UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

"Extensión al Redmine: Análisis cuantitativo de riesgos en proyectos de software"

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

Autor:

Darian Reyes Fernández de Bulnes

Tutora:

Ing. Yanet Nieto Doce

Asesores:

Ing. Yancy Martínez Pérez

Ing. Jorge Martínez Padrón

Ciudad de La Habana

Abril 2011

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y autorizo 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 _____.

Darian Reyes Fernández de Bulnes	Ing. Yanet Nieto Doce
_____	_____
Firma del Autor	Firma de la Tutora

DATOS DE CONTACTO

Tutora:

Nombres y Apellidos: Yanet Nieto Doce.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: ynieto@uci.cu

Ingeniera en Ciencias Informáticas, graduada en la UCI en el año 2009. Con 2 años de experiencia en su desempeño laboral, se ha desarrollado como profesora en adiestramiento en el Centro de Informática Industrial (CEDIN) perteneciente a la Facultad 5. Líder del proyecto SCADA Meteorología.

Asesores:

Nombres y Apellidos: Yancy Martínez Pérez.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: ymartinezp@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la UCI en el año 2006. Profesor con categoría de Instructor, con 5 años de experiencia en su desempeño laboral. Ha desempeñado varios roles en la producción como jefe de grupo y líder de proyecto. Actualmente es Jefe de Grupo de Gestión de Proyectos del departamento de Dirección de Proyectos del CEDIN.

Nombres y Apellidos: Jorge Martínez Padrón.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: jmpadron@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la UCI en el año 2010. Con 1 año de experiencia en su desempeño laboral. *Ha desempeñado varios roles en la producción como jefe de grupo y líder de proyecto. Actualmente es Jefe de Grupo de Gestión de Proyectos del departamento de Dirección de Proyectos del CEDIN.*

***

AGRADECIMIENTOS

Agradezco a todos los que ayudaron de una forma u otra a realizar esta tesis de grado.

DEDICATORIA

A mi familia.

RESUMEN

Un riesgo, es un evento incierto que si se produce, tiene un efecto positivo o negativo sobre al menos un objetivo de un proyecto; como pueden ser tiempo, coste, alcance o calidad. La gestión de riesgos es una de las disciplinas más significativa en los sistemas de gestión -mejora la toma de decisiones, minimiza las pérdidas y los impactos operativos- pues permite seguir una serie de pasos que posibilitan un resultado factible para un determinado proyecto, usuario o entidad. Gestionar los riesgos implica: identificarlos, analizarlos, controlarlos y eliminar las fuentes antes de que empiecen a afectar el cumplimiento de los objetivos del proyecto.

En los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI), la gestión de riesgos no se realiza con todo el rigor que requiere, ya que no se ha incorporado ninguna herramienta factible, que brinde a los líderes de proyecto un arma a la hora de combatir las incertidumbres que pueda presentar en el futuro cierto proyecto. Por tal motivo se hace necesaria la implementación de una herramienta que ofrezca solucionar estas inconveniencias.

Esta investigación tiene como principal objetivo, desarrollar una aplicación adjunta al gestor de proyectos Redmine, capaz de automatizar eficientemente el análisis cuantitativo de riesgos, proceso del manejo general de los mismos.

Siguiendo esta meta se obtuvo como resultado una extensión a la aplicación web Redmine, que integra el proceso anteriormente mencionado y que cumple con los objetivos trazados al inicio de su confección.

Palabras claves: análisis, cuantitativo, extensión, Redmine, riesgo.

ÍNDICE

INTRODUCCIÓN

La producción de software es una de las esferas científico-técnicas más prometedoras en la actualidad. Diariamente se construye un mundo cada vez más informatizado, donde los software deben contar con altos estándares de calidad y ser desarrollados en corto tiempo.

Los proyectos de la UCI se encuentran inmersos en un sinnúmero de situaciones, muchas de estas conocidas en términos informáticos como “riesgos”; los cuales atentan contra el correcto avance de los mismos. Aunque los riesgos deben ser analizados y gestionados por el equipo de desarrollo y muchos pueden ser comunes a otros detectados anteriormente, se debe realizar un análisis de las esencias y particularidades de cada proyecto para prever las amenazas y vulnerabilidades que se pudieran presentar en el ciclo de vida del software.

