UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 9

Diseño de MoGeRisk: sistema de gestión de riesgos basado en MoGeRi.

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

Autor: Yorjenys Alarcón Urquiza

Tutor: Ing. Yampier Medina Tarancón

Co-tutor(a): MSc. Yeleny Zulueta Veliz

Ciudad de La Habana, junio de 2010

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente Trabajo de Diploma y reconozco 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 ____ días del mes de _____ del año _____.

_____

Autor:

Yorjenys Alarcón Urquiza

_____ _____

Tutor: Cotutora:

Yampier Medina Tarancón **Yeleny Zulueta Veliz**

RESUMEN

La Gestión de Proyectos es la disciplina para organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definidos. Un proyecto es un esfuerzo temporal, único y progresivo, emprendido para crear un producto o un servicio también único. La Gestión de Riesgos forma parte del área de conocimiento de la GP y constituye un enfoque estructurado para manejar la incertidumbre relativa a una amenaza, a través de una secuencia de actividades humanas que incluyen evaluación de riesgo, estrategias de desarrollo para manejarlo y mitigación del riesgo utilizando recursos gerenciales.

La presente investigación consiste en el análisis y diseño de un sistema que permita la GR basado en el Modelo de Gestión de Riesgos para proyectos de desarrollo de software. Para ello se realiza un estudio exhaustivo de los procesos que integran dicho modelo y de varios sistemas informáticos que implementan y soportan la GR. También se propone un análisis de las herramientas de desarrollo de software a utilizar, quedando finalmente plasmados los resultados del Análisis y Diseño del sistema propuesto.

PALABRAS CLAVES

Gestión, riesgos, proyecto, modelo, análisis y diseño.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA. TECNOLOGÍAS Y HERRAMIENTAS. 6
1.1 Introducción 6
1.2 Conceptos fundamentales relacionados con la Gestión de Riesgos. 6
1.2.1 ¿Qué es Riesgo? 6
1.2.2 Gestión de Riesgos 7
1.2.3 Modelo de gestión de riesgos 7
1.3 Modelo de Gestión de Riesgos para Proyectos de Software en la UCI *(MoGeRi)* 8
1.4 Análisis de soluciones existentes. 8
1.4.2 Análisis comparativo de herramientas para la Gestión de Riesgos. 9
1.4.3 Herramientas para gestión de riesgos en la UCI. 10
1.6 Tecnologías y herramientas a utilizar 11
1.6.1 Metodologías de desarrollo de Software. 12
1.6.2 Lenguaje de modelado. 13
1.6.3 Herramientas CASE. 14
1.6.4 Lenguajes para el desarrollo Web 15
1.6.5 Frameworks 16
1.7 Conclusiones Parciales 17
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA MoGeRisk. 18
2.1 Introducción 18
2.2 Modelo de Dominio 18
2.2.1 ¿Cuándo se aplica un modelo de dominio? 18
2.2.2 Conceptos principales del entorno de la aplicación. 19
2.2.3 Diagrama de clases del modelo de dominio 20
2.3 Especificación de los requisitos de software. 21
2.3.1 Requerimientos Funcionales. 21
2.3.2 Requerimientos No Funcionales. 23
2.4 Descripción del Sistema. Modelos de Casos de Uso del Sistema *(CUS)* 26
2.4.1 Determinación y justificación de los actores del sistema 26
2.4.2 Casos de uso del sistema 27

2.4.3 Diagramas de casos de uso del sistema 29
2.4.4 Descripción textual de los Casos de Uso. 30
2.4.5 Patrones de Casos de Uso 37
2.5 Conclusiones parciales 38
CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA MoGeRisk. 39
3.1 Introducción 39
3.2 Análisis 39
3.2.1 Modelo de análisis 39
3.2.2 Clases del análisis 40
3.3 Diseño 42
3.3.1 Modelo de diseño 42
3.3.2 Extensiones UML para Diseño Web. 43
3.4 Patrones de Diseño utilizados. 45
3.5 Conclusiones parciales 47
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA 48
4.1 Introducción 48
4.2 Métricas para validar el diseño 48
4.2.1 Métricas a nivel de sistema 48
4.2.2 Métricas a nivel de acoplamiento 51
4.2.3 Métricas a nivel de clases 51
4.2.4 Métricas a nivel de métodos 53
4.3 Métricas que se utilizaron 54
4.3.1 Tamaño Operacional de la Clase. 55
4.3.2 Relaciones de Clases 56
4.4 Resumen de los resultados de la evaluación. 58
4.5 Conclusiones parciales 59
CONCLUSIONES 60
RECOMENDACIONES 61
REFERENCIAS BIBLIOGRÁFICAS 62
BIBLIOGRAFÍA CONSULTADA 66
ANEXOS 67

INTRODUCCIÓN

La sociedad actual está encabezada en gran medida por las nuevas tecnologías, donde la informática juega un papel fundamental en todos los ámbitos. Hablar de informática, es hablar de un tema apasionante en todos los sentidos, que hace soñar sobre el futuro, discutir sobre las tecnologías apropiadas y sus costos, las políticas para desarrollar una industria, institución y un país. Pero fundamentalmente hablar de computación o informática es hablar de la necesidad de recursos humanos capacitados, de los cambios en la forma de trabajar y los nuevos empleos, de las nuevas posibilidades de desarrollo individual y de la sociedad.

El camino por la invulnerabilidad económica en Cuba ha encontrado en la informática una posible base de desarrollo del país y una fuente de ingresos, como resultado del correcto aprovechamiento de las ventajas del alto capital humano disponible. Por ello, en los últimos años, el Gobierno Cubano ha creado varias estrategias con el fin de convertir la informática en una de las ramas más productivas y de las que más recursos aporte.

El software tiene un doble papel, es un producto, pero simultáneamente es el vehículo para hacer entrega de éste. El mismo hace entrega de lo que se considera como el activo más importante del siglo XXI, la información. Transforma datos para que sean más útiles en un entorno local, gestiona dicha información para mejorar la competitividad, ofrece el medio de adquirirla en todas sus formas y proporciona el acceso a redes a nivel mundial.

En Cuba el desarrollo de las buenas prácticas de producción de software es aún incipiente, existe un alto porcentaje de soluciones artesanales y a la medida lo que no permite los grandes avances a que se aspiran, por otro lado, las exigencias de la producción no fomentan la mejora de la calidad de los procesos de desarrollo o de sus productos pues se está a expensas a las exigencias en tiempo, costo y calidad (Casañola, 2007).

La Gestión de Riesgos (GR) se encuentra dentro del área de conocimiento de la Gestión de Proyectos (GP). El desarrollo de software, como muchos procesos de carácter ingenieril, está expuesto a la incidencia de múltiples riesgos: durante el proceso de desarrollo es muy posible que se introduzcan cambios que afecten los objetivos de este, dichos cambios generalmente se traducen en riesgos y pueden

influir en aspectos de gran importancia como son el coste, la planificación temporal o la calidad del producto.

El concepto de riesgo en el contexto de la GP de software cuenta con múltiples definiciones elaboradas por expertos en el tema. No obstante, en Presman se dice que un riesgo es un problema potencial, puede ocurrir o no. Pero sin tener en cuenta el resultado, realmente es una buena idea identificarlo, evaluar su probabilidad de aparición, estimar su impacto, y establecer un plan de contingencia por si ocurre el problema (Presman, 2005).

La GR de un proyecto incluye los procesos relacionados con la planificación de la gestión de riesgos, la identificación y el análisis de riesgos, las respuestas a los riesgos, el seguimiento y control de riesgos de un proyecto, así como la comunicación; la mayoría de estos procesos se actualizan durante el proyecto. Los objetivos de la gestión de los riesgos del proyecto son aumentar la probabilidad y el impacto de los eventos positivos, y disminuir la probabilidad y el impacto de los eventos adversos para el proyecto.

Estos procesos interactúan entre sí. Cada proceso puede implicar el esfuerzo de una o más personas o grupos de personas, dependiendo de las necesidades del proyecto. Cada proceso tiene lugar por lo menos una vez en cada proyecto y se realiza en una o más fases del proyecto, si el proyecto se encuentra dividido en fases (Jacobson, 2004).

Con el objetivo de lograr una vinculación fuerte entre la Universidad y la Industria se creó, hace siete años la Universidad de la Ciencias Informáticas (UCI); introduciendo un nuevo concepto basado en la relación Universidad-Industria, una Universidad Productiva. En esta universidad la producción es un problema social, político, y económico, donde los estudiantes y profesores estén vinculados a la producción (Casañola, 2007).

La UCI es la primera universidad surgida al calor de la Batalla de Ideas, por lo que juega cada día un papel protagónico y decisivo en esta nueva forma de trinchera de lucha. Es una institución que combina la formación, la producción y la investigación.

Este centro a través del concurso de los mejores especialistas del país para lograr una solución de calidad y de impacto internacional promueve el desarrollo de productos y servicios informáticos en ramas

como la salud y la educación donde Cuba tiene un reconocido prestigio en el mundo. Desarrolla programas de informatización de la sociedad cubana a través de la relación con entidades nacionales. Los resultados alcanzados se extienden por todo el país.

En los proyectos productivos de la UCI se ha identificado que existen muchos de ellos en los que el desarrollo de software tiene deficiencias en la definición de los flujos de procesos, roles y responsabilidades, los cuales no siempre responden a sus necesidades y a la metodología utilizada, afectándose la eficiencia, la calidad, y el tiempo de desarrollo de un producto (Casañola, 2007).

Existen varios modelos de GR que se utilizan en proyectos de desarrollo de software. La UCI cuenta con un modelo de Gestión de Riesgos, conocido como Modelo de Gestión de Riesgos para proyectos de desarrollo de software (MoGeRi), elaborado teniendo en cuenta las características de su entorno productivo y las mejores prácticas utilizadas a nivel mundial.

Aunque hay herramientas a nivel internacional que se utilizan en la GR, ninguna satisface las necesidades particulares del centro, al no implementar un modelo de GR propio o no estar accesibles por licencias o restricciones económicas.

Teniendo en cuenta la situación problemática planteada el **problema científico** al que se le dará solución en esta investigación es el siguiente: Insuficiencia en la GR en los proyectos de desarrollo de software de la UCI, debido a la inexistencia de un sistema para la GR.

El **objeto de estudio** de esta investigación lo constituyen los procesos de la GR del modelo MoGeRi, y el **objetivo general** que se persigue es diseñar un sistema para gestionar los riesgos de un proyecto productivo basado en MoGeRi. El **campo de acción** de la investigación es el diseño de un sistema de GR.

Una vez establecido el problema científico y el objetivo de esta investigación se puede plantear como **hipótesis** que a partir del análisis de los procesos del modelo MoGeRi se logrará el diseño de un sistema para la GR en la UCI.

Al concluir la investigación se tendrá el diseño de MoGeRisk. Para obtener estos resultados es necesario llevar a cabo las siguientes **tareas**:

1. Describir a nivel mundial y nacional las soluciones que en la actualidad implementan y soportan los procesos de GR.

2. Caracterizar cómo se utilizan actualmente los procesos de GR en los proyectos productivos de la UCI.

3. Analizar el lenguaje de programación, el lenguaje de modelado, la metodología de desarrollo de software y las herramientas CASE a utilizar.

4. Realizar el diseño de los procesos de GR a partir de las herramientas, las técnicas y buenas prácticas seleccionadas.

5. Evaluar el diseño.

Se utilizaron los siguientes métodos científicos:

Histórico-lógico: se utilizará con el objetivo de estudiar todo lo referente a los procesos de GR, así como, las soluciones que en la actualidad implementan y soportan estos procesos. Este método ayuda a dar cumplimiento al estudio del estado del arte de los procesos de gestión de riesgos.

Analítico-Sintético: Este método se utilizará para organizar y sintetizar toda la información obtenida del estudio del estado del arte, las tecnologías, metodologías, lenguajes y herramientas propuestas para la solución. La utilización de este método permite comprender mejor toda la información anterior, la cual será de gran utilidad para lograr un adecuado análisis y diseño del sistema.

Método de la Modelación: Partiendo de todo lo que se investigue y aprenda se realizarán los modelos correspondientes al ciclo de vida del software, que ayudarán a dar cumplimiento a las tareas de diseño de los procesos involucrados en la solución.

Método de observación: Este método sirve de apoyo para obtener información sobre el tema a tratar, y ayuda a dar solución a prácticamente todas las tareas. Haciendo uso de él se puede ver cómo funciona actualmente la gestión de riesgos, de qué manera fue desarrollado el sistema que actualmente se tiene en

la Universidad, las ventajas y desventajas de las tecnologías, metodologías, lenguajes y herramientas a utilizar en el desarrollo del nuevo sistema.

Entrevistas: Diseño y teoría de muestreo.

Se realizaron conversaciones planificadas para obtener conocimientos cualitativos sobre problemas que afectan la producción en la UCI y la utilización de las prácticas de la GR.

Se realizó con el objetivo de identificar el grado de conocimiento de los involucrados acerca de la GR, su utilización en la UCI, y su aplicación en los proyectos productivos. La población a estudiar fue el personal involucrado en los proyectos de desarrollo de software en la UCI y como unidad de estudio el individuo que integra este grupo.

Se consideró una muestra de 24 individuos seleccionando el muestreo intencional, técnica no probabilística que permite elegir explícitamente los elementos que son representativos o con posibilidades de brindar mayor información. En este caso la muestra se seleccionó teniendo en cuenta la experiencia en la gestión de riesgos y en los proyectos productivos de la UCI. La guía para entrevista se corresponde con el anexo 43.

La siguiente investigación se encuentra estructurada en cuatro capítulos. En el primero de estos se muestra un estudio del comportamiento de la GR en la UCI y en el mundo. Además se hace una descripción sobre las herramientas, metodología y lenguajes que se utilizan y se proponen. El capítulo dos cuenta con una descripción de la solución propuesta a través del modelo de dominio. Se detallan los requisitos funcionales y no funcionales, así como los casos de usos que se generan. El tercero recoge los diagramas de clases del análisis y el diseño, teniendo en cuenta los patrones de diseño utilizados. En el capítulo cuatro se aplican métricas para medir la calidad del diseño, inspiradas en el estudio de la calidad del DOO.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA. TECNOLOGÍAS Y HERRAMIENTAS.

1.1 Introducción

En este capítulo se abordan aspectos que serán de ayuda y guía para la comprensión de las diferentes definiciones y conceptos relacionados con la gestión de riesgos así como el modelo MoGeRi. Se muestra un estudio del comportamiento de la GR en la UCI y en el mundo. Además se hace una descripción sobre las herramientas, metodología y lenguajes que se proponen.

1.2 Conceptos fundamentales relacionados con la Gestión de Riesgos.

Como parte de la fundamentación teórica se hace necesario tener en cuenta algunos conceptos que serán de gran ayuda para la correcta comprensión de esta investigación, como son precisamente la definición de GR y otros asociados al tema.

1.2.1 ¿Qué es Riesgo?

El SEI (*Software Engineering Institute*) define al Riesgo como la posibilidad de sufrir una pérdida (SEI, 2004).

En (Presman, 2005) se puede encontrar la definición de riesgo dada por Robert Charette (Charette, 1989) donde plantea en primer lugar que, el riesgo afecta a los futuros acontecimientos. En segundo lugar, el riesgo implica cambios. En tercer lugar, el riesgo implica elección, y la incertidumbre que entraña ésta. Cuando se considera el riesgo en el contexto de la Ingeniería de Software, los tres pilares de Charette se hacen continuamente evidentes.

El término riesgo se utiliza en la mayoría de los casos como cualquier ocurrencia de un evento no deseado (Navarro, 2007).

Partiendo de las definiciones anteriores se puede afirmar que un riesgo, es la ocurrencia de eventos no deseados, que pueden o no suceder comprometiendo los resultados y objetivos del proyecto u organización. Por tanto, el riesgo irá acompañado de todo cambio o decisión que se produzca en el proyecto, pues siempre representan incertidumbre ante lo que puede ocurrir.

1.2.2 Gestión de Riesgos

El SEI afirma que la GR es la práctica compuesta de procesos, métodos y herramientas que posibilita la gestión de los riesgos en un proyecto y que provee de un entorno disciplinado para la toma de decisiones proactivas en base a determinar constantemente que puede ir mal, identificar cuáles son los riesgos más importantes en los cuales enfocarse e implementar estrategias para gestionarlos (SEI, 2004).

Según la Guía de los Fundamentos de la Dirección de Proyectos (PMBoK) la GR del proyecto incluye los procesos relacionados con la planificación de la GR, la identificación y el análisis de riesgos, las respuestas a los riesgos, y el seguimiento y control de riesgos de un proyecto; la mayoría de estos procesos se actualizan durante el proyecto. Los objetivos de la GR del Proyecto son aumentar la probabilidad y el impacto de los eventos positivos, y disminuir la probabilidad y el impacto de los eventos adversos para el proyecto (PMBoK, 2004).

