual. *Pergamino Virtual.* [En línea] 2012. [Citado el: 26 de Febrero de 2013.] http://www.pergaminovirtual.com.ar/definicion/Java.html.

qt-project.org. 2012. Qt Project. *Qt Project.* [En línea] 2012. [Citado el: 26 de Febrero de 2013.] http://qt-project.org/wiki/QtCreatorWhitepaper_Spanish.

Rational. 2007. Rational. *Rational.* [En línea] 2007. [Citado el: 18 de Enero de 2013.] http://www.rational.com.ar/herramientas/requisitepro.html.

Reynoso, Billy. 2013. *Introducción a la Arquitectura de Software.* BUENOS AIRES : s.n., 2013.

Rouse, Margaret;. 2005. SearchCIO-Midmarket. *SearchCIO-Midmarket.* [En línea] 2005. [Citado el: 25 de Febrero de 2013.] http://translate.google.com.cu/translate?hl=es&sl=en&u=http://searchcio-midmarket.techtarget.com/definition/Rational-Rose&prev=/search%3Fq%3Drational%2Brose%26hl%3Des%26biw%3D1024%26bih%3D605&sa=X&ei=UwMtUYjQBoqw0AHY-IG4Ag&ved=0CF4Q7gEwCQ.

Sánchez. 2010. Ingeniería de Software. *Ingeniería de Software.* [En línea] 2010. [Citado el: 25 de Febrero de 2013.] http://esanchezg220792.blogspot.com/2012/12/case-computer-aided-software.html.

Sandoval Carvajal , Master María Marta. 2012. LA TRAZABILIDAD EN EL PROCESO DE REQUERIMIENTOS DE SOFTWARE. [aut. libro] Master Maria Adilia García Vargas Master Maria Marta Sandoval Carvajal. *LA TRAZABILIDAD EN EL PROCESO DE REQUERIMIENTOS DE SOFTWARE.* 2012.

scruz334. 2007. Introducción a la Ingeniería de Software. *Introducción a la Ingeniería de Software.* [En línea] 2007. [Citado el: 25 de Febrero de 2013.] http://scruz334.blogspot.es/1194151560/.

SlideShare. 2010. Gestión de proyectos de TI, RUP-UML. *Gestión de proyectos de TI, RUP-UML.* [En línea] 2010. [Citado el: 25 de Febrero de 2013.] http://www.slideshare.net/zoilapalacios/02-rup.

Softqatest. 2012. Software, calidad y test. *Software, calidad y test.* [En línea] 2012. [Citado el: 17 de Enero de 2013.] http://www.softqatest.net/?p=77.

Software Corporation. 2003. Rational Rose. *Rational Rose.* [En línea] 2003. [Citado el: 25 de Febrero de 2013.] http://www.slideshare.net/vivi_jocadi/rational-rose.

Sosa López, Dailyn;. 2013. Eumed.net. [En línea] 2013. [Citado el: 07 de Junio de 2013.] http://www.eumed.net/libros-gratis/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm.

Sparxsystems. 2010. Sparxsystems. *Sparxsystems.* [En línea] 2010. [Citado el: 18 de Enero de 2013.] http://www.sparxsystems.com.ar/products/ea.html.

REFERENCIAS BIBLIOGRÁFICAS

Sybven. 2011. Corporación Sybven. *Corporación Sybven.* [En línea] 2011. [Citado el: 18 de Enero de 2013.] http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=101:ibm-rational-requisitepro&catid=82:gestion-de-requerimientos-&Itemid=143.

Tabares, Marta Silvia. 2010. UN MÉTODO PARA LA TRAZABILIDAD DE REQUISITOS EN EL PROCESO UNIFICADO DE DESARROLLO. *UN MÉTODO PARA LA TRAZABILIDAD DE REQUISITOS EN EL PROCESO UNIFICADO DE DESARROLLO.* [En línea] 2010. [Citado el: 17 de Enero de 2013.] http://www.scielo.org.co/scielo.php?pid=S1794-12372007000200007&script=sci_arttext.

Tallercmasmas. 2012. blog de programación. *blog de programación.* [En línea] 2012. [Citado el: 26 de Febrero de 2013.] http://tallercmasmas.blogspot.com/2010/04/breve-descripcion.html.

Targetware. 2010. visual-paradigm-para-um. *visual-paradigm-para-um.* [En línea] 2010. [Citado el: 25 de Febrero de 2013.] http://www.software.com.ar/visual-paradigm-para-uml.html ---Software.com.ar.

Universidad Politécnica de Catalunya. 2012. Matriz de Trazabilidad de los Requerimientos. [aut. libro] Anónimo. *Matriz de Trazabilidad de los Requerimientos.* s.l. : UNIVERSIDAD POLITÉCNICA DE CATALUNYA, España : s.n, 2012.

Zapata Ruiz, Juan. 2012. Proceso Unificado de Rational. *Proceso Unificado de Rational.* [En línea] 2012. [Citado el: 25 de Febrero de 2013.] http://www.slideshare.net/juanzhoo/urp.

Zator Systems. 2013. Curso de C++. *Curso de C++.* [En línea] 2013. [Citado el: 26 de Febrero de 2013.] http://www.zator.com/Cpp/E1_2.htm.

zonaqt. 2010. Inicio rápido con Qt4. *Inicio rápido con Qt4.* [En línea] 2010. [Citado el: 26 de Febrero de 2013.] http://www.zonaqt.com/book/export/html/282.

.

BIBLIOGRAFÍA

Anónimo. 2013. Fundamentos del diseño del software. *Fundamentos del diseño del software.* 2013.

Bárbara A. Mcdonald Landazuri. 2005. Definición de Perfiles en Herramientas. [aut. libro] Bárbara A. Mcdonald Landazuri. *Definición de Perfiles en Herramientas.* s.l. : Madrid, España : Campus de Montegancedo S/N, Boadilla del Monte, 2005.

Blogspot. 2010. blogspot. *blogspot.* [En línea] 2010. [Citado el: 22 de Febrero de 2013.] http://iswp3.blogspot.com/2010/11/descripcion-de-las-fases-del-rup.html.

EcuRed. 2010. EcuRed. *EcuRed.* [En línea] 2010. [Citado el: 22 de Abril de 2013.] http://www.ecured.cu/index.php/Pruebas_de_caja_negra.

Díaz Flores, Miriam Milagros. 2013. *Prácticamente de todo.* 2013.

Itzcoalt Alvarez, Joiz M. 2013. Desarrollo Ágil con SCRUM. [aut. libro] Joiz Net Itzcoalt Alvarez M. *Desarrollo Ágil con SCRUM.* 2013.

XPmetodologia. 2013. Metodología XP. *Metodología XP.* [En línea] 2013. [Citado el: 03 de Marzo de 2013.] https://sites.google.com/site/xpmetodologia/marco-teorico/ventaja.

Campderrich Falgueras, Benet. 2003. *Ingeniería de Software.* Catalunya, España : UOC, 2003.