La administración de riesgos -una de las áreas de conocimiento de la gestión de proyectos- generalmente no se realiza con todo el rigor que requiere; lo que implica que los proyectos presenten numerosos problemas durante su desarrollo, al no tenerse en cuenta la mayor cantidad de riesgos predecibles de acuerdo con las características del software a producir.

Para dar respuesta a esto dentro de la UCI, en el curso 2008/2009, fue desarrollada una extensión para la gestión de riesgos adjunta a la herramienta de gestión de proyectos TRAC. Ésta permitía gestionar los riesgos durante el ciclo de vida de un proyecto, sin embargo, no analizaba numéricamente el efecto de los riesgos de mayor prioridad, ni contenía un método que aportara información de alta valía para la toma de decisiones en casos de incertidumbre.

Luego, en el curso 2009/2010, se implementó otra extensión igualmente adjunta a la herramienta TRAC, la cual hacía un análisis cuantitativo de riesgos empleando métodos especializados, con el objetivo de dar continuidad a la herramienta anteriormente mencionada. A pesar de esto, la herramienta solamente cubría la cuarta etapa del proceso de gestión de riesgos y no presentaba las características para adjuntarse a la aplicación de gestión de proyectos actualmente utilizada.

Con el paso del tiempo la UCI ha adquirido gran prestigio entre sus clientes por la calidad y eficiencia de los productos desarrollados, consecuentemente, se ha provocado una fuerte demanda de software por parte de instituciones y empresas tanto nacionales como extranjeras. Debido a esto, se ha tenido la necesidad de crear diferentes centros productivos en todas las facultades, cada uno especializado en la producción de software de un ámbito determinado.

En todos los centros productivos fue imprescindible mantener funcionando una aplicación para la gestión de proyectos dada la complejidad de los mismos; y fue decisivo implantar para toda la comunidad universitaria la aplicación Redmine, herramienta de código libre con interfaz web para la gestión de todos los proyectos productivos. A pesar de que la misma se encuentra en una etapa muy temprana de su desarrollo, ha demostrado ofrecer eficiencia y calidad.

Por tanto, la **situación problémica** es la siguiente: Las inconveniencias presentadas en todos los trabajos sobre el análisis cuantitativo de riesgos en proyectos de producción de software dentro de la UCI, pueden traer consigo que no se determine la mejor decisión en la dirección de proyectos cuando algunas condiciones o resultados son inciertos, debido a no contarse con la posibilidad de conocer el nivel de impacto de los riesgos inherentes al proyecto, lo cual conlleva a afirmar que la probabilidad de lograr los objetivos específicos del mismo resulte una incógnita. Además, es prácticamente imprescindible una herramienta que no se conforme con identificar los riesgos, sino que además los analice cuantitativamente, a través de la aplicación de gestión de proyectos -Redmine- generalizada para todos los proyectos de la Universidad.

Partiendo de la situación problémica se formula el siguiente **problema científico**: ¿Cómo analizar cuantitativamente los riesgos en un proyecto productivo de la UCI durante el ciclo de vida del software?

Se plantea como **objeto de estudio** el análisis cuantitativo de riesgos en proyectos de producción de software y como **campo de acción** las técnicas para el análisis cuantitativo de riesgos en proyectos de producción de software.

Luego, el **objetivo general** de la presente investigación es: Desarrollar una aplicación adjunta al Redmine que permita el análisis cuantitativo de riesgos durante el ciclo de vida del software.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas investigativas**:

1. Definición del proceso de análisis cuantitativo de riesgos, así como su flujo de trabajo, entradas, salidas y las técnicas aplicadas en el mismo.
2. Selección de las herramientas existentes para la gestión de riesgos con el fin de garantizar su reutilización.
3. Selección de las herramientas y técnicas para el desarrollo de la aplicación.
4. Levantamiento de los requisitos del sistema para cumplir con las condiciones que debe satisfacer la aplicación.

5. Modelado de los casos de usos para guiar el desarrollo del sistema.
6. Definición de la arquitectura del sistema y de los patrones que serán aplicados para evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
7. Modelado del sistema en términos de componentes para mostrar las dependencias lógicas entre los mismos.
8. Aplicación de las teorías de probabilidad y estadística sobre al análisis cuantitativo de riesgos en el desarrollo de la herramienta.
9. Realización de pruebas para detectar los errores del sistema desarrollado.