Según los conceptos anteriormente mencionados, como todos giran sobre una misma idea, se llega a la conclusión de que el objetivo fundamental de la GR es identificar, controlar y eliminar las fuentes de riesgos antes que comiencen a afectar los resultados del proyecto. Por tanto, la gestión de riesgos es una de las disciplinas más importantes de los sistemas de gestión de proyectos. Proporciona en las organizaciones un marco para administrar con eficacia y eficiencia, la incertidumbre y los riesgos asociados y oportunidades.

1.2.3 Modelo de gestión de riesgos

Según Lavell un Modelo de Gestión de Riesgo (MGR) consiste en construir la información mínima que permita calcular el riesgo que se va a asumir y prever las reservas que permitirían la supervivencia aún en caso de que ocurran impactos. Identificar a los actores involucrados para: a) elaborar la información y definir las responsabilidades para la elaboración de las opciones de respuesta, y b) establecer los plazos para alcanzar niveles de bienestar y de disminución de los riesgos.

“Un MGR consiste en construir la información mínima que permita calcular el riesgo que se va a asumir, y prever las reservas (financieras, sociales, psicológicas, emocionales, etc.) que permitirían la supervivencia

en condiciones adecuadas, a pesar de la ocurrencia de ciertos impactos probables en determinado período de tiempo" (ITDG, 2008).

1.3 Modelo de Gestión de Riesgos para Proyectos de Software *(MoGeRi)*.

El Modelo de Gestión de Riesgos para Proyectos de Software (MoGeRi) cuenta con 6 procesos, Planificación de la Gestión de Riesgos, Identificación de riesgos, Análisis de riesgos, Planificación de la Respuestas de los riesgos, Seguimiento y Control de los riesgos y Comunicación de la información de los riesgos.

A continuación se muestra la figura 1, donde intervienen cada uno de los procesos anteriormente mencionados:

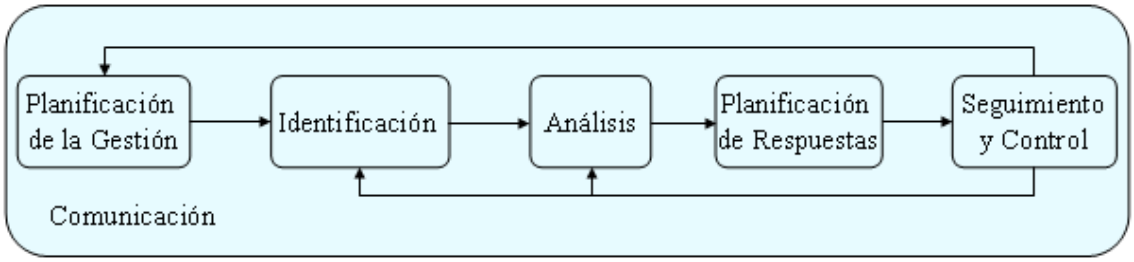


Figura 1. Procesos del MoGeRi (Zulueta, 2007).

El funcionamiento del modelo se basa en la realización de un conjunto de actividades por cada proceso, y a su vez el cumplimiento de cada actividad está representado por el desarrollo de diferentes tareas.

Las tareas se describen utilizando una tabla que tiene el proceso al que pertenece con un identificador y nombre de este, una actividad con identificador y nombre de la misma donde se desarrolla la tarea con identificador y nombre, los datos de entrada que son una serie de informaciones previamente realizadas (documentos, informes, registros, otras tareas y planes del proyecto), todas vinculadas con el tema de la GR.

1.4 Análisis de soluciones existentes.

Controlar y administrar el riesgo, proporcionando la máxima integridad, disponibilidad y confidencialidad es una tarea difícil que se puede mejorar con el uso de herramientas que permitan automatizar estos

procesos de gestión. Existen disímiles maneras de gestionar adecuadamente el riesgo, así como herramientas que ayudan a mejorar las predicciones, a planificar lo inesperado y a aumentar la confianza en la calidad de las decisiones bajo incertidumbre. Ejemplo de estas herramientas son: @Risk, TopRank, WelcomRisk, Risk Matrix, Risk +, entre otras.

1.4.2 Análisis comparativo de herramientas para la Gestión de Riesgos.

Son disímiles las herramientas que se pueden utilizar para apoyarse en la realización de una correcta GR.

El siguiente cuadro permite realizar una comparación entre las principales características de las herramientas anteriormente descritas:

Herramientas	Reportes	Libre	Gráficos	Entorno	Se integra a:
@Risk	Sí	No	Sí	Escritorio	MS. Excel
TopRank	Sí	No	Sí	Escritorio	MS. Excel
PrecisionTree	Sí	No	Sí	Escritorio	MS. Excel
Crystal Ball	Sí	No	Sí	Escritorio	MS. Excel
Risk Matrix	Sí	No	Sí	Escritorio	MS. Excel
RiskTrak	Sí	No	Sí	Web	No se integra
RiskNavR	Sí	No	No	Web	MS. Access
RiskRadar	Sí	No	Sí	Web	No se integra
Risk +	Sí	No	Sí	Escritorio	MS. Project

TRIMS	Sí	Sí	Sí	Escritorio	No se integra
WelcomRisk	Sí	No	Sí	Web	MS. Project, Primavera
PILAR	Sí	Sí		Escritorio	No se integra

Tabla 1. Cuadro comparativo de herramientas para la GR (Mendoza, 2009).

En el cuadro de comparación anterior, las herramientas analizadas tienen características similares, sin embargo, presentan diferencias en cuanto a la plataforma en que son utilizadas así como al software que se integran. Una de las principales deficiencias que se deben de destacar que casi la totalidad de las herramientas son propietarias, y no libres.

En la actualidad existen una gran cantidad de herramientas de software provistas en el mercado y relacionadas con las actividades de GR, específicamente, algunas de ellas orientan su funcionalidad al soporte de la identificación de riesgos y análisis de los riesgos.

1.4.3 Herramientas para gestión de riesgos en la UCI.

La gestión de proyectos es la disciplina que se encarga de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definidos.

El uso de herramientas para la gestión de proyectos se ha ido generalizando al paso del tiempo con el objetivo de facilitar el trabajo de los especialistas que se dedican a estas tareas. Entre estas herramientas se encuentra el *RedMine* desarrollada con el framework *Ruby on Rails*. Es un una aplicación web que tiene la ventaja de ser software libre bajo licencia GPL (GNU General Public License v2).

Entre sus principales características tenemos las siguientes:

- Soporte a múltiples proyectos.
- Publicación de Noticias, Documentos, Wiki y archivos.

- Foros.
- Seguimiento al Tiempo (Time Tracking).
- Integración con manejadores de configuración de código tales como SVN (Subversion), CVS y otros.
- Gestión de Riesgos.

Debido a la gran cantidad de funcionalidades que facilita se toma como propuesta de herramienta para la gestión de los proyectos informáticos de la Universidad de las Ciencias Informáticas teniendo en la actualidad un entorno para cada centro que aunque no está en su versión final si se encuentra funcional y en proceso de desarrollo de mejoras que incluye integración con otras herramientas para la gestión documental como Alfresco y otras (Perez, 2010).

En la Universidad se han dado los primeros pasos para la creación de aplicaciones que apoyen el proceso de GR en los proyectos productivos, tal es el caso del *RedMine*, sistema que cuenta con un módulo dedicado a dicha gestión, capaz de automatizar la identificación, el análisis y la mitigación de los riesgos.

1.6 Tecnologías y herramientas a utilizar.

El proceso de desarrollo del software, define el conjunto de actividades precisas para convertir los requisitos de los usuarios en el conjunto seguro y resistente de artefactos que componen un producto de software. Las tendencias presentes, luego del perfeccionamiento de los procesos del software durante años, han llevado a cabo dos corrientes significativas: los llamados métodos ligeros y métodos pesados.

Los métodos ligeros o ágiles, proponen mejorar la calidad del software teniendo como premisa la comunicación inmediata y directa, mientras que los métodos pesados obtienen sus resultados a través de orden y documentación.

Después de haber realizado junto al arquitecto de MoGeRisk un análisis profundo de las metodologías y las herramientas que le dan soporte, se decidió por RUP como metodología con UML como lenguaje de modelado junto a Visual Paradigm for UML 6.1 Enterprise Edition como herramienta CASE y PHP 5 como lenguaje de programación para desarrollar el sistema.

1.6.1 Metodologías de desarrollo de Software.

Se hace necesario definir metodologías para guiar el proceso de desarrollo de un producto de software. Las metodologías se definen por pasos a seguir para el cumplimiento de un objetivo. El objetivo dentro del desarrollo del software es producir un producto de alta calidad que cumpla con los requerimientos del cliente.

Entre las características generales que presentan dichas metodologías se encuentran las siguientes:

1. No pueden aplicarse a todo tipo de proyectos.
2. Están orientadas en función de los nuevos principios de desarrollo del software.
3. Pueden ser ajustables de acuerdo a las características del proyecto.

1.6.1.1 **RUP** *(Proceso Unificado de Desarrollo de Software)*.

Es una metodología pesada. Describe detalladamente todas las actividades, roles, responsabilidades, productos de trabajo y herramientas para definir el quién, cómo, qué y cuándo en un proyecto de desarrollo de software. Esta metodología unifica los mejores elementos de las anteriores, por lo que se encuentra preparada para el desarrollo de grandes y complejos proyectos. RUP, representa un ideal de referencia para todo el equipo de desarrollo.

Es además, un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software, se puede especializar para gran variedad de sistemas de software, distintas áreas de aplicación, tipos de organizaciones, niveles de actitud y tamaños de proyecto. Se caracteriza por estar divido en fases. En cada una de estas se producirán una o varias iteraciones, cuyo tamaño varía según la complejidad del proyecto. Dentro de cada una de ellas se seguirá un modelo de cascada en los flujos de trabajo que lo requieran.

Las fases de RUP concluyen con un hito bien definido, en cada uno de estos se deben tomar acuerdos y decisiones, garantizando el cumplimiento de los objetivos y metas antes de la transición a la nueva fase (Cortés, 2004). Ver la figura 3.

Los hitos por cada una de las fases son:

1. Inicio: visión de los objetivos.
2. Elaboración: prototipo de la arquitectura.
3. Construcción: capacidad operacional inicial.
4. Transición: liberación del producto.

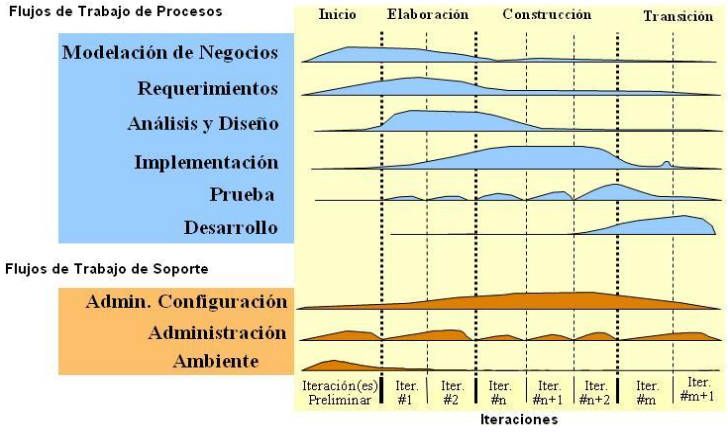


Figura 2. RUP. Ciclo de vida (Cortés, 2004).

1.6.2 Lenguaje de modelado.

El modelado de sistemas de software es una técnica para tratar con la complejidad inherente a estos sistemas. El uso de modelos ayuda al ingeniero de software a "visualizar" el sistema a construir. Además, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente.

1.6.2.1 UML 2.1 *(Unified Modeling Language)*.

El Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad. Es un lenguaje que permite visualizar, especificar, construir y documentar, modelos de sistemas de software, incluyendo su estructura y diseño, que capta la información sobre la estructura estática y el comportamiento dinámico de un sistema.

Este lenguaje de modelado permite modelar sistemas utilizando técnicas orientadas a objetos, especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos, documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.). Puede conectarse con lenguajes de programación (ingeniería directa e inversa), cubrir todas las vistas necesarias para desarrollar y luego desplegar los sistemas, dado a que es un lenguaje muy expresivo, las cuestiones relacionadas con el tamaño, propias de los sistemas complejos y críticos, y equilibrar expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.6.3 Herramientas CASE.

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste en términos de tiempo y costo. Estas herramientas ayudan 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 costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación y detección de errores.

1.6.3.1 Visual Paradigm for UML 6.1 Enterprise Edition *(VP-EE)* (Cuevas, 2009).

Visual Paradigm para UML es una herramienta CASE que soporta UML 2.1 como lenguaje de modelado. Esta útil herramienta apoya el ciclo de vida completo de desarrollo del software, análisis, diseño, implementación y prueba. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación.

Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código.

El análisis textual es una técnica útil y práctica para la captura de los requisitos del sistema y la identificación de las clases candidatas, Visual Paradigm es una de las pocas herramientas CASE que soporta el análisis textual.

Ofrece un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad, además presenta disponibilidad de múltiples plataformas y en múltiples versiones. Está disponible para varios sistemas operativos como Windows, Linux, Unix.

1.6.4 Lenguajes para el desarrollo Web

El desarrollo de aplicaciones Web ha tenido un auge en todo el mundo gracias a las ventajas que las mismas ofrecen a empresas e instituciones. Para el desarrollo de estas existen numerosos lenguajes informáticos que se dividen en dos grupos, el primer grupo abarca los lenguajes que corren en el lado del cliente y el segundo los lenguajes que corren en el lado del servidor. Las técnicas de desarrollo Web y lenguajes del lado del cliente más utilizados son HTML (*HyperText Markup Language*) y JavaScript, mientras que los lenguajes de programación del lado del servidor más usados en software libre son Java, PHP y Perl.

1.6.4.1 PHP 5 *(Hypertext Pre-processor)* (León, 2009).

PHP es un lenguaje de programación interpretado, el cual fue diseñado originalmente para la creación de páginas web dinámicas aunque últimamente también ha intervenido en la creación de aplicaciones con interfaces gráficas usando bibliotecas específicas.

Es capaz de combinarse con servidores de bases de datos tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de aplicaciones Web robustas y, arroja resultados muy interesantes y prometedores para aquellas páginas que deseen ser activas y dinámicas. La ejecución e interpretación de PHP es completamente en el servidor Web: en este se encuentra almacenado el script por lo que el cliente solo puede recibir el resultado de la ejecución.

El uso de PHP presenta ventajas como:

1. Es capaz de leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.

2. Cuenta con una biblioteca sumamente amplia por defecto de funciones.

3. No requiere definición de tipos de variables ni manejo detallado del bajo nivel.
4. Tiene capacidad de conexión con la mayoría de los manejadores de base de datos utilizados en la actualidad.
5. Posee una amplia documentación en su Web oficial de Internet. En la misma se encuentran muy bien explicadas todas las funciones del sistema, contando con ejemplos detallados.
6. Pertenece a la alternativa de código abierto (*Open Source*), por lo que se presenta como una elección de fácil acceso para todos.
7. Permite las técnicas de Programación Orientada a Objetos.
8. Permite crear formularios para la Web.

1.6.5 Frameworks

Este no es más que una estructura de soporte definida mediante la cual otro proyecto de software puede ser desarrollado y organizado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

1.6.5.1 **Symfony 1.4.3** (León, 2009)**.**

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas Unix, Linux, como en plataformas Windows. Este posee una serie de características que se enuncian a continuación:

Características:

1. Fácil de instalar y configurar en la mayoría de las plataformas.
2. Independiente del sistema gestor de bases de datos.
3. Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos
4. Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
5. Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
6. Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
7. Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
8. Fácil de extender, lo que permite su integración con las librerías de otros fabricantes.

1.7 Conclusiones Parciales

En la primera parte de este capítulo se hizo referencia a conceptos relacionados con el objeto de estudio definido en la introducción del documento. Se desarrolla el estudio de algunas de las tecnologías y herramientas actuales que se utilizarán para realizar el diseño y otras que serán empleadas durante el desarrollo de la aplicación, la decisión de utilizarlas fue tomada siguiendo fundamentalmente la política de uso de herramientas con soporte multiplataforma y licencias de utilización libre.

Tras haber realizado junto al arquitecto de MoGeRisk un análisis profundo de las metodologías y las herramientas que le dan soporte, se decidió por RUP como metodología de desarrollo con UML 2.1 como lenguaje de modelado junto a Visual Paradigm como herramienta CASE, PHP 5 como lenguaje de programación para desarrollar el sistema y Symfony como framework para optimizar el desarrollo de las aplicaciones web.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA MoGeRisk.

2.1 Introducción

En este capítulo se hace una descripción de la solución propuesta a través del modelo de dominio representando los principales conceptos asociados al trabajo. Se detallan los requisitos funcionales y no funcionales, los casos de usos que se generan a partir de los requisitos funcionales y una explicación de cada uno de ellos a través de las descripciones textuales.

2.2 Modelo de Dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Las clases del dominio aparecen en tres formas típicas:

1. Objetos del negocio que representan cosas que se manipulan en el negocio.
2. Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
3. Sucesos que ocurrirán o han ocurrido.

El modelo de dominio se describe mediante diagramas UML (especialmente mediante diagrama de clases). Estos diagramas muestran las clases del dominio y cómo se relacionan unas con otras mediante asociaciones (Jacobson, 2004).

El modelo de dominio ayuda además a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común.

2.2.1 ¿Cuándo se aplica un modelo de dominio?

El Modelo de Dominio se aplica cuando:

1. Los flujos de información son difusos (múltiples orígenes, sólo eventos, sucesos).
2. Imposibilidad de determinar subsistemas (exceso de interconexiones).

3. Solapamiento de responsabilidades.
4. Múltiples responsabilidades.
5. Difícil establecimiento de reglas de funcionamiento.

2.2.2 Conceptos principales del entorno de la aplicación.

Proyecto: Entorno donde se realizará la gestión de riesgos.

Gestión de Riesgo: Conjunto de acciones que se llevan a cabo para la estimación y control de los riesgos en un proyecto.

Proceso: Parte fundamental por la que está compuesta la gestión de riesgos.

Riesgo: Ocurrencia de eventos no deseados, que pueden o no suceder comprometiendo los resultados y objetivos de un proyecto u organización.

Equipo de GR: Conjunto de personas encargadas de realizar los procesos de la gestión de riesgos.

Registro de Riesgo: Documento donde se registran todas las características de los riesgos (Ver figura 4).

MoGeRi: Modelo de gestión de riesgos.

Registro de Riesgos

Proyecto: Nombre del proyecto.

Nombre del riesgo. Un nombre que lo identifique de manera rápida.

Descripción del riesgo. Permite tener una representación más completa del alcance del riesgo y de los perjuicios potenciales para el proyecto.

Frecuencia.

Causas o fuentes.

Eventos potenciales.

Posibles respuestas.

Variaciones del mismo riesgo que se puedan identificar para evitar tener el mismo riesgo con distintos nombres o tener riesgos identificados que se solapen.

Personas o entidades implicadas en el riesgo y que en caso de cualquier cambio o decisión deben ser consultadas previamente

Análisis del Riesgo.

Fecha de Análisis								
Impacto								
Probabilidad								

Estrategia para tratamiento del riesgo

Respuestas: Se hace referencia a la planificación de la respuesta en el PGR si esta ya fue incluida en él, si no, se describen las posibles respuestas par el riesgo.

Monitorización

Indicadores de que se pueda producir el riesgo. La evaluación de éstos indicadores definirá cuan cerca estamos en cada momento de que se produzca el riesgo.

Modos de evaluación de los indicadores, previsión de la frecuencia con que debe reevaluarse el riesgo, personas o entidades implicadas en la reevaluación, etc.

Figura 3. Formato para registro de riesgos (Zulueta, 2007).

2.2.3 Diagrama de clases del modelo de dominio

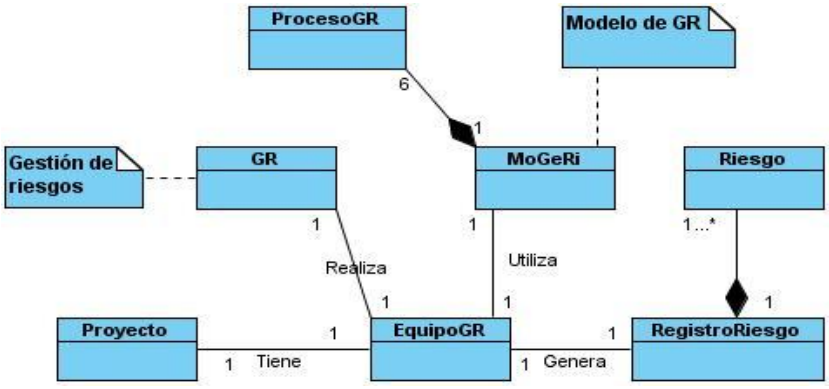


Figura 4. Diagrama de clases del modelo de dominio (Elaboración propia).

2.3 Especificación de los requisitos de software.

A través de los años se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo.

La especificación de requerimientos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema y es contra lo que se va a estar verificando si se están cumpliendo las metas trazadas.

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales (Rosales, 2008):

1. *Los requerimientos funcionales (RF):* definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

2. *Los requerimientos no funcionales (RNF):* tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

2.3.1 Requerimientos Funcionales.

RF1 Autenticar usuario.

RF2 Gestionar proyectos.

RF2.1 Insertar proyectos.

RF2.2 Modificar proyectos.

RF3 Gestionar equipo.

RF3.1 Insertar equipo.

RF3.2 Modificar equipo.

RF4 Gestionar documentos.

RF4.1 Insertar documento.

RF4.2 Modificar documento.

RF5 Programar actividades.

RF5.1 Insertar actividades.

RF5.2 Modificar actividades.

RF5.3 Seleccionar actividades.

RF6 Comunicar resultados.

RF7 Asignar actividades.

RF8 Estimar costos.

RF9 Mostrar experiencias.

RF10 Seleccionar herramientas y técnicas.

RF10.1 Insertar herramientas y técnicas.

RF10.2 Modificar herramientas y técnicas.

RF10.3 Seleccionar herramientas y técnicas.

RF11 Gestionar riesgos.

RF11.1 Insertar riesgo.

RF11.2 Modificar riesgo.

RF12 Caracterizar riesgos.

RF13 Estimar exposición.

RF14 Mostrar riesgos priorizados.

RF15 Gestionar estrategias.

RF15.1 Insertar estrategia.

RF15.2 Modificar estrategia.

RF16 Gestionar respuestas.

RF16.1 Insertar respuesta.

RF16.2 Modificar respuesta.

RF17 Planificar respuestas.

RF18 Gestionar métricas.

RF18.1 Insertar métrica.

RF18.2 Modificar métrica.

RF19 Mostrar registro de riesgos.

RF20 Documentar experiencias.

2.3.2 Requerimientos No Funcionales.

Los RNF son propiedades o cualidades que el producto debe tener:

RNF 1 Requerimiento de Apariencia o interfaz:

La interfaz del sistema será a través de una aplicación Web. Diseño sencillo y amigable, permitiendo que no sea necesaria una capacitación para utilizar la herramienta en el Sistema Operativo (SO) Windows, sí para Linux. Debe ser además un diseño formal y serio teniendo en cuenta el fin con el que se desarrolla la aplicación.

RNF 2 Requerimiento de Usabilidad:

La aplicación debe ser concebida para ser utilizada por personas que tengan conocimientos medios de informática en el trabajo con aplicaciones web en sentido general en los SO Windows y Linux. Garantizará una conexión segura con la base de datos que tendrá almacenada la información del sistema ayudando así a la gestión de los datos.

RNF 3 Requerimiento de Rendimiento:

Debido a la importancia de la información que se procesa en la aplicación, el sistema deberá ser lo más estable y confiable posible.

RNF 4 Requerimiento de Soporte:

El sistema debe responder en un tiempo relativamente corto a las peticiones del usuario (menos de 5s). La BD deberá resistir una gran cantidad de visitas simultáneas, así como un gran número de operaciones paralelas sobre la misma.

RNF 5 Requerimientos de Software.

PC clientes:

Sistema Operativo tanto Windows (win9.x o versión superior) como Linux (cualquiera de sus distribuciones). El Navegador Web compatible con HTML 2.0 y CSS, ejemplo de eso puede ser: Internet Explorer 4.2 (o superior), Mozilla Firefox 3.0 (o superior).

PC Servidor Web:

Sistema Operativo Windows (win9.x o versión superior) o Linux (cualquiera de sus versiones). Servidor Web Apache 2.2.X o superior.

Servidor de Bases de Datos:

Sistema Operativo Windows (win9.x o versión superior) o Linux (cualquiera de sus versiones). Servidor de Bases de Datos PostgreSQL 8.3.

RNF 6 Requerimiento de Seguridad:

La seguridad se trata desde las primeras fases de desarrollo del sistema. La asignación de usuarios y sus opciones sobre el sistema se garantizan desde el módulo de Seguridad del sistema. Los mecanismos de seguridad no deben imposibilitar que los usuarios autorizados accedan a la información, la misma debe estar disponible en todo momento. Una parte de la seguridad la aporta el framework de desarrollo utilizado con el controlador frontal que permite revisar todas las particiones al sistema y con eso el usuario que las realiza.

La base de datos deberá estar disponible las 24 horas. En caso de una falla en el servidor (interrupción del servicio eléctrico, desconexión momentánea o caída de la red) se deberá contar con un respaldo para continuar con las actividades de cada módulo. Se usan mecanismos de encriptación de los datos que por cuestiones de seguridad no deben viajar al servidor en texto claro, como es el caso de las contraseñas.

RNF 7 Requerimiento de Confiabilidad:

Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario para evitar entradas inadecuadas.

RNF 8 Restricciones de acuerdo al diseño.

El diseño se realizará aplicando el paradigma de la Programación Orientada a Objetos (POO). Se hará uso de las tecnologías que ofrece el framework Symfony. El sistema deberá ser desarrollado como una aplicación Web. Será libre y multiplataforma.

2.4 Descripción del Sistema. Modelos de Casos de Uso del Sistema *(CUS)*.

Luego de haber realizado una pequeña descripción del sistema, se da paso a la descripción del modelo de casos de uso del sistema haciendo uso de las ventajas que brinda el lenguaje de modelado UML, se formulan las funcionalidades del sistema y representación mediante un diagrama, para ello es de vital importancia definir los actores y los casos de uso que representarán las responsabilidades del mismo.

2.4.1 Determinación y justificación de los actores del sistema

Un actor es un rol que cumple un usuario, puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema. En la bibliografía se ve reflejado los dos casos, la explicación se describe a continuación (Jacobson, 2004).

Actores	Justificación
Administrador	Se encarga de gestionar los recursos del sistema. Tiene la posibilidad de insertar, modificar o eliminar cada uno de los proyectos del sistema.
Analista	Realiza el análisis a los riesgos.
Documentador	Documenta las experiencias personales.
Identificador	Identifica los riesgos del proyecto.
Líder	Gestiona los datos de los integrantes del equipo.
Planificador de respuestas	Gestiona los datos de las estrategias de los riesgos del proyecto.
Seguimiento y control	Realiza el cálculo de métricas a los proyectos.
Usuario	Accede al sistema.
Equipo de GR	Llevar a cabo las tareas de GR. Recopilar, procesar y consolidar datos. Elaborar los

	informes previstos en el desarrollo de los procesos.

Tabla 2. Justificación de los actores del sistema (Elaboración propia).

2.4.2 Casos de uso del sistema

Un caso de uso constituye una técnica utilizada para describir el comportamiento del sistema, a través de un documento narrativo que define la secuencia de acciones que obtienen resultados de valor para un actor que utiliza un sistema para completar un proceso, sin importar los detalles de la implementación (Rosales, 2008).

Código	Nombre de caso de uso	Módulo	Justificación de la selección
1	Autenticar usuario	Seguridad	Este CU es muy importante, de él dependen los privilegios que se puedan otorgar.
2	Gestionar proyectos	Seguridad	Se gestionan los datos de los proyectos.
3	Gestionar equipo	Gestión de riesgos	Se gestionan los datos de los integrantes del equipo.
4	Gestionar documento	Documentar experiencia	Se gestionan todos los documentos correspondientes a la GR.
5	Programar actividades	Gestión de riesgos	Para programar con fecha cada una de las actividades durante la gestión de riesgos.
6	Comunicar resultados	Gestión de riesgos	Para comunicarle a todos los miembros del proyecto los resultados y al equipo de GR cualquier mensaje.
7	Asignar actividades	Gestión de riesgos	Asignarles a los miembros del equipo de GR las actividades correspondientes.
8	Estimar costos	Gestión de riesgos	Estimar los costos que pueden traer consigo los riesgos.

9	Mostrar experiencia	Documentar experiencia	Se muestran las experiencias personales de cada uno de los roles.
10	Seleccionar herramientas o técnicas	Gestión de riesgos	Seleccionar las herramientas o técnicas para la identificación de los riesgos.
11	Gestionar riesgos	Gestión de riesgos	Se gestionan los datos de cada uno de los riesgos identificados.
12	Caracterizar los riesgos	Gestión de riesgos	Caracterizar cada uno de los riesgos identificados.
13	Estimar la exposición al riesgo	Gestión de riesgos	Este es muy importante porque trae consigo el resultado final del riesgo para el proyecto.
14	Mostrar riesgos priorizados	Gestión de riesgos	Muestra el registro de los riesgos priorizados según la exposición.
15	Gestionar estrategia	Gestión de riesgos	Se gestionan los datos de las estrategias de los riesgos del proyecto.
16	Gestionar respuesta	Gestión de riesgos	Se gestionan los datos de las respuestas a las estrategias de los riesgos.
17	Planificar respuesta	Gestión de riesgos	Planificar con fecha las respuestas a las estrategias de los riesgos.
18	Gestionar métricas	Gestión de riesgos	Gestionar los datos de las métricas para evaluar la GR.
19	Mostrar registro de riesgos	Gestión de riesgos	Mostrar el registro de los riesgos.
20	Documentar experiencias	Documentar experiencia	Documentar las experiencias personales de cada uno de los roles.

Tabla 3. Casos de uso del sistema (Elaboración propia).

2.4.3 Diagramas de casos de uso del sistema

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona una respuesta a eventos que se producen en el mismo.

2.4.3.1 Subsistema Documentación de Experiencia.

Este subsistema agrupa los CUS encargados de garantizar el acceso a la información almacenada en el sistema, así como, guardar la nueva.

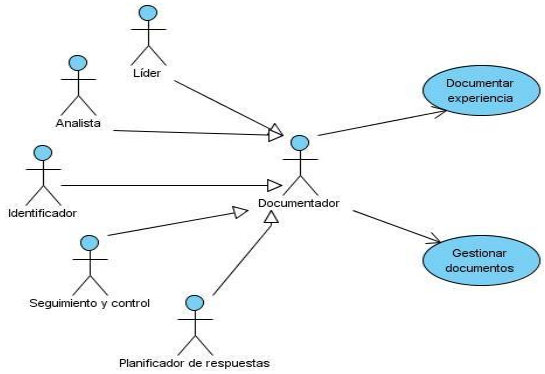


Figura 5. Diagrama de casos de uso del sistema. Subsistema documentación de experiencia (Elaboración propia).

2.4.3.2 Subsistema Gestionar riesgos.

Este subsistema agrupa los CUS encargados de garantizar la identificación y el análisis de la GR.

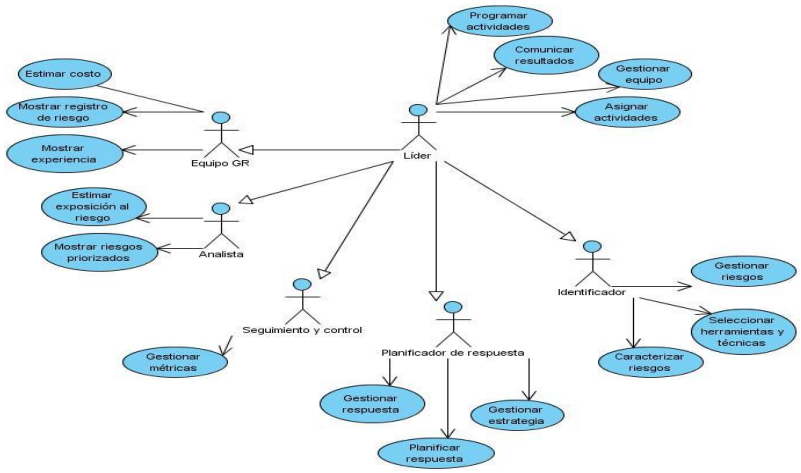


Figura 6. Diagrama de casos de uso del sistema. Subsistema gestionar riesgos (Elaboración propia).

2.4.3.3 Subsistema Seguridad.

Este subsistema agrupa los CUS encargados de garantizar la seguridad del sistema.

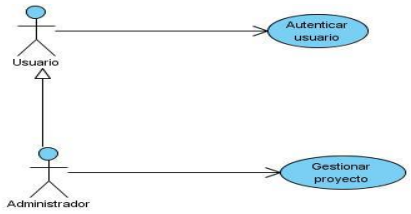


Figura 7. Diagrama de casos de uso del sistema. Subsistema seguridad (Elaboración propia).

2.4.4 Descripción textual de los Casos de Uso.

2.4.4.1 Subsistema Seguridad.

Nombre del CU	Autenticar usuario
Actores	Usuario
Objetivo	Brindar la sesión requerida en dependencia del usuario que esté logueado.

Resumen	El CU se inicia cuando el usuario introduce sus datos del dominio UCI (usuario y contraseña) en el sistema para autenticarse, según los datos introducidos le son asignados sus privilegios y en caso que no sean correctos se le niega el acceso mostrando un mensaje de error.
Referencia	RF1
Precondiciones	Debe ser un usuario que pertenezca al dominio UCI
Pos condiciones	El sistema otorga los permisos correspondientes a cada usuario.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
	1. El sistema muestra la interfaz para la autenticación.
2. El usuario escribe su nombre de usuario y contraseña del dominio UCI.	2.1 El sistema verifica que los datos entrados por el usuario sean correctos.
	2.2 El sistema muestra la información según los privilegios del usuario.
Flujo alterno	
	Si los datos no son correctos el sistema muestra el mensaje de error y se retorna la paso 1.
Prioridad	Crítico
	

Tabla 4. Descripción del CU autenticar usuario (Elaboración propia).