De acuerdo con todo lo anterior la **idea a defender** es la siguiente: El desarrollo de una herramienta que permita analizar cuantitativamente el efecto de los riesgos mejorará significativamente el proceso de toma de decisiones en los proyectos productivos.

En el cumplimiento de las tareas investigativas se utilizan varios **métodos y técnicas** para la búsqueda y procesamiento de la información, como son:

- A nivel teórico:
  - Analítico-sintético: Para la identificación de conceptos empleados dentro de la gestión de riesgos, analizando documentos para la extracción de los elementos más importantes sobre el tema en cuestión.
  - Análisis histórico-lógico: Para conocer con mayor profundidad los antecedentes y las tendencias actuales referidas a la gestión de riesgos, conociendo así la trayectoria histórica que tiene a través de su origen y las herramientas para el desarrollo de la misma, enfocándose en el análisis cuantitativo de riesgos.
  - Modelación: Se utilizará para representar las funcionalidades que tendrá la extensión, cumpliendo con un determinado nivel de similitud estructural y funcional con la realidad de la herramienta.
- A nivel empírico:
  - Entrevista: Se realizarán conversaciones planificadas con algunos expertos sobre el tema de gestión de riesgos y con los jefes de algunos proyectos dentro de la universidad, para obtener información sobre las dificultades que presentan las herramientas de dicha rama actualmente en uso dentro de la UCI.

- Observación: Se realizará una observación participativa y no participativa a la gestión de riesgos desarrollada en los proyectos productivos de la UCI, para lograr una percepción planificada dirigida a las dificultades presentes en la misma.

Estructura capitular:

Capítulo 1: Fundamentación teórica:

Este primer capítulo tiene como objetivo hacer un estudio valorativo de los fundamentos teóricos generales que sirven como punto de partida a la solución del problema. También se hace alusión a diferentes herramientas que existen en el mundo y en la UCI relacionadas con la gestión de riesgos. Además de las técnicas, tecnologías, herramientas y metodologías en los que se apoya el trabajo para dar solución al problema planteado, como lo es la Simulación de Montecarlo para el análisis cuantitativo de riesgos.

Capítulo 2: Características del Sistema:

En este capítulo se analizará el alcance de la aplicación, lo que permitirá tener una visión de los objetivos que se pretenden alcanzar con su desarrollo. Se muestran además, los requisitos funcionales y no funcionales. De forma general se describen los actores del sistema, realizándose el modelo de casos de uso con sus correspondientes descripciones. Se realiza una comparación de las técnicas abordadas con anterioridad y se define la o las técnicas a implementar.

Capitulo 3: Arquitectura y Diseño del Sistema:

El contenido de este capítulo es definir la arquitectura y diseño del sistema. Se muestra la realización de los casos de uso, los diagramas de clases y de secuencia para cada uno de estos casos de uso, así como la descripción detallada de las clases. Finalmente, se definen las clases persistentes mediante el modelo físico de datos.

Capítulo 4: Implementación y Pruebas del Sistema:

En este último capítulo se implementa el sistema en términos de componentes y se desarrolla además el diagrama de despliegue. Se cumple con el propósito de diseñar y ejecutar los casos de prueba para los diferentes requisitos funcionales del sistema, asegurando la calidad del mismo siendo extensión del Redmine.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción del capítulo

El presente capítulo, ofrece información sobre el estado del proceso de gestión de riesgos en los proyectos de software, tanto a nivel mundial como nacional, profundizándose conceptual y técnicamente en el análisis cuantitativo de riesgos. En pos de su desarrollo se realizará un análisis acerca del objeto de estudio de la investigación y se explicará brevemente los pasos para la gestión de riesgos; así como las metodologías de desarrollo, tecnologías y herramientas utilizadas que darán solución al trabajo.

1.2 Riesgo

Para definir el término riesgo se hace referencia a las denominaciones que han realizado diferentes autores:

- Posibilidad de sufrir una pérdida. [1]
- Incertidumbre asociada a un evento futuro o a un evento supuesto. [2]
- Evento o condición incierta que, en caso de ocurrir, tiene un efecto positivo o negativo sobre los objetivos de un proyecto. [3]
- Contingencia o proximidad de un daño o peligro. [4]
- Posibilidad que un evento adverso, desgracia o contratiempo pueda manifestarse produciendo una pérdida. [5]