Nombre del CU	Gestionar Proyecto
Actores	Administrador
Objetivo	Gestionar los datos de los proyectos (nombre y líder).
Resumen	El administrador tiene la posibilidad de insertar y modificar cada uno de los proyectos del sistema.
Referencia	RF2
Precondiciones	El administrador debe estar previamente autenticado.

Pos condiciones	El administrador debe haber insertado o modificado al menos un proyecto.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El administrador selecciona la opción "Gestionar proyectos**"** del menú.	2. El sistema muestra una interfaz con la lista de los proyectos y diferentes opciones que le permiten al administrador:
2.1 Insertar, ver sección "Insertar nuevo proyecto"	
2.2 Modificar, ver sección "Modificar proyecto"	
Sección Insertar nuevo proyecto	
3. El administrador toma la opción de insertar nuevo proyecto.	4. El sistema muestra la interfaz para agregar al proyecto.
5. El administrador introduce los datos del nuevo proyecto (nombre y líder) y selecciona la opción “Aceptar”.	6. El sistema valida los datos de entrada.
	7. El sistema almacena la información en su base de datos, muestra el listado con el nuevo proyecto y da la opción de generar un documento con éste.
	8. Retornar al paso 2, culminando así el caso de uso.
Flujo alterno	
	6.1 El sistema solicita los datos del nuevo proyecto ya que hubo un error a la hora de entrar los datos.
	6.2 El sistema retoma el paso 2.
Sección Modificar proyecto	
3. El administrador selecciona la opción de “Modificar proyecto” y elige el proyecto a modificar.	4. El sistema muestra la interfaz para modificar el proyecto.
	5. El sistema modifica los datos seleccionados por el administrador.
	6. El sistema muestra el nuevo listado con los datos del proyecto ya modificados y da la opción de generar un documento con éste.
	7. Retomar paso 2.
Flujo alterno	
3.1 El administrador selecciona la opción de cancelar.	3.1.2 El sistema cancela la operación de modificar proyecto.
	3.1.3 Retomar paso2.
Prioridad	Crítico

Tabla 5. Descripción del CU gestionar proyecto (Elaboración propia).

1.4.4.2 Subsistema Documentar de Experiencia.

La descripción de los CU correspondientes al subsistema documentar experiencia que no se encuentran en este epígrafe se pueden ver en los anexos 1.

Nombre del CU	Gestionar Documento
Actores	Documentador
Objetivo	Gestionar los documentos correspondientes a la GR
Resumen	El documentador tiene la posibilidad de insertar y modificar cada uno de los documentos correspondientes a la GR.
Referencia	RF4
Precondiciones	El documentador debe estar previamente autenticado.
Pos condiciones	El documentador debe haber insertado o modificado al menos un documento.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El documentador selecciona la opción "Gestionar documento **"** del menú.	2. El sistema muestra una interfaz con la lista de los documentos y diferentes opciones que le permiten al equipo de GR:
2.1 Insertar, ver sección "Insertar nuevo documento"	
2.2 Modificar, ver sección "Modificar documento"	
Sección Insertar nuevo documento	
3. El documentador toma la opción de insertar nuevo documento.	4. El sistema muestra la interfaz para agregar el documento.
5. El documentador introduce (sube a la BD) el nuevo documento y selecciona la opción “Aceptar”.	6. El sistema valida los datos de entrada.

	7. El sistema almacena la información en su base de datos.
	8. Se muestra el listado con el nuevo documento
	9. Retornar al paso 2, culminando así el caso de uso.
Flujo alterno	
	6.1 El sistema solicita los datos del nuevo documento ya que hubo un error a la hora de entrar los datos.
	6.2 El sistema retoma el paso 2.
Sección Modificar documento	
3. El documentador selecciona la opción de "Modificar documento" y elige el documento a modificar.	4. El sistema muestra la interfaz para modificar los datos del documento (nombre del documento, proyecto y Roll al que pertenece).
	5. El sistema modifica los datos seleccionados por el documentador.
	6. El sistema muestra el nuevo listado con los datos del documento ya modificados.
	7. Retomar paso 2.
Flujo alterno	
3.1 El documentador selecciona la opción de cancelar.	3.1.2 El sistema cancela la operación de modificar documento.
	3.1.3 Retomar paso2.
Prioridad	Crítico

Tabla 6. Descripción del CU gestionar documento (Elaboración propia).

1.4.4.3 Subsistema Gestionar Riesgo.

La descripción de los CU correspondientes al subsistema gestionar riesgo que no se encuentran en este epígrafe se pueden ver en los anexos 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14.

Nombre del CU	Gestionar Equipo
Actores	Líder
Objetivo	Gestionar los datos de los integrantes del equipo (nombre y roll).
Resumen	El líder tiene la posibilidad de insertar o modificar cada uno de los integrantes del equipo del sistema.
Referencia	RF3
Precondiciones	El líder debe estar previamente autenticado.
Pos condiciones	El líder debe haber insertado o modificado uno o más integrantes del equipo.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El líder selecciona la opción “Gestionar equipo” del menú.	2. El sistema muestra una interfaz con la lista de los integrantes del equipo y diferentes opciones que le permiten al líder:
2.1 Insertar, ver sección “Insertar nuevo integrante”.	
2.2 Modificar, ver sección “Modificar integrante”.	
Sección Insertar nuevo integrante	
3. El líder toma la opción de insertar nuevo integrante del equipo.	4. El sistema muestra la interfaz para agregar al integrante.
5. El líder introduce los datos (nombre y rol que desempeña) del nuevo integrante y selecciona la opción “Aceptar”.	6. El sistema valida los datos de entrada.
	7. El sistema almacena la información en su base de datos.
	8. Muestra el listado con el nuevo integrante y da la opción de generar un documento con éste listado.
	9. Retornar al paso 2, culminando así el caso de uso.
Flujo alterno	
	6.1 El sistema solicita los datos del nuevo integrante ya que hubo un error a la hora de entrar los datos.
	6.2 El sistema retoma el paso 2.
Sección Modificar integrante	
3. El líder selecciona la opción de “Modificar integrante” y elige el que va a modificar.	4. El sistema muestra la interfaz para modificar los datos del integrante.
	5. El sistema modifica los datos seleccionados por el líder.

	6. El sistema muestra el nuevo listado con los datos del integrante ya modificados y da la opción de generar un documento con éste listado.
	7. Retomar paso 2.
Flujo alterno	
3.1 El administrador selecciona la opción de cancelar.	3.1.2 El sistema cancela la operación de modificar integrante.
	3.1.3 Retomar paso2.
Prioridad	Crítico

Tabla 7. Descripción del CU gestionar equipo (Elaboración propia).

Nombre del CU	Realizar análisis
Actores	Analista
Objetivo	Realizar el análisis cualitativo a los riesgos.
Resumen	El caso de uso se inicia cuando el analista solicita realizar el análisis cualitativo a un riesgo, el sistema muestra un listado con los riesgos que no se les ha realizado dicho análisis, el analista selecciona uno e introduce los valores de probabilidad e impacto al sistema.
Referencia	RF13
Precondiciones	El analista debe estar previamente autenticado. El listado de riesgos no atendidos debe tener al menos un riesgo.
Pos condiciones	Queda almacenada la probabilidad y el impacto de al menos un riesgo, teniendo como resultado su exposición.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El analista selecciona la opción “Análisis cualitativo” del menú.	2. El sistema muestra el listado de los riesgos a los que no se les ha hecho análisis cualitativo.
3. El analista selecciona el riesgo al que va a insertarle la probabilidad e impacto.	4. El sistema muestra la interfaz para insertar los datos al riesgo.

5. El analista introduce los datos solicitados y selecciona la opción “Aceptar”.	6. El sistema valida los datos y almacena la información en su base de datos.
	7. El sistema muestra la interfaz con la exposición al riesgo y da la oportunidad de generar un documento con estos datos.
	8. Retomar acción 2.
Flujo alterno	
5.1 El analista desea cancelar el llenado de los datos, para esto oprime el botón “Cancelar”.	5.2 El sistema cancela la operación de realizar análisis.
Prioridad	Crítico
	

Tabla 8. Descripción del CU realizar análisis (Elaboración propia).

2.4.5 Patrones de Casos de Uso

En este epígrafe se expondrá los patrones de CU, con un enfoque más práctico, centrados en los que se utilizaron para lograr el modelo de CU, artefacto perteneciente al Flujo de Trabajo de Requerimientos.

Los patrones de utilizados son:

CRUD (Creating, Reading, Updating, Deleting): Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Parcial: El patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras.

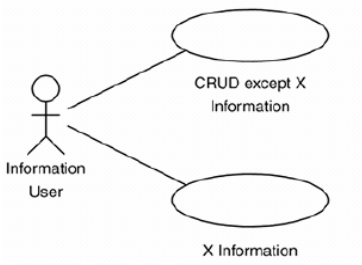


Figura 8. CRUD Parcial (Palmkvist, 2004).

2.5 Conclusiones parciales

En este capítulo se dio paso a desarrollar la propuesta de solución planteada, obteniéndose las funcionalidades que debe poseer el sistema, las cuales se representaron mediante Diagramas de Casos de Uso describiéndose, paso a paso, todas las acciones de los actores del sistema con los casos de uso con los que interactúan. Se identificaron los actores. Teniendo en cuenta todas estas características se puede comenzar a construir el sistema poniendo en práctica el cumplimiento de los requisitos tanto funcionales como no funcionales planteados en este capítulo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA MoGeRisk.

3.1 Introducción

En este capítulo se expondrá lo referente al análisis y diseño del sistema de GR MoGeRisk, correspondiente al flujo de Análisis y Diseño de la metodología RUP, metodología empleada para la modelación del sistema. Se realizan los diagramas de clases del análisis y los diagramas de clases del diseño, teniendo en cuenta los patrones de diseño utilizados.

3.2 Análisis

3.2.1 Modelo de análisis

El Modelo de Análisis contribuye a refinar los requerimientos y permite razonar sobre los aspectos internos del sistema y se considera como la primera aproximación al modelo de diseño. El Modelo de Análisis puede o no realizarse para construcción de un sistema, RUP ofrece tres maneras de emplear el modelo de análisis en la construcción de un sistema:

1. El proyecto utiliza el modelo de análisis para describir los resultados del análisis y mantiene la consistencia de este a lo largo del ciclo de vida del proyecto.

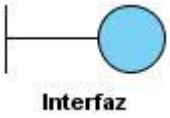
2. El proyecto utiliza el modelo de análisis para describir los resultados del análisis pero considera a este como una herramienta intermedia (es decir de mayor interés durante la Fase de Elaboración).

3. EL proyecto no utiliza el modelo de análisis para describir los resultados del análisis.

Para la modelación del sistema en cuestión se adoptado la segunda manera de emplear el modelo de análisis, por tal motivo luego de pasar a la Fase de Construcción se deja de actualizar el modelo de análisis y cualquier cambio que ocurra en el funcionamiento interno del sistema se resuelve como parte integrada al modelo de diseño.

3.2.2 Clases del análisis

Representan abstracciones de una o varias clases, las cuales se caracterizan por centrarse en el tratamiento de los requisitos funcionales. Las clases del análisis siempre encajan con tres estereotipos básicos:



Las clases interfaz se utilizan para modelar la interacción entre el sistema y sus actores. Al modelar partes del sistema que implican a sus actores implica que clarifican y reúnen los requisitos en los límites del sistema. Representan abstracciones de de ventanas, formularios, paneles, interfaces de comunicaciones, terminales, entre otros.

Las clases controladoras (control) representan coordinaciones, secuencias, transacciones, y control de objetos y se utilizan con frecuencia para encapsular el control de un caso de uso en concreto. También se utilizan para representar derivaciones y cálculos, como la lógica de negocio, que no pueden asociarse con ninguna información concreta, de larga duración, almacenada por el sistema.

Las clases entidades se utilizan para modelar información que posee una vida larga y que es a menudo persistente. La mayoría de estas clases se derivan de una entidad del negocio.

Cada estereotipo implica una semántica específica, lo cual constituye un método potente y consistente de identificar y describir las clases del análisis.

3.2.2.1 Diagramas de clases del análisis.

Subsistema Seguridad

Figura 9 . Diagrama de clases del análisis. Autenticar usuario (Elaboración propia).

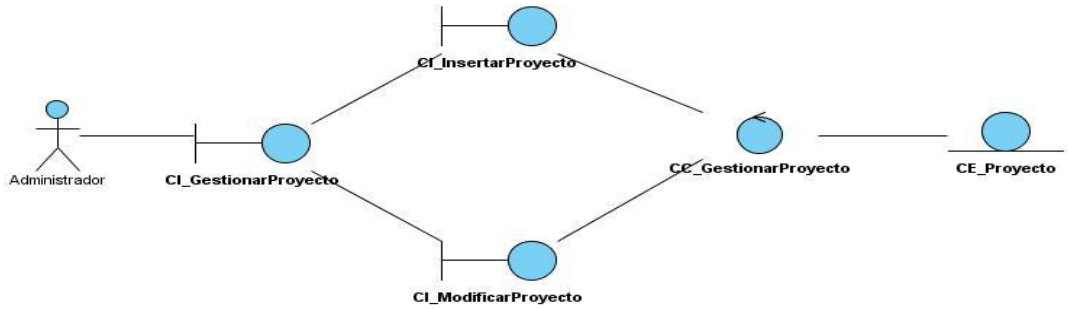


Figura 10. Diagrama de clases del análisis. Gestionar proyecto (Elaboración propia).

Subsistema Documentar Experiencia. El diagrama de clases del análisis, correspondiente al subsistema documentar experiencia que no se encuentra en este epígrafe se puede ver en el anexo 15.

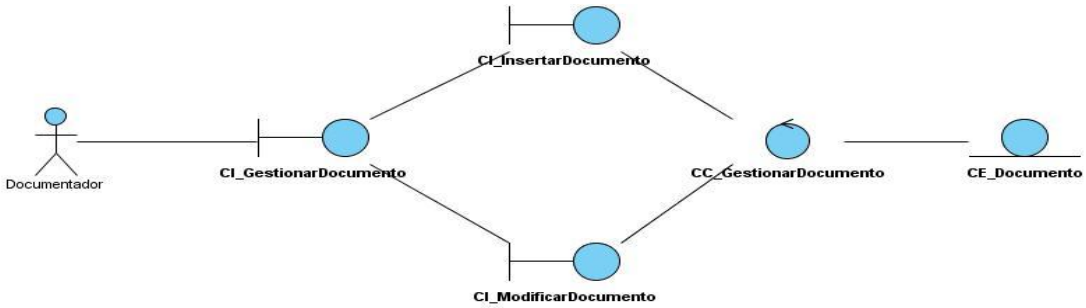


Figura 11. Diagrama de clases del análisis. Gestionar documento (Elaboración propia).

Subsistema Gestionar Riesgos. Los diagramas de clases del análisis, correspondientes al subsistema gestionar riesgos que no se encuentran en este epígrafe se pueden ver en los anexos 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27 y 28.

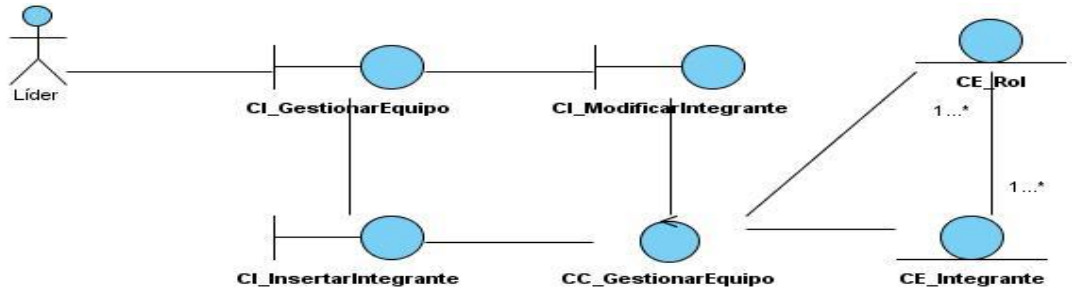


Figura 12. Diagrama de clases del análisis. Gestionar equipo (Elaboración propia).

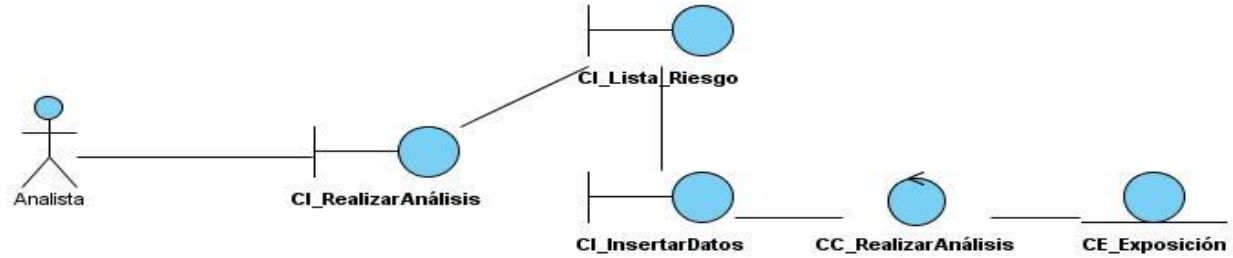


Figura 13. Diagrama de clases del análisis. Realizar análisis (Elaboración propia).

3.3 Diseño

3.3.1 Modelo de diseño

El Modelo de Diseño es un modelo que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, unidas a ciertas restricciones del dominio del negocio y por consiguiente de la aplicación. Según RUP el modelo de diseño se comienza a construir al final de la fase de Elaboración y se culmina al comienzo de la fase de Construcción, lo cual contribuye a lograr una arquitectura estable, que posibilita crear un plano del Modelo de Implementación.

Es importante aclarar que para comenzar las tareas del diseño es necesario tener definida la Línea Base de la Arquitectura, no así la del Modelo de Análisis, aunque esto puede variar según el modelo de desarrollo con que se trabaje. Además si va a trabajar usando Base de Datos, la modelación de la misma debe estar en una versión la cual no debe tener cambios, cuando iniciamos el diseño.

En el caso especifico del sistema a modelar, se tomó el modelo de análisis como una herramienta intermedia (Ver Epígrafe 3.2.1), por lo cual se comenzó a modelar el diseño luego de tener el análisis

culminado, esto puede demorar el proceso de desarrollo en la fase de elaboración pero posibilita que cuando se comience la fase de Construcción no tengamos que regresar a refinar el Modelo de Análisis y nos concentramos en el flujo de diseño.

3.3.2 Extensiones UML para Diseño Web

Como se expuso en el capítulo dos de este trabajo, el lenguaje de modelado utilizado es UML el cual define extensiones para el modelado de aplicaciones web:

De: / A:	Client Page	Server Page	Form	Prototipos
Client Page	*Link Redirect*	*Link*	*Aggregation*	
Form	*Builds Redirect*	*Redirect*		
Server Page	*Aggregated by*	*Submit*		

Tabla 9. Extensiones UML para Diseño Web (Elaboración propia).

3.3.2.1 Diagramas de clases del Diseño

Subsistema Seguridad

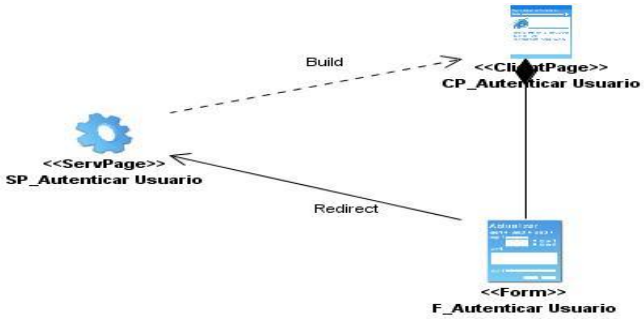


Figura 14. Diagrama de clases del diseño. Autenticar Usuario (Elaboración propia).

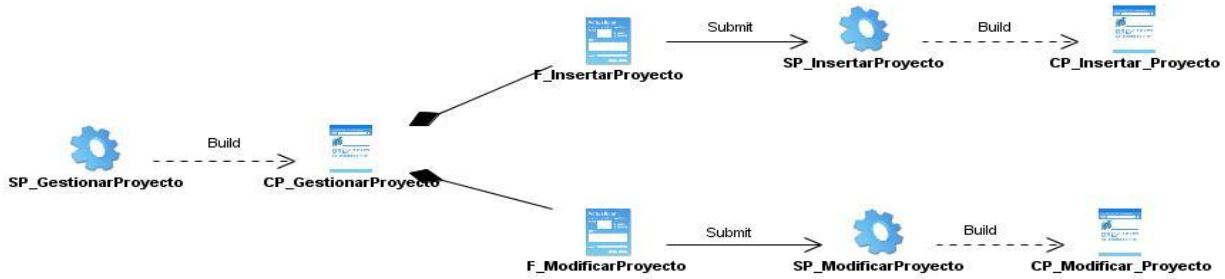


Figura 15. Diagrama de clases del diseño. Gestionar proyecto (Elaboración propia).

Subsistema Documentar Experiencia. El diagrama de clases del diseño, correspondiente al subsistema documentar experiencia que no se encuentra en este epígrafe se puede ver en el anexo 29.

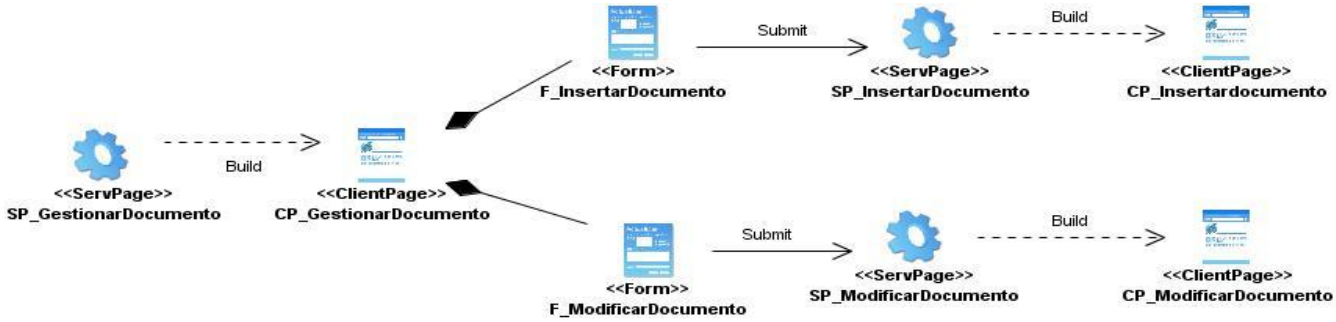


Figura 16. Diagrama de clases del diseño. Gestionar documento (Elaboración propia).

Subsistema Gestionar Riesgos. Los diagramas de clases del diseño, correspondientes al subsistema gestionar riesgos que no se encuentran en este epígrafe se pueden ver en los anexos 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 42.

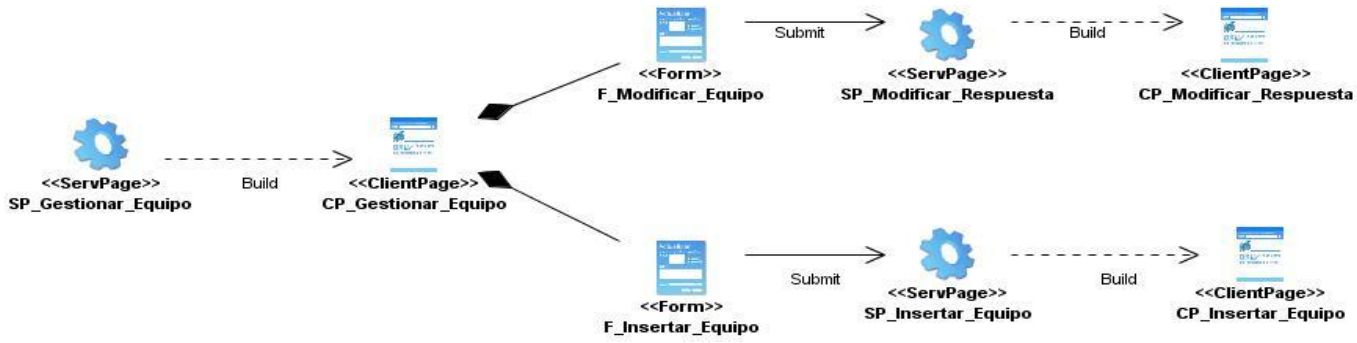


Figura 17. Diagrama de clases del diseño. Gestionar equipo (Elaboración propia).

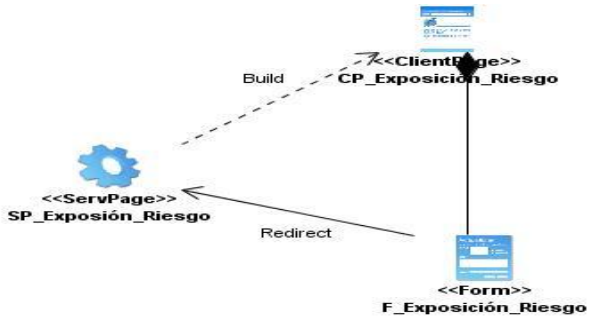


Figura 18. Diagrama de clases del diseño. Realizar análisis (Elaboración propia).

3.4 Patrones de Diseño utilizados.

Dentro de los muchos patrones que existen están los patrones de diseño y en este epígrafe se exponen cuales fueron usados para el modelado de esta aplicación.

Patrones GRASP

- Experto

El patrón de diseño experto está diseñado en que la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras se están considerando los mismos aspectos del sistema: Lógica de negocio, Persistencia a la base de datos y Interfaz de usuario.

- Controlador

El patrón de diseño controlador está diseñado para asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

- Alta Cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase, además se caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Por tanto la aplicación de este patrón posibilita, que las clases sean más fáciles de comprender, de reutilizar y conservar.

- Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras, con la que conoce y con que concurre a ellas. El bajo acoplamiento posibilita que los cambios locales en una clase, no ocasionen cambios en otras; sean más fáciles de entender cuando estén aisladas, por lo que son más fáciles de reutilizar.

Importante destacar que estos cuatro patrones no se pueden ver por separado, si no como un todo que posibilita la asignación de responsabilidades.

Patrones GoF *(Gang of Four)*

Patrones creacionales

- Abstract Factory *(Fábrica abstracta)*

Con este patrón creacional se proporciona una interfaz para la creación de familias de objetos relacionados sin especificar sus clases concretas. Se utiliza porque el sistema es independiente de cómo se crean sus objetos, este patrón se encarga de proporcionar el objeto creado y la aplicación lo utiliza sin necesidad de conocer como se realizó el proceso de creación de dicho objeto. Este objeto es el que le permite al sistema saber la configuración que deberá tener la interfaz a visualizar según los permisos y roles del usuario que se ha autenticado.

Patrones Estructurales

- Facade *(Fachada)*

Fachada es un patrón estructural que tiene como objetivo establecer una interfaz simple y fácil de comprender para los usuarios. En dicha interfaz se añaden las funcionalidades más utilizadas y se habilita

un punto de entrada para unificar el subsistema en cuestión. La función de este patrón en la aplicación a desarrollar es mediar entre la interfaz y las llamadas al servidor. Es decir, encapsula los pedidos AJAX del servidor, los envía al controlador frontal.

Front-Controller *(Controlador Frontal)* (Fowler, 2003)

Se recomienda utilizar este patrón porque con el todas las peticiones tienen que pasar por un servlet Controlador. El controlador proporciona un punto de entrada único que controla y gestiona las peticiones Web realizadas por los clientes. Teniendo este único punto de entrada se evita tener que repetir la misma lógica de control en todos los .jsp.

Normalmente se utiliza junto con un Dispatcher que significa que la aplicación delega a la capa controladora la petición del usuario desde el navegador Web. Este es responsable de redirigir el flujo de ejecución hacia el .jsp adecuado. Este Dispatcher puede ser realizado por el propio controlador o estar en una clase a parte. El Patrón de Diseño Front Controller funciona como fichero de punto de entrada único a la aplicación.

3.5 Conclusiones parciales

En este capítulo se trató el resultado de este trabajo, el análisis y diseño del sistema propuesto, donde se abordaron aspectos teóricos de los Modelos de Análisis y Diseño, los artefactos generados, así como los patrones de diseño empleados.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

4.1 Introducción

La medición es fundamental para cualquier disciplina de ingeniería, permite tener una visión más profunda de lo que se evalúa o se mide y la Ingeniería del Software (IS) no está ajena. Existen varios tipos de métricas que pueden utilizarse en la en la realización de proyectos de software para gestionar, predecir y mejorar la calidad de software. La finalidad del uso de métricas es evaluar sistemas para conseguir alta calidad y robustez. En este capítulo se aplican métricas para medir la calidad del diseño, inspiradas en el estudio de la calidad del DOO.

4.2 Métricas para validar el diseño

A continuación se describe un conjunto representativo de métricas tratando de abordar todos los posibles niveles y características en sistemas orientado a objetos basado en un marco de referencia definido por (Rodríguez y Harrison, 1999).

4.2.1 Métricas a nivel de sistema

El conjunto de métricas MOOD (Metrics for Object oriented Design) opera a nivel de sistema. Se refieren a mecanismos estructurales básicos en el paradigma de la orientación a objetos como encapsulación (MHF y AHF), herencia (MIF y AIF), polimorfismo (PF) y paso de mensajes (COF). En general, las métricas a nivel de sistema pueden derivarse de otras métricas usando métodos estadísticos como la media, etc. Éstas son utilizadas para identificar características del sistema

4.2.1.1 Proporción de métodos heredados. *(Method Inheritance Factor –MIF-)*

MIF se define como proporción de la suma de todos los métodos heredados en todas las clases entre el número total de métodos (localmente definidos más los heredados) en todas las clases.

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

Donde:

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

Y:

Md (CI) es el número de métodos declarados en una clase.

Ma (CI) es el número de métodos que pueden ser invocados en relación a Ci.

Mi (Ci) es el número de métodos heredados (y no redefinidos Ci).

TC es el número total de clases en el sistema.

Sus autores proponen a MIF como una medida de la herencia y como consecuencia, una medida del nivel de reúso.

4.2.1.2 Proporción de atributos heredados. *(Attribute Inheritance Factor –AIF-)*

AIF se define como la proporción del número de atributos heredados entre el número total de atributos.

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

Donde:

$$A_a(C_i) = A_d(C_i) + A_i(C_i)$$

Y:

Ad (Ci) es el número de atributos declarados en una clase.

Aa (Ci) es el número de atributos que pueden ser invocados asociados a Ci.

Ai (Ci) es el número total de atributos heredados (y no redefinidos) en Ci.

TC es el número total de clases en el sistema.

Al igual que MIF, AIF se considera un medio para expresar el nivel de reusabilidad en un sistema.

4.2.1.3 Proporción de polimorfismo. *(Polymorphism Factor –PF-)*

PF se define como la proporción entre el número real de posibles diferentes situaciones polimórficas para una clase Ci entre el máximo número posible de situaciones polimórficas en Ci. En otras palabras, el número de métodos heredados redefinidos dividido entre el máximo número de situaciones polimórficas distintas.

$$PF = \frac{\sum_{i=1}^{TC} M_0(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

Donde:

$$M_d(C_i) = M_n(C_i) + M_o(C_i)$$

Y:

Mn (Ci) es el número de métodos nuevos.

Mo (Ci) es el número de métodos redefinidos.

DC (Ci) es el número de descendientes de Ci.

TC es el número total de clases en el sistema.

PF es una medida del polimorfismo y una medida indirecta de la asociación dinámica en un sistema. El polimorfismo es debido a la herencia. Esta métrica no cumple todas las propiedades definidas para ser válida ya que en un sistema sin herencia, el valor de PF resulta indefinido, exhibiendo una discontinuidad.

4.2.2 Métricas a nivel de acoplamiento

Acoplamiento es el uso de métodos o atributos definidos en una clase que son usados por otra. Las clases tienen que interactuar entre ellas para formar sistemas, y esta interacción puede indicar su complejidad. Métricas representativas de acoplamiento son las siguientes:

4.2.2.1 Acoplamiento entre objetos. *(Coupling Between Objects –CBO-)*

CBO de una clase es el número de clases a las cuales una clase está ligada. Se da dependencia entre dos clases cuando una clase usa métodos o variables de la otra clase. Las clases relacionadas por herencia no se tienen en cuenta.

Las clases deberían de ser lo más independientes posible, y aunque siempre es necesaria una dependencia entre clases, cuando ésta es grande, la reutilización puede ser más cara que la reescritura. Al reducir el acoplamiento se reduce la complejidad, se mejora la modularidad y se promueve la encapsulación.

4.2.3 Métricas a nivel de clases

Este conjunto de métricas identifica características dentro de las clases, destacando diferentes aspectos de sus abstracciones y ayudando a descubrir clases que podrían necesitar ser rediseñadas.

4.2.3.1 Respuesta de una clase. *(Response For a Class –RFC-)*

RFC es el cardinal del conjunto de todos los métodos que pueden ser invocados en respuesta a un mensaje a un objeto de la clase o por alguno método en la clase. Esto incluye todos los métodos accesibles dentro de la jerarquía de la clase. En otras palabras, cuenta las ocurrencias de llamadas a otras clases desde una clase particular.

RS RFC = donde RS es el conjunto respuesta para la clase.

El conjunto respuesta para la clase puede ser expresado de la siguiente manera:

$$RS = \{M\} \bigcup_i \{R_i\}$$

Donde:

{Ri} es el conjunto de métodos llamados por el método i; y {M} es el conjunto de todos los métodos en la clase.