La presencia de un riesgo puede venir dado por cambios de opinión, de acciones o de lugares. Además, implica una elección y la incertidumbre que ésta entraña. [6] Un riesgo tiene una causa que, de manifestarse (evento de riesgo), entraña una consecuencia (efecto) que tiene su origen en la incertidumbre que está presente en los proyectos. [7]

1.3 Gestión de riesgos

Se puede definir la gestión de riesgos de un proyecto como el proceso sistemático-científico vinculado a la identificación, análisis y respuesta a los riesgos durante todo el ciclo de vida de dicho proyecto. [8] La metodología para gestión de riesgos MAGERIT 2.0 sugiere los siguientes procesos: identificación de riesgos, análisis de riesgos, planificación de respuestas a los riesgos, seguimiento de riesgos y control de riesgos. (Ver Ilustración 1)

Ilustración 1: Modelo de gestión de riesgos.

Los procesos de gestión de riesgos incluyen lo siguiente:

- Identificación de riesgos: Determinar qué riesgos pueden afectar al proyecto, documentar sus características y organizarlos en categorías.
- Análisis:
  - Análisis cualitativo de riesgos: Priorizar los riesgos para realizar otros análisis o acciones posteriores, evaluando y combinando su probabilidad de ocurrencia y su impacto.
  - Análisis cuantitativo de riesgos: Analizar numéricamente el efecto de los riesgos identificados y priorizados en los objetivos generales del proyecto.
- Planificación de la respuesta a los riesgos: Desarrollar opciones y acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto.
- Seguimiento de riesgos: Realizar el seguimiento de los riesgos identificados y analizados. Se chequean los indicadores de probabilidad e impacto haciendo una comparación para darles tratamiento.
- Control de riesgos: Supervisar los riesgos residuales, identificar nuevos riesgos, ejecutar planes de respuesta a los riesgos y evaluar su efectividad a lo largo del ciclo de vida del proyecto. Se verifican los planes de mitigación y contingencia.

La presente investigación estará centrada en el proceso de análisis cuantitativo de riesgos, ya que el proceso de identificación de riesgos ya está en desarrollo y el de análisis de cualitativo de riesgos está

automatizado en el Redmine. Además éste proceso es uno de los más importantes y complejos; y con solo los procesos anteriores, la gestión de riesgos estaría significativamente incompleta.

1.3.1 Análisis cuantitativo de riesgos

El análisis cuantitativo de riesgos se realiza en función de los riesgos priorizados en el proceso de análisis cualitativo de riesgos, por tener un posible impacto significativo sobre las demandas recurrentes del proyecto. El proceso presenta un método cuantitativo para tomar decisiones en caso de incertidumbre y utiliza técnicas tales como la simulación de Montecarlo o el análisis mediante árbol de decisiones; con el objetivo de:

- Cuantificar los posibles resultados del proyecto y sus probabilidades.
- Evaluar la probabilidad de lograr los objetivos específicos del proyecto.
- Identificar los riesgos que requieren una mayor atención mediante la cuantificación de su contribución relativa al riesgo general del proyecto.
- Determinar la mejor decisión a tomar por la dirección de proyectos cuando algunas condiciones o resultados son inciertos.

El análisis cuantitativo de riesgos, debe repetirse como parte del seguimiento y control, para determinar si el riesgo general del proyecto ha sido reducido satisfactoriamente. Los resultados pueden indicar la necesidad de más o menos acciones de gestión de riesgos, y son la entrada al proceso de planificación de la respuesta a los riesgos. [9] Por lo tanto, este proceso queda dividido en tres etapas fundamentales: entradas, técnicas y herramientas y salida. (Ver Ilustración 2)

Ilustración 2: Proceso de análisis cuantitativo de riesgos.