RFC es una medida de la complejidad de una clase a través del número de métodos y de la comunicación con otras clases, ya que incluye todos los métodos llamados desde fuera de la clase. Cuanto mayor es RFC, más complejidad tiene el sistema ya que se puede invocar un mayor número de métodos como repuesta a un mensaje.

4.2.3.2 Falta de cohesión en los métodos. *(Lack of Cohesion in Methods –LCOM-)*

LCOM establece en qué medida los métodos hacen referencia a atributos.

Considérese una clase C1 con n métodos M1, M2,…, Mn. Sea {Ij} = el conjunto de variables instancias por el método Mi.

Hay n conjuntos tales que {I1},... {In}; Sea P = {(Ii, Ij) | Ii ∩ Ij = ∅}, y Q = {(Ii, Ij) | Ii ∩ Ij_ ∅}. Si todos los conjuntos n {I1},... {In} son ∅, entonces P = ∅.

$$LCOM = \begin{cases} |P| - |Q| & si\ |P| > |Q| \\ 0 & \text{caso contrario} \end{cases}$$

LCOM es una medida de la cohesión de una clase midiendo el número de atributos comunes usados por diferentes métodos, indicando la calidad de la abstracción hecha en la clase. Un valor alto de LCOM implica falta de cohesión, es decir, escasa similitud de los métodos. Problemas con esta métrica, dos clases pueden tener LCOM=0, mientras una tiene más variables comunes que la otra y no se dan guías para la interpretación de esta métrica.

Por lo tanto, se ha sugerido una nueva medida para LCOM, LCOM*:

Considérese un conjunto de métodos {Mi} (i=1,…, m) accediendo a un conjunto de atributos {Aj} (j=1,…, a) y el número de métodos que acceden a cada atributo como μ (Aj). Entonces se define LCOM* como:

$$LCOM^* = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1 - m}$$

Esta métrica sólo puede calcularse cuando m>1.

Cuando todos los métodos acceden a todos los atributos, entonces Σμ (Aj) = ma, y por lo tanto LCOM* = 0. Esto indica una cohesión perfecta. Valores cercanos a 0 indican que la mayoría de los métodos accede a la mayoría de las instancias. Por el contrario, cuando cada método accede solo a un atributo, entonces Σ μ (Aj) = a y por lo tanto LCOM* = 1, lo cual indica una falta de cohesión.

4.2.4 Métricas a nivel de métodos

Métodos y atributos se encuentran en el nivel de granularidad más bajo, por lo que las características son bien conocidas por los métodos tradicionales. A este nivel es donde se suelen aplicar las métricas tradicionales como Líneas de Código, cyclomatic complexity, etc. Aunque en los SOO estas métricas no son tan importantes ya que los métodos tienden a distribuir la complejidad entre las llamadas a objetos más que dentro de los propios métodos.

4.2.4.1 Líneas de código. *(Lines of Code per method –LOC-)*

LOC es el número de líneas activas de código (líneas ejecutables) en un método. El tamaño de un método es usado para evaluar la comprensibilidad, reusabilidad y mantenibilidad del código.

Los umbrales para evaluar el tamaño dependen del lenguaje de programación y de la complejidad de los métodos. En sistemas OO el número de líneas de código de los métodos debería ser bajo. Se sugiere un umbral de 24 LOC para métodos en C++ y de 8 en Smalltalk para descubrir que métodos son candidatos a ser divididos en varios.

No es una métrica recomendada para sistemas OO, pero es fácil de recoger y utilizar. Es fácil ver que LOC para la misma funcionalidad, varían con el lenguaje de programación y el estándar de codificación utilizado.

4.2.4.2 Número de mensajes enviados. *(Number of Messages Send –NOM-)*

NOM mide el número de mensajes enviados en un método, segregados por el tipo de mensaje. Los tipos incluyen:

- Unarios, mensajes sin argumentos.

- Binarios, mensajes con un argumento que pertenecen a tipos especiales (por ejemplo concatenación y funciones matemáticas).

- Clave, mensajes con uno o más argumentos.

NOM cuantifica el tamaño del método de una manera relativamente no sesgada. Se sugiere un umbral de 9. Un valor alto puede indicar un estilo funcional y/o una colocación pobre de responsabilidades. Lenguajes como C++ pueden llamar a subsistemas no OO y estas llamadas, no deberían ser contadas en el número de mensajes enviados.

4.3 Métricas que se utilizaron

Para medir la calidad del diseño se utilizaron métricas básicas inspiradas en el estudio de la calidad del DOO referenciadas por (Presman, 2005) teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez miden los principales atributos de calidad de software. Siendo esto la principal razón por la que se decide utilizar las métricas, Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC), propuestas por Presman. A demás que las métricas anteriormente analizadas no cuentan con una bibliografía actual y se comentan también algunos de los problemas que se encuentran al analizar las métricas, como definiciones ambiguas que dan lugar a varias interpretaciones, la falta de concordancia para el propósito que fueron concebidas e incluso la falta de validez teórica.

Atributos de calidad que se miden:

- Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización. Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- Complejidad del mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

4.3.1 Tamaño Operacional de la Clase.

El TOC está dado por el número de métodos asignados a una clase.

Atributo que afecta	Modo en que afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica que disminuya el grado de reutilización de la clase.

Tabla 10. Tamaño Operacional de la Clase (Elaboración propia).

Responsabilidad	
Criterio	Cantidad de clases
Bueno	30
Medio	56
Alto	22

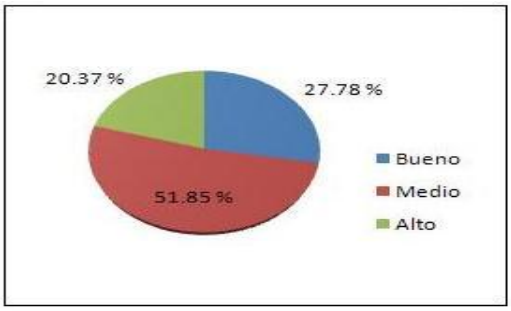


Figura 19. Gráfico que muestra la Responsabilidad en la métrica TOC (Elaboración propia).

Complejidad de implementación	
Criterio	Cantidad de clases
Bueno	82
Medio	12
Alta	14

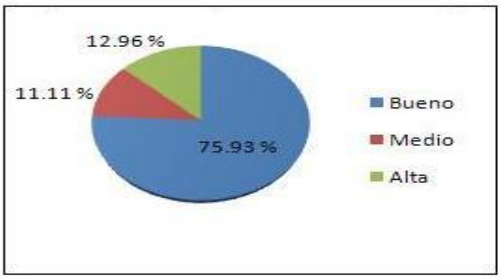


Figura 20. Gráfico que muestra la Complejidad de Implementación en la métrica TOC (Elaboración propia).

Reutilización	
Criterio	Cantidad de clases
Bueno	52
Medio	56
Alto	0

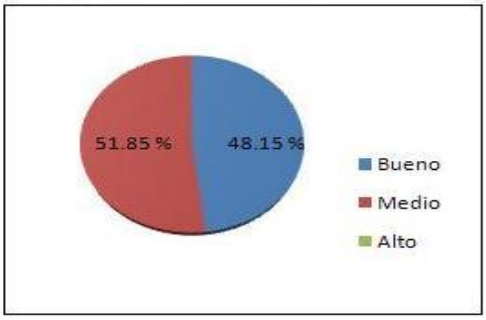


Figura 21. Gráfico que muestra la Reutilización en la métrica TOC (Elaboración propia).

4.3.2 Relaciones de Clases

Esta métrica está dada por la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. Se le aplica a las mismas clases que le fue aplicada la métrica TOC. Los aspectos de calidad que se miden son: Acoplamiento, Complejidad de mantenimiento y Cantidad de pruebas (Jacobson, 2004).

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 11. Relaciones entre Clases (Elaboración propia).

Acoplamiento	
Criterio	Cantidad de clases
Ninguno	52
Bajo	20
Medio	36
Alto	0

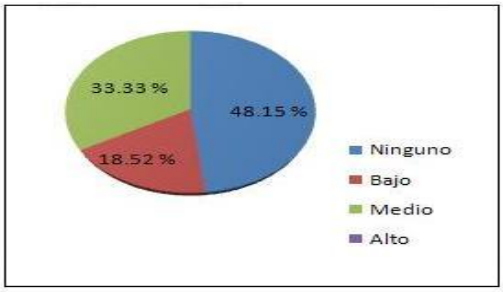


Figura 22. Gráfico que muestra el Acoplamiento en la métrica RC (Elaboración propia).

Complejidad de mantenimiento	
Criterio	Cantidad de clases
Baja	72
Media	36
Alta	0

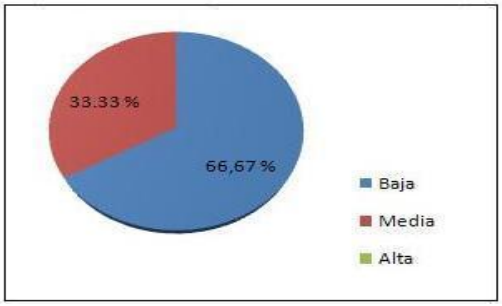


Figura 23. Gráfico que muestra la Complejidad de Mantenimiento en la métrica RC (Elaboración propia).

Cantidad de pruebas	
Criterio	Cantidad de clases
Baja	81
Media	27
Alta	0

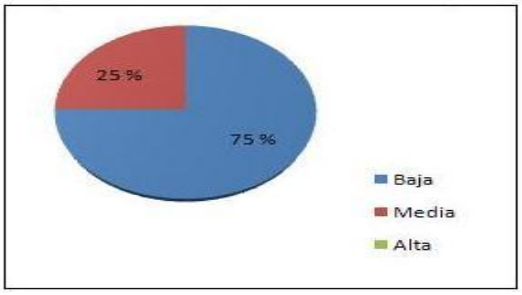


Figura 24. Gráfico que muestra la Cantidad de Pruebas en la métrica RC (Elaboración propia).

4.4 Resumen de los resultados de la evaluación.

A continuación se exponen algunos gráficos que muestran un resumen de las métricas utilizadas para la validación del diseño propuesto.

Atributos	TOC	RC	Total
Responsabilidad	0,2		0,2
Cantidad de pruebas		0,25	0,25
Reutilización	0,52		0,52
Acoplamiento		0,33	0,33
Complejidad de implementación	0,13		0,13
Complejidad de mantenimiento		0,33	0,33

Tabla 12. Resumen de los resultados alcanzados en la aplicación de las métricas para validar el diseño (Elaboración propia).

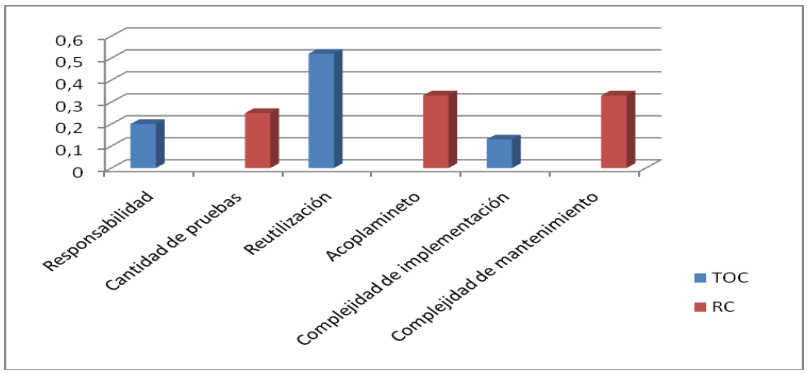


Figura 25. Gráfico que muestra un resumen de los resultados alcanzados después de aplicar las métricas para validar el diseño (Elaboración propia).

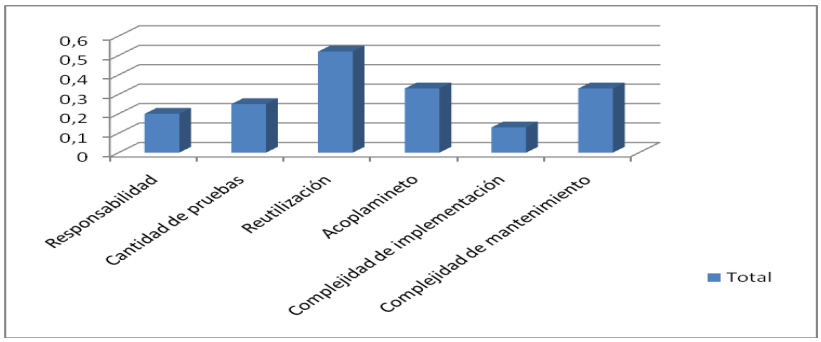


Figura 26. Gráfico que muestra un resumen total de los resultados de los atributos de calidad aplicados (Elaboración propia).

El resultado de la aplicación de estas métricas demuestra que por lo general las clases no están cargadas en responsabilidad, existe bajo acoplamiento entre las mismas; así como un nivel medio de reutilización. Indican además que el diseño no es complejo, son bajas la complejidad de mantenimiento y la complejidad en las pruebas. Estos resultados confirman la calidad del diseño.

4.5 Conclusiones parciales

En este capítulo se ha hecho un análisis de los resultados obtenidos en este trabajo. Para la validación se analizaron diferentes métricas decidiendo aplicar otras como el TOC y la RC obteniéndose resultados satisfactorios, midiéndose por los atributos anteriormente relacionados.

CONCLUSIONES

Al término de este trabajo se puede arribar a las siguientes conclusiones:

- Con el estudio de los procesos del modelo MoGeRi, se obtuvo el conocimiento necesario para identificar los procesos de negocio.
- El análisis de varios sistemas informáticos que implementan y soportan la GR y la selección correcta de la metodología de software, permitió la modelación considerada del dominio y la captura precisa de los requisitos teniendo en cuenta las necesidades del cliente.
- La modelación efectiva del dominio permitió la construcción del diseño de manera tal que respondiera a las características específicas del mismo.
- La acertada utilización de la herramienta CASE permitió la correcta representación del sistema a través de modelos sustentados por UML.
- La aplicación de buenas prácticas de RUP en las Fases de Elaboración y Construcción permitirá al resto del equipo de trabajo lograr implementar las funcionalidades críticas del sistema.
- Se logra un diseño que cumpla las restricciones del Framework de desarrollo, aplicando patrones de diseño para la óptima construcción del mismo.
- Se identifican de manera temprana subsistemas, que pueden convertirse en componentes reutilizables para aplicaciones de similares características.
- De forma general los resultados obtenidos son positivos, tomando como referencia la aplicación de métricas y atributos para validar la solución.

RECOMENDACIONES

Desde el propio comienzo de la investigación se detectó la importancia de un sistema que soporte e implemente la GR basado en un modelo que se ajuste a las condiciones reales de producción de la universidad, por tanto se recomienda:

- Realizar la implementación de la propuesta que se presenta en este trabajo, con el fin de obtener al menos una versión del producto.
- Que una vez terminado el producto se adjunte como un módulo de la herramienta de GP utilizada en la UCI RedMine, para comprobar su rendimiento y uso del usuario con la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. Jacobson, Ivar, Booch, Grady y Rumbauch, James. *El Proceso Unificado de Desarrollo de Software.* La Habana : Félix Varela, 2004.

2. Presman, Roger S. *Ingeniería de Sofware. Un Enfoque Práctico.* La Habana : Felix Varela, 2005.

3. Cortés, Joel Iván Rea. *Venus: Construcción de una herramienta I-CASE para diseño OO, y su Entorno de Apoyo a Proyectos Integrado (EAPI) .* México, Universidad de las Américas Puebla : s.n., 2004. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rea_c_ji/

4. Pérez, Javier Eguíluz. *Introducción a AJAX.* 2008. http://www.librosweb.es/ajax.

5. Álvarez, M. A. *Lenguaje de Marcas: HTML.* 2006. http://www.webestilo.com/html/cap1a.phtml.

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

7. PMBoK, G. D. *Guia de los fundamentos de la Dirección de Proyectos (Guia del PMBoK).* Project Management Institute, Four Campus Boulevard, Newtown Square : s.n., 2004.

8. Zulueta, Yeleny. *Tesis Modelo de Gestión de Riesgos en Proyectos de Desarrollo de Software.* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n., 2007.

9. SEI. *Continuous Risk Management Guidebook.* Software Engineering Institute, Carnegie Mellon University : s.n., 2004.

10. Cano, R. *Gestión de los riesgos en el Proyecto "A Jugar".* La Habana, Universidad de las Ciencias Informáticas : s.n., 2008.

11. Martin S. Feather, Steven L. Cornford, Leila Meshkat, Luke Voss. *OVECOMING OBSTACLES TO THE EXCHANGE OF INFORMATION BETWEEN RISK TOOLS.* Jet Propulsion Laboratory, California Institute of Technology : 4800 Oak Grove Drive, Pasadena, CA 91109-8099, 2007.

12. Solenzal, C. G. *Gestión de los riesgos en el Proyecto "A Jugar".* La Habana, Universidad Ciencias Informáticas : s.n., 2008.

13. Charette, R. N. *Software Engineering Risk Analysis and Management, Mc Graw-Hill/ Intertext.* 1989.

14. Navarro, M. E. *PLAN DE ADMINISTRACIÓN DEl RIESGO INFORMÁTICO.* 2007.

15. Rosenberg H. T, A. Gallo. *Continuous Risk Management at NASA. Software Management Conference.* 1999.

16. León, Mireydis Orellana. *"Análisis y Diseño de los Procesos de Gestión de Personal para Akademos v2.0.".* Ciudadad Habana, Universidad de las Ciencias Informáticas : s.n., 2009.

17. Casañola, Yaimí Trujillo. *Modelo de Factoría de Software aplicando inteligencia.* Ciudad habana : s.n., 2007.

18. Mendoza, Yusdenys Pérez. *Propuesta de técnicas de apoyo a las decisiones para la gestión de riesgos.* Ciudad de la Habana, Universidad de la Ciencias Informáticas. : s.n., 2009.

19. Cuevas, David Tavares. *Análisis y Diseño del sistema de manejo integral de perforación de pozos.* Ciudad Habana, Universidad de la Ciencias Informáticas : s.n., 2009.

20. ITDG, S P. *Enfoque de gestión de riesgos.* 2008.

21. Lavell, A. *Sustentos Teórico – conceptuales sobre el riesgo y la Gestión Local del Riesgo en el marco del desarrollo .* 2003.

22. Alba, P. *Guía para la gestión de riesgos a través de RUP.* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n., 2008.

23. Pérez. *Guía para la gestión de riesgos a través de RUP.* Ciudad de la Habana., Universidad de las Ciencias Informáticas : s.n., 2008.

24. Ministerio de administraciones públicas**.** *MAGERIT. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información .* 2006.

25. Seijo, R Y. *Gestión de Riesgos en el Proyecto de Informatización del Conocimiento Geológico en Cuba.* Ciudad de la Habana. Universidad de las Ciencias Informáticas : s.n., 2007.

26. García. *Gestión de Riesgos en el Proyecto de Informatización del Conocimiento Geológico en Cuba.* Ciudad de la Habana. Universidad de las Ciencias Informáticas : s.n., 2007.

27. PALISADE. 2010. www.palisade-lta.com

28. AXIS. *Aplicaciones de Análisis de Riesgos.* 2005.

29. ISSC. *IRisk Management Tools Introduction.* 2008.

30. Niasi, L S D. *Identificación de Riesgos de Proyectos de Software en Base a Taxonomías.* Córdova : s.n., 2005.

31. Mañas, J A. *PILAR. Herramientas para el Análisis y la Gestión de Riesgos.Comunicación.* 2004.

32. Sánchez, Alain. *Análisis y diseño del módulo Emisión de Carta de Crédito del subsitema Comercio Exterior del proyecto Modernización del Sistema Bancario Cubano.* Ciudad de la Habana. Universidad de las Ciencias Informáticas : s.n., 2009.

33. Rosales, Carlos Luis Serrano. *Desarrollo de Biblioteca de Métodos Numéricos (BMN), referente a Sistemas de Ecuaciones Lineales, Sistemas de Gran Dimensión y Poco Densos e Integración Numérica.* Ciudad Habana. Universidad de las Ciencias Informáticas : s.n., 2008.

34. Zaragoza, Mercedes Escobar. *Aplicación y Mejora del Modelo de Gestión de Riesgos MoGeRi.* Ciudad de la Habana. Universidad de las Ciencias Informáticas**. :** s.n., 2009.

35. Eyssautier de la Mora, Maurice. *Metodología de la investigación: desarrollo de la inteligencia*, 5 edición (en español), Cengage Learning Editores, pp. 97. ISBN 9706863842. 2006.

36. PMI. *Project Management Body of Knowledge.* 2004.

37. Calisoft. *Métricas para Costo-Proyecto.* Universidad de las Ciencias Informáticas. Ciudad de la Habana. : s.n., 2009.

38. Palmkvist, Karin. *Use Cases Patterns and Blueprints.* Addison-Wesley. 2004. http://eva.uci.cu/mod/resource/view.php?id=22430

39. Perez, Dr. Pedro Yobanis Piñero. *Manual de configuración de paquete herramientas para la gestión de proyectos v0.1. C*iudad de la Habana. Universidad de las Ciencias Informáticas. : s.n., 2010.

40. Fowler, Martín. *Patterns of Enterprise Application Architecture.* Boston. Estados Unidos : Addison-Wesley : s.n., 2003.

41. Harrison, Daniel Rodríguez y Rachel. *Medición en la Orientación a Objetos.* School of Computer Science, Cybernetics & Electronic Engineering : s.n., 1999. http://www.cc.uah.es/drg/b/RodHarRama00.pdf

BIBLIOGRAFÍA CONSULTADA

1. —. *Una guía del cuerpo de conocimientos de la administración de proyectos (PMI).* 1996.

2. Zulueta, Y. y Despaigne, E. *MoGeRi: Un modelo para la gestión de riesgos en proyectos de software.* 2008.

3. Zulueta, Y. y Hernández, A. *Retos en la gestión de los riesgos en proyectos de software.* Universidad de las Ciencias Informáticas : s.n., 2008.

4. Ayaach, Farid. *Métricas de Calidad de Software.* http://www.ldc.usb.ve/~abianc/materias/ci4712/metricas.pdf

ANEXOS

Anexo 2. Descripción del CU gestionar riesgos (Elaboración propia).

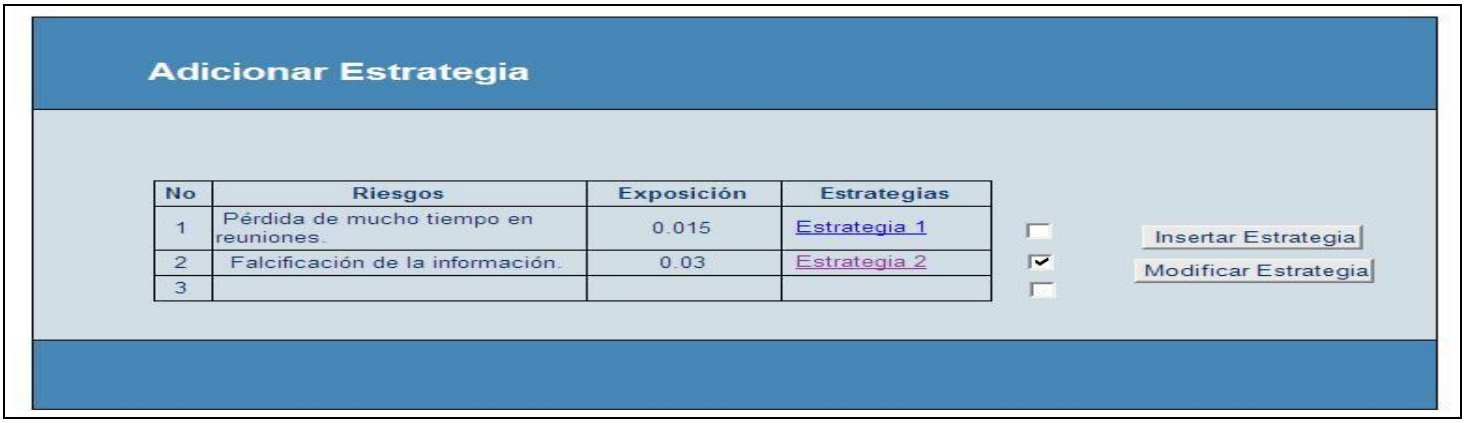
Nombre del CU	Gestionar Riesgos
Actores	Identificador
Objetivo	Gestionar los datos de los riesgos del proyecto (nombre y registro del riesgo).
Resumen	El Identificador tiene la posibilidad de insertar o modificar cada uno de los riesgos del proyecto del sistema.
Referencia	RF11
Precondiciones	El identificador debe estar previamente autenticado.
Pos condiciones	El identificador debe haber insertado o modificado al menos un riesgo.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El identificador selecciona la opción “Gestionar riesgos” del menú.	2. El sistema muestra una interfaz con la lista de los riesgos y diferentes opciones que le permiten al identificador:
2.1 Insertar, ver sección “Insertar nuevo riesgo”.	
2.2. Modificar, ver sección “Modificar riesgo”.	
Sección Insertar nuevo riesgo	
3. El identificador toma la opción de insertar un nuevo riesgo.	4. El sistema muestra la interfaz para agregar el riesgo.
5. El identificador introduce los datos del nuevo riesgo según el registro de riesgos (ver figura 3) y selecciona la opción “Aceptar”.	6. El sistema valida los datos de entrada.
	7. El sistema almacena la información en su base de datos.
	8. Muestra el listado con el nuevo riesgo y da la oportunidad de generar un documento con éste.
	9. Retornar al paso 2, culminando así el caso de uso.
Flujo alterno	
	6.1 El sistema solicita los datos del nuevo riesgo ya que hubo un error a la hora de entrar los datos.
	6.2 El sistema retoma el paso 2.
Sección Modificar riesgo	
3. El identificador selecciona la opción de “Modificar riesgo” y elige el riesgo a modificar.	4. El sistema muestra la interfaz para modificar los datos del riesgo.
	5. El sistema modifica los datos seleccionados por el identificador.

	6. El sistema muestra el nuevo listado con los datos del riesgo ya modificados y da la opción de generar un documento con éste.
	7. Retomar paso 2.
Flujo alterno	
3.1 El identificador selecciona la opción de cancelar.	3.1.2 El sistema cancela la operación de modificar riesgo.
	3.1.3 Retomar paso2.
Prioridad	Crítico

Anexo 4. Descripción del CU gestionar estrategia (Elaboración propia)**.**

Nombre del CU	Gestionar estrategia
Actores	Planificador de respuesta
Objetivo	Gestionar los datos de las estrategias de los riesgos del proyecto.
Resumen	El Planificador de respuesta tiene la posibilidad de insertar o modificar cada una de las estrategias de los riesgos del proyecto del sistema.
Referencia	RF15
Precondiciones	El Planificador de respuesta debe estar previamente autenticado. El listado de riesgos priorizados debe tener al menos un riesgo insertado.
Pos condiciones	El Planificador de respuesta debe haber insertado o modificado una o más estrategias.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El Planificador de respuesta selecciona la opción “Gestionar Estrategia” del menú.	2. El sistema muestra una interfaz con la lista de los riesgos priorizados.

3. El Planificador de respuesta selecciona la opción “Definir Estrategia” para el riesgo deseado.	4. El sistema muestra una interfaz con la lista de estrategias del riesgo y las diferentes opciones que le permiten al planificador de respuesta:
4.1 Insertar, ver sección “Insertar Estrategia”.	
4.2. Modificar, ver sección “Modificar Estrategia”.	
Sección Insertar estrategia	
5. El planificador de respuesta toma la opción de insertar estrategia.	6. El sistema muestra la interfaz para agregar la estrategia.
7. El planificador de respuesta introduce los datos de la nueva estrategia y selecciona la opción “Aceptar”.	8. El sistema valida los datos de entrada.
	9. El sistema almacena la información en su base de datos.
	10. Muestra el listado con la nueva estrategia.
	11. Retornar al paso 4, culminando así el caso de uso.
Flujo alterno	
	8.1 El sistema solicita los datos de la nueva estrategia ya que hubo un error a la hora de entrar los datos.
	8.2 El sistema retoma el paso 4.
Sección Modificar estrategia	
5. El planificador de respuesta selecciona la opción de “Modificar Estrategia”.	4. El sistema muestra la interfaz para modificar la estrategia.
	5. El sistema modifica los datos seleccionados por el planificador.
	6. El sistema muestra el nuevo listado con los datos de la estrategia ya modificados.
	7. Retomar paso 4.
Flujo alterno	
5.1 El planificador de respuesta selecciona la opción de cancelar.	5.2 El sistema cancela la operación de modificar la estrategia.
	5.3 Retomar paso 4.
Prioridad	Crítico



Anexo 5. Descripción del CU gestionar respuesta (Elaboración propia)**.**

Nombre del CU	Gestionar Respuesta
Actores	Planificador de respuesta
Objetivo	Gestionar los datos de las respuestas a las estrategias de los riesgos del proyecto.
Resumen	El Planificador de respuesta tiene la posibilidad de insertar o modificar cada una de las respuestas a las estrategias de los riesgos del proyecto del sistema.
Referencia	RF16
Precondiciones	El Planificador de respuesta debe estar previamente autenticado. El listado de riesgos priorizados debe tener al menos un riesgo insertado. Debe haber al menos una estrategia en el listado.
Pos condiciones	El Planificador de respuesta debe haber insertado o modificado una o más respuestas.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El Planificador de respuesta selecciona la opción “Gestionar Respuesta” del menú.	2. El sistema muestra una interfaz con la lista de los riesgos priorizados.
3. El Planificador de respuesta selecciona la opción “Definir Respuesta” para el riesgo deseado.	4. El sistema muestra una interfaz con la lista de las estrategias del riesgo.
5. El Planificador de respuesta selecciona la opción “Definir Respuesta” para la estrategia deseada.	6. El sistema muestra una interfaz con la lista de las respuestas de la estrategia del riesgo y las diferentes opciones que le permiten al planificador de respuesta:
4.1 Insertar, ver sección “Insertar Respuesta”.	
4.2. Modificar, ver sección “Modificar Respuesta”.	

Sección Insertar respuesta	
5. El planificador de respuesta toma la opción de insertar respuesta.	6. El sistema muestra la interfaz para agregar la respuesta.
7. El planificador de respuesta introduce los datos de la nueva respuesta y selecciona la opción “Aceptar”.	8. El sistema valida los datos de entrada.
	9. El sistema almacena la información en su base de datos.
	10. Muestra el listado con la nueva respuesta.
	11. Retornar al paso 4, terminando así el caso de uso.
Flujo alterno	
	8.1 El sistema solicita los datos de la nueva respuesta ya que hubo un error a la hora de entrar los datos.
	8.2 El sistema retoma el paso 4.
Sección Modificar respuesta	
. El planificador de respuesta selecciona la opción de “Modificar Respuesta”.	6. El sistema muestra la interfaz para modificar la respuesta.
	7. El sistema modifica los datos seleccionados por el planificador de respuesta.
	8. El sistema muestra el nuevo listado con los datos de la respuesta ya modificados.
	9. Retomar paso 4.
Flujo alterno	
5.1 El planificador de respuesta selecciona la opción de cancelar.	5.2 El sistema cancela la operación de modificar la respuesta.
	5.3 Retomar paso 4.
Prioridad	Crítico

Anexo 9. Descripción del CU seleccionar herramienta o técnica (Elaboración propia)**.**

Nombre del CU	Seleccionar herramientas y técnicas

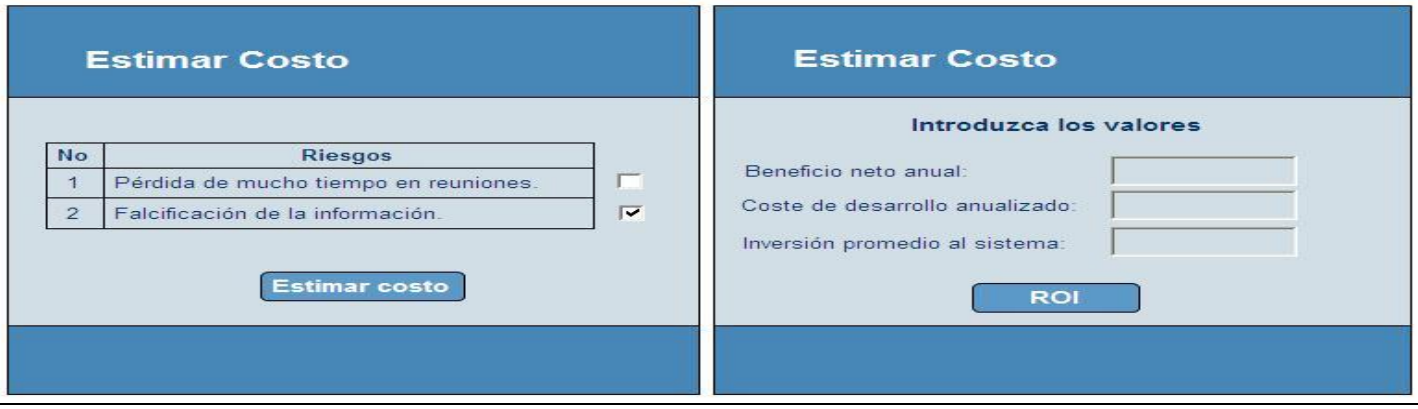
Actores	Identificador
Objetivo	Seleccionar herramientas y técnicas para identificar los riesgos.
Resumen	El identificador tiene la posibilidad de insertar, modificar y seleccionar herramientas y técnicas correspondientes a la identificación de los riesgos.
Referencia	RF10
Precondiciones	El identificador debe estar previamente autenticado.
Pos condiciones	El identificador de GR debe haber insertado, modificado o seleccionado al menos una actividad.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El identificador selecciona la opción "Gestionar herramienta o técnica **"** del menú.	2. El sistema muestra una interfaz con la lista de las herramientas o técnicas y diferentes opciones que le permiten al identificador de riesgos:
2.1 Insertar, ver sección "Insertar nueva herramienta o técnica "	
2.2 Modificar, ver sección "Modificar herramienta o técnica "	
2.3 Seleccionar, ver sección "Seleccionar herramienta o técnica "	
Sección Insertar nueva herramienta o técnica	
3. El identificador toma la opción de insertar una nueva herramienta o técnica.	4. El sistema muestra la interfaz para agregar la herramienta o técnica.
5. El identificador introduce una nueva herramienta o técnica y selecciona la opción “Aceptar”.	6. El sistema valida los datos de entrada.
	7. El sistema almacena la información en su base de datos.
	8. Muestra la nueva herramienta o técnica y da la opción de generar un documento con esos datos.
	9. Retornar al paso 2, culminando así el caso de uso.
Flujo alterno	
	6.1 El sistema solicita los datos de la nueva herramienta o técnica ya que hubo un error a la hora de entrar los datos.
	6.2 El sistema retoma el paso 2.
Sección Modificar herramienta o técnica	
3. El identificador selecciona la opción de “Modificar herramienta o técnica”.	4. El sistema muestra la interfaz para modificar la herramienta o técnica.
	5. El sistema modifica los datos seleccionados por el identificador.
	6. El sistema muestra la lista de las herramientas o técnicas ya modificada y da la opción de generar un documento con esos datos.