1.3.1.1 Entradas

- Activos de los procesos de la organización: Información de proyectos anteriores similares ya completados, estudios de proyectos similares por especialistas en riesgos y bases de datos de riesgos que pueden estar disponibles en fuentes de la industria o de propiedad exclusiva.
- Enunciado del alcance del proyecto: Describe, en detalle, los productos terminados del proyecto y el trabajo necesario para crear tales productos. Proporciona un entendimiento común del alcance del proyecto entre los interesados del mismo, describiendo sus principales objetivos. Permite al equipo del proyecto realizar una planificación más detallada, guía el trabajo del equipo del proyecto durante la ejecución y proporciona la línea base para evaluar si las solicitudes de cambio o trabajo adicional están comprendidas dentro o fuera de los límites del proyecto.
- Plan de gestión de riesgos: Algunos elementos claves del plan de gestión de riesgos para el análisis cuantitativo de riesgos incluyen los roles y responsabilidades para la gestión de riesgos, presupuestos y actividades de gestión de riesgos del cronograma, categorías de riesgo, estructura de desglose del riesgo y tolerancias al riesgo ya revisadas.
- Registro de riesgos: Algunos elementos clave del registro de riesgos para el análisis cuantitativo de riesgos incluyen la lista de riesgos identificados, la lista de prioridades -analizadas mediante la matriz de prioridad - (Ver Ilustración 3) o clasificaciones relativas a los riesgos del proyecto, junto a los riesgos agrupados por categorías.
- Plan de gestión de proyectos: El plan de gestión de proyectos incluye:
    - Plan de gestión del cronograma del proyecto: Establece el formato y los criterios para desarrollar y controlar el cronograma del proyecto.
    - Plan de gestión de costes del proyecto: Establece el formato y los criterios para planificar, estructurar, estimar, preparar el presupuesto y controlar los costes del proyecto. [9]

Probabilidad	Amenazas				
0.90	0.05	0.09	0.18	0.36	0.72
0.70	0.04	0.07	0.14	0.28	0.56
0.50	0.03	0.05	0.10	0.20	0.40
0.30	0.02	0.03	0.06	0.12	0.24
0.10	0.01	0.01	0.02	0.04	0.08
Impacto	0.05	0.10	0.20	0.40	0.80

Ilustración 3: Matriz de prioridad.

1.3.1.2 Técnicas y herramientas

- Técnicas de recopilación y representación de datos:
  - Juicio de expertos: Expertos en la materia, como pueden ser ingenieros o estadistas, internos o externos a la organización, validan los datos y las técnicas.
  - Entrevistas: Se usan para cuantificar la probabilidad y el impacto de los riesgos sobre los objetivos del proyecto. La información necesaria para su realización depende del tipo de distribuciones de probabilidad que se vayan a usar. Por ejemplo, para algunas distribuciones comúnmente usadas, la información se podría recopilar agrupándola en escenarios optimistas -bajo-, pesimistas -alto- y más probables; y en media y desviación estándar para las otras distribuciones. Documentar el fundamento de los rangos de riesgo es un componente importante de la entrevista de riesgos, ya que puede suministrar información sobre la fiabilidad y la credibilidad del análisis.
  - Distribuciones de probabilidad: Representan la incertidumbre de los valores obtenidos, tales como las duraciones de las actividades del cronograma y los costes de los componentes del proyecto. Las distribuciones discretas pueden usarse para representar eventos inciertos, como el resultado de una prueba o un posible escenario en un árbol de decisiones. Las distribuciones asimétricas representan formas que son compatibles con los datos generalmente desarrollados durante el análisis de los riesgos del proyecto; y las uniformes pueden usarse si no hay ningún valor obvio que sea más probable que cualquier otro entre límites altos y bajos especificados -como en la etapa inicial del concepto de diseño-. Sobre este acápite se profundizará más adelante teniendo en cuenta su peso dentro del trabajo.

- Técnicas de análisis cuantitativo de riesgos y de modelado:
  - Análisis de sensibilidad: Ayuda a determinar qué riesgos tienen el mayor impacto posible sobre el proyecto. Este método, examina en qué medida la incertidumbre de cada elemento del proyecto afecta al objetivo que está siendo examinado, cuando todos los demás elementos inciertos se mantienen con sus valores de línea base. Una representación típica del análisis de sensibilidad es el diagrama con forma de tornado, útil para comparar la importancia relativa de las variables que tienen un alto grado de incertidumbre con aquellas que son más estables.
  - Análisis del valor monetario esperado: Es un concepto estadístico que calcula el resultado promedio de los valores monetarios que pueden darse en un escenario futuro. Generalmente, el valor monetario esperado de las oportunidades se expresará con valores positivos, mientras que el de las amenazas será negativo. El valor monetario esperado se calcula multiplicando el valor de cada posible resultado por su probabilidad de ocurrencia, más la suma de sus resultados. Esto se emplea comúnmente en el análisis mediante árbol de decisiones.
  - Análisis mediante árbol de decisiones: Normalmente, se estructura usando un diagrama de árbol de decisiones que describe una situación que se está considerando, y las implicaciones de cada una de las opciones disponibles y sus posibles escenarios. Incorpora el coste de cada opción disponible, las probabilidades de cada escenario posible y las recompensas de cada camino lógico alternativo. Al resolver el árbol de decisiones se obtiene el valor monetario esperado u otra medida de interés para la organización correspondiente a cada alternativa, cuando todas las recompensas y las decisiones subsiguientes son cuantificadas. Es una forma de calcular los resultados financieros de una interdependencia o riesgo en un proyecto. Es una técnica para determinar los riesgos generales relacionados con una serie de riesgos.
  - Modelado y simulación: La modelación es uno de los enfoques cuantitativos más ampliamente usados para la toma de decisiones. Es un método para aprender sobre un sistema real experimentado con el modelo que lo representa. Un modelo de simulación contiene las expresiones matemáticas y las relaciones lógicas que describen como calcular el valor de las salidas dados los valores de las entradas. Cualquier modelo de simulación tiene dos entradas: entradas controlables y entradas probabilísticas.

    Al realizar un experimento de simulación, un analista de riesgos selecciona el valor o los valores para las entradas controlables y posteriormente se generan de forma aleatoria valores

para las entradas probabilísticas. Así, el modelo utiliza los valores de ambas entradas para calcular el valor o los valores de la salida.

Después de revisar los resultados de la simulación, el analista de riesgos es capaz de recomendar decisiones para las entradas controlables que proporcionarán la salida para el sistema real. La simulación se ha empleado con éxito en una variedad de aplicaciones, por ejemplo: se utiliza en el desarrollo de nuevos proyectos con el objetivo de determinar la probabilidad de que un producto nuevo sea rentable. Durante la simulación de un proyecto se usa un modelo que traduce las incertidumbres especificadas -a un nivel detallado del proyecto- en su impacto posible sobre los objetivos del proyecto. Normalmente, se realiza usando la simulación de Montecarlo.

1.3.1.3 Salida

- Registro de riesgos actualizado: El registro de riesgos es un componente del plan de gestión de proyectos. Se inicia en el proceso de identificación de riesgos y se actualiza en el análisis cuantitativo de riesgos. Las actualizaciones incluyen los siguientes componentes principales:
    - Análisis probabilístico del proyecto: Se realizan estimaciones de los posibles resultados del cronograma y los costes del proyecto, listando las fechas de conclusión y costes posibles con sus niveles de confianza asociados. Esta salida, normalmente expresada como una distribución acumulativa, se usa con las tolerancias al riesgo de los interesados para permitir la cuantificación de las reservas para contingencias de coste y tiempo. Dichas reservas para contingencias son necesarias para reducir el riesgo de desviación de los objetivos del proyecto establecidos a un nivel aceptable para la organización.
    - Probabilidad de lograr los objetivos de coste y tiempo: Con los riesgos que afronta el proyecto, la probabilidad de lograr sus objetivos bajo el plan en curso, puede estimarse usando los resultados del análisis cuantitativo de riesgos.
    - Lista priorizada de riesgos cuantificados: Esta lista de riesgos incluye aquellos riesgos que representan la mayor amenaza o presentan la mayor oportunidad para el proyecto. Se incluyen los riesgos que requieren la mayor contingencia de costes y aquellos que tienen más probabilidad de influir sobre el camino crítico.

- Tendencias en los resultados del análisis cuantitativo de riesgos: A medida que se repite el análisis, puede hacerse evidente una tendencia que lleve a conclusiones que afecten a las respuestas a los riesgos. [9]

1.4 Probabilidad y estadística en el análisis cuantitativo de riesgos

Probabilidad es la parte de la Matemática que maneja con números la incertidumbre. La incertidumbre se presenta en casi todas las situaciones que nos rodean. Los modelos probabilísticos o aleatorios describen la incertidumbre y permiten medirla.

1.4.1 Simulación de Montecarlo

La simulación de Montecarlo realiza el análisis de riesgos con la creación de modelos de posibles resultados mediante la sustitución de un rango de valores -una distribución de probabilidad- para cualquier factor con incertidumbre inherente. Luego, calcula los resultados una y otra vez, usando en cada caso un grupo diferente de valores aleatorios de las funciones de probabilidad, dependiendo del número de incertidumbres y de los rangos especificados. Puede ser necesario realizar miles o decenas de miles de cálculos. La simulación de Montecarlo produce distribuciones de valores de los resultados posibles, de esta forma, proporciona una visión mucho más completa de lo que puede suceder.