	7. Retomar paso 2.
Sección Seleccionar herramienta o técnica	
3. El identificador elige la opción “Seleccionar herramienta o técnica”.	4. El sistema selecciona la herramienta o técnica deseada.
	5. El sistema muestra la lista de las herramientas o técnicas con la seleccionada y da la opción de generar un documento con esos datos.
	6. Retomar el paso 2.
Flujo alterno	
3.1 El identificador selecciona la opción de cancelar.	3.1.2 El sistema cancela la operación seleccionar herramientas o técnicas.
	3.1.3 Retomar paso2.
Prioridad	Crítico

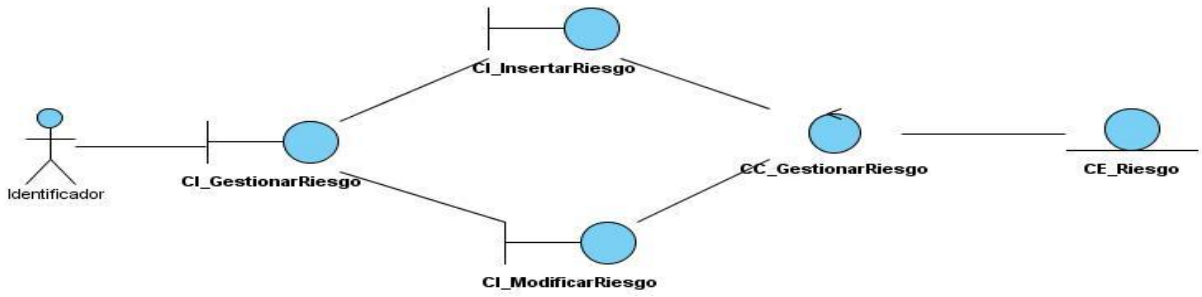
Anexo 13. Descripción del CU estimar costo (Elaboración propia)**.**

Nombre del CU	Estimar costo
Actores	Equipo de GR
Objetivo	Estimar el costo que puede traer consigo cada riesgo.
Resumen	El caso de uso se inicia cuando el equipo de GR solicita estimar el costo de un riesgo, el sistema muestra un listado con los riesgos que no se les ha realizado dicha estimación, el equipo de GR selecciona uno e introduce los valores de beneficio neto anual, coste de desarrollo anualizado e inversión promedio al sistema.
Referencia	RF8
Precondiciones	El equipo de GR debe estar previamente autenticado. El listado de riesgos no atendidos debe tener al menos un riesgo.

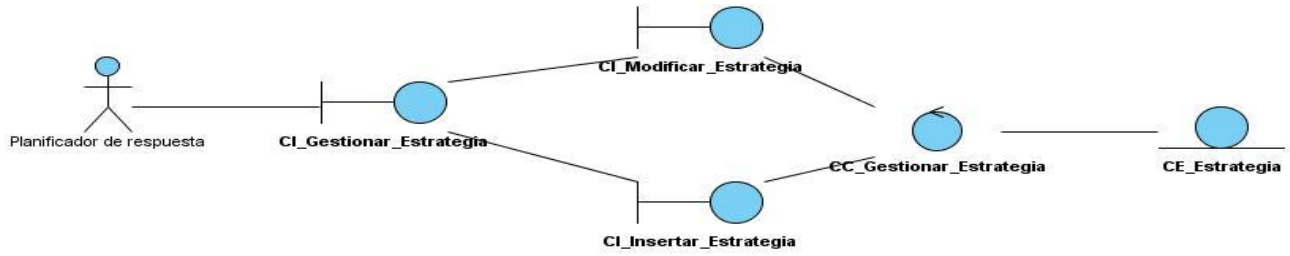
Pos condiciones	Queda almacenado el beneficio neto anual, el coste de desarrollo anualizado y la inversión promedio de al menos un riesgo, teniendo como resultado Retorno de la Inversión (ROI), que no es más que el costo.
Flujo de eventos	
Acciones del actor	**Respuesta del sistema**
1. El equipo de GR selecciona la opción “Estimar costo” del menú.	2. El sistema muestra el listado de los riesgos a los que no se les ha hecho la operación.
3. El equipo de GR selecciona el riesgo al que va a insertarle los valores.	4. El sistema muestra la interfaz para insertar los valores al riesgo.
5. El equipo de GR introduce los valores solicitados y selecciona la opción “Aceptar”.	6. El sistema valida los datos y almacena la información en su base de datos.
	7. El sistema muestra la interfaz con la estimación del costo del riesgo.
	8. Retomar acción 2.
Flujo alterno	
5.1 El equipo de GR desea cancelar el llenado de los valores, para esto oprime el botón “Cancelar”.	5.2 El sistema cancela la operación de de estimar el costo.
Prioridad	Crítico



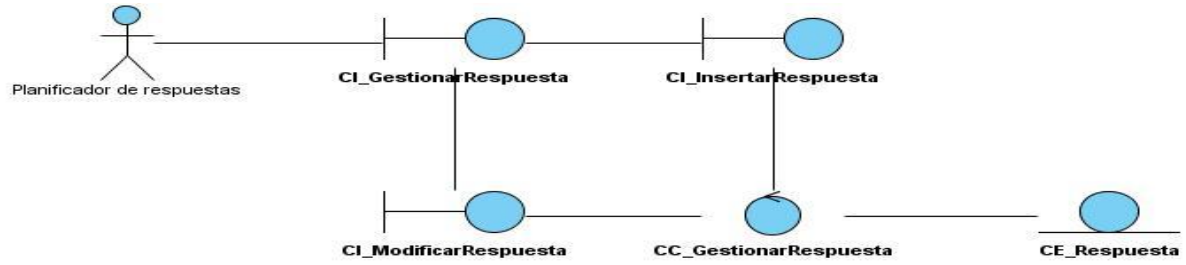
Anexo 17. Diagrama de clases del análisis. Gestionar riesgos (Elaboración propia)**.**



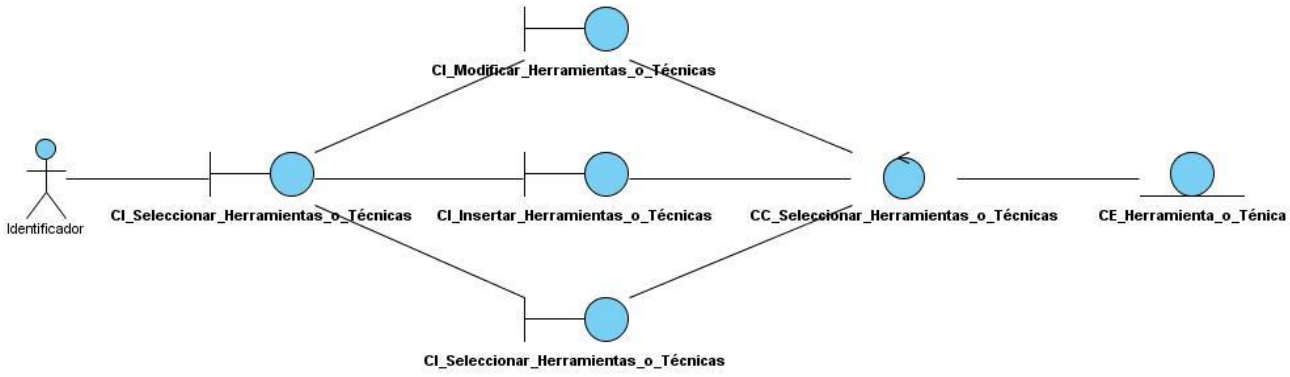
Anexo 19. Diagrama de clases del análisis. Gestionar estrategia (Elaboración propia)**.**



Anexo 20. Diagrama de clases del análisis. Gestionar respuesta (Elaboración propia)**.**

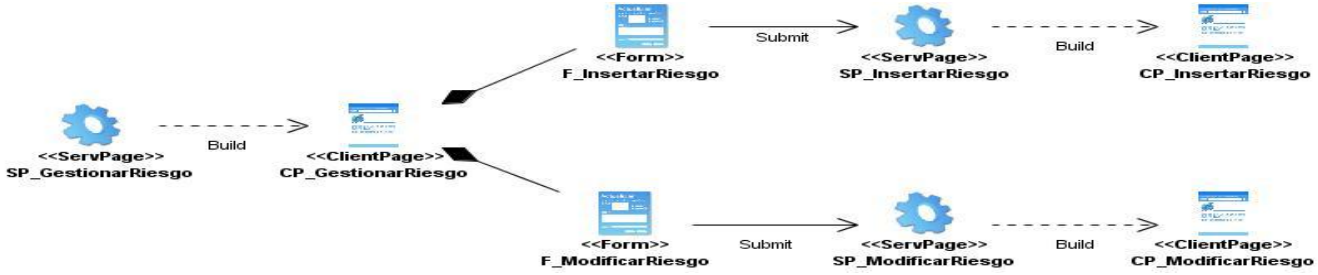


Anexo 24. Diagrama de clases del análisis. Seleccionar herramientas o técnicas (Elaboración propia)**.**

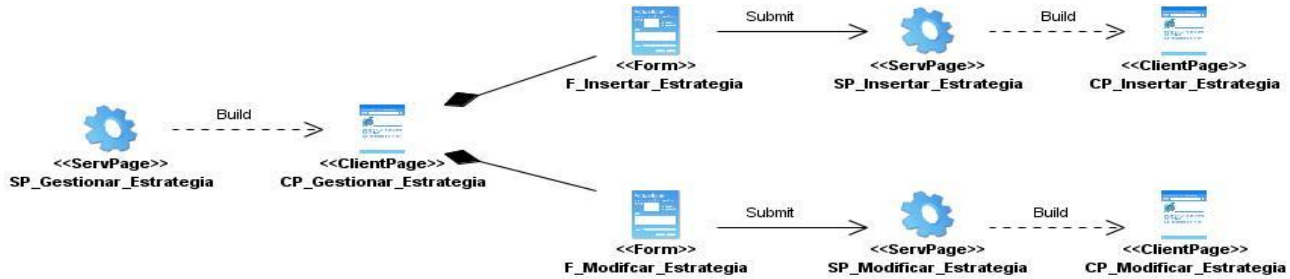


Anexo 28. Diagrama de clases del análisis. Estimar Costo (Elaboración propia)**.**

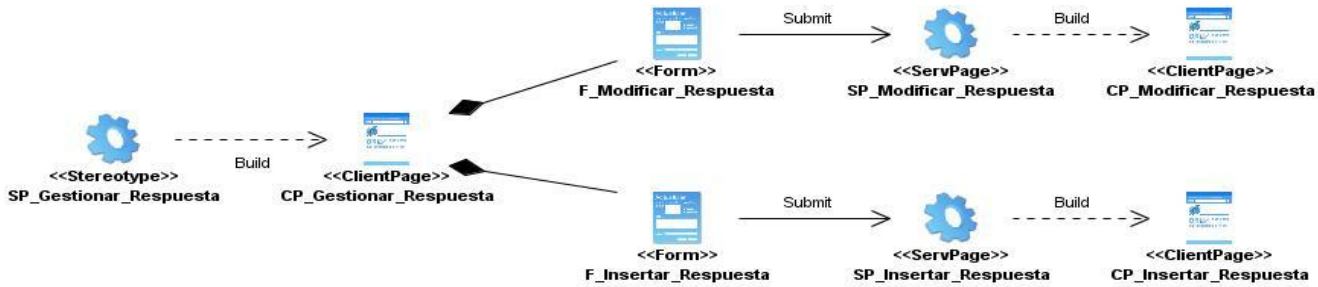
Anexo 31. Diagrama de clases del diseño. Gestionar Riesgos (Elaboración propia)**.**



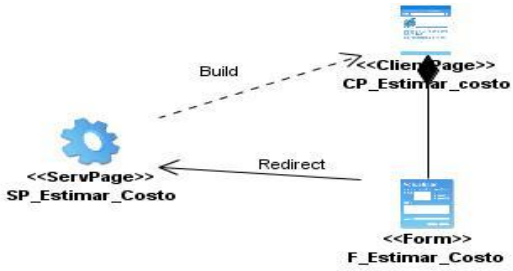
Anexo 33. Diagrama de clases del diseño. Gestionar estrategia (Elaboración propia)**.**



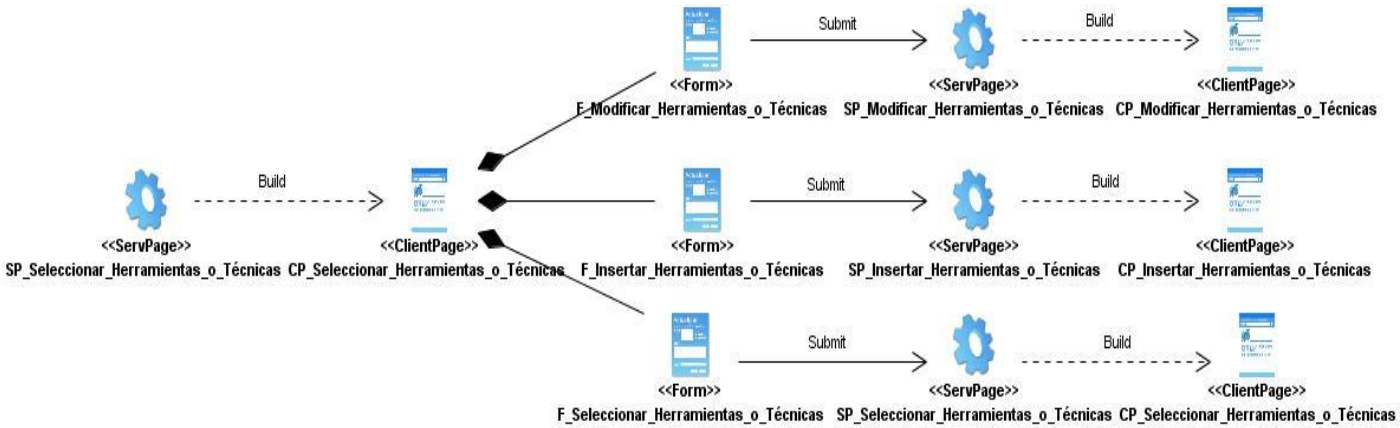
Anexo 34. Diagrama de clases del diseño. Gestionar respuesta (Elaboración propia)**.**



Anexo 37. Diagrama de clases del diseño. Estimar costo (Elaboración propia)**.**



Anexo 41. Diagrama de clases del diseño. Seleccionar herramientas o técnicas (Elaboración propia)**.**



Anexo 43. Guía para entrevista.

Introducción

- Saludo.
- Informar brevemente el objetivo: Identificar el grado de conocimiento de os involucrados acerca de la GR, su utilización en la UCI y su aplicación en los proyectos productivos.
- Reconocer la importancia, respeto y confidencialidad de los criterios del entrevistado.

Desarrollo

1. ¿Cuáles han sido sus experiencias en el trabajo relacionado con el desarrollo de software?

2. ¿En los proyectos en los que ha trabajado o trabaja, se ha tenido conocimiento sobre los riesgos que puedan afectar su buen término?
3. ¿Cuándo el proyecto ha enfrentado algún problema, se han documentado las experiencias sobre su solución, para que puedan ser utilizadas posteriormente?
4. ¿Conoce el concepto del término *Riesgo*?
5. ¿Conoce el concepto del término *Gestión de Riesgos*?
6. ¿Qué entiende por riesgo para un proyecto de desarrollo de software?
7. ¿Se realizan acciones de tratamiento o GR en la UCI? ¿Cuáles?
8. ¿Qué importancia le concede a la GR para el cumplimiento de los objetivos del proyecto?
9. ¿Considera relevante y necesario la adopción de un sistema para la GR en los proyectos productivos de la UCI?
10. ¿Qué características debe cumplir este sistema?

Conclusiones

- Agradecer su cooperación.
- Despedida.