Un elemento importante en los procesos de simulación, es identificar las distribuciones de probabilidad apropiadas para los datos. Normalmente, se requiere analizar la información empírica o histórica y ajustarla a alguna distribución; en otros casos, dicha información no se encuentra disponible y quien construye el modelo de simulación debe utilizar su juicio personal para determinar qué distribución utilizar.

Mediante el uso de distribuciones de probabilidad, las variables pueden generar diferentes probabilidades de que se produzcan diferentes resultados. Las distribuciones de probabilidad son una forma mucho más realista de describir el comportamiento incierto de las variables en un análisis de riesgo. Existen un gran número de éstas, pero las que más se usan en campos relacionados con el desarrollo de software, como la ingeniería, investigación y desarrollo son:

- Binomial: Distribución que aparece de forma natural al realizar repeticiones independientes de un experimento que tenga respuesta binaria, generalmente clasificada como éxito o fracaso. Por ejemplo, esa respuesta puede ser si un determinado grupo de programadores trabajará con un determinado Entorno de Desarrollo Integrado (IDE) o no. Este modelo se aplica a poblaciones

finitas de las que se toman elementos al azar con reemplazo, y también a poblaciones conceptualmente infinitas, como por ejemplo las piezas que produce una máquina, siempre que el proceso de producción sea estable -la proporción de piezas defectuosas se mantiene constante a largo plazo- y sin memoria -el resultado de cada pieza no depende de las anteriores-.

- Normal: -Curva de campana o Campana de Gauss- Se define la media -valor esperado- y una desviación estándar para describir la variación con respecto a la media. Los valores intermedios cercanos a la media tienen mayor probabilidad de producirse. Es una distribución simétrica y describe muchos fenómenos naturales, como puede ser la estatura de una población.
- Uniforme: Se realiza cuando todos los valores tienen la misma probabilidad de producirse. Se define el mínimo y el máximo. Ejemplos de variables que se distribuyen de forma uniforme son los costos de manufacturación o los ingresos por las ventas futuras de un nuevo producto.
- Triangular: En esta distribución se definen los valores mínimo, más probable y máximo. Los valores situados alrededor del valor más probable tienen más probabilidades de producirse. Las variables que se pueden describir con una distribución triangular son el historial de ventas por unidad de tiempo y los niveles de inventario, por poner un ejemplo.

1.4.2 Estadística

Independientemente de las definiciones clásicas que se puedan encontrar en diccionarios o manuales, la Estadística es una ciencia que facilita la toma de decisiones mediante la representación ordenada de los datos observados en tablas y gráficos estadísticos. [10] Reduciendo los datos observados a un pequeño número de medidas estadísticas que permitirán la comparación entre diferentes series de datos, y estimando la probabilidad de éxito que tiene cada una de las decisiones posibles. Es una disciplina científica dedicada a la aplicación y desarrollo de la teoría y las técnicas apropiadas para la recolección, clasificación, presentación, análisis e interpretación de información cuantitativa obtenida por observación o experimentación. [11]

Existen varios factores de relevancia en esta ciencia:

- Tablas de distribución de frecuencias: Su objetivo es analizar los resultados mediante la creación de intervalos fijos (clases), mostrando la cantidad de resultados por grupo (frecuencia absoluta), y la frecuencia con la que estos resultados se obtuvieron (frecuencia relativa).
- Histograma: Se construye a partir de la tabla estadística de resultados, representando sobre cada intervalo, un rectángulo. El criterio para calcular la altura de cada rectángulo es el de mantener la

proporcionalidad entre las frecuencias absolutas o relativas de cada intervalo y el área de los mismos. [12]

- Estadígrafos: Son medidas descriptivas calculadas en una muestra -sus valores son variables, dependientes de la muestra-. [12] Como parte del conjunto de estadígrafos existentes se encuentran:
    - Media: Valor promedio de un conjunto de datos numéricos.
    - Mediana: Valor único que ocupa el centro de un conjunto de datos ordenados ascendente o descendentemente.
    - Moda: Dato que más se repite en un conjunto de datos.
    - Recorrido: Diferencia entre el valor más alto y el más bajo en un conjunto de datos.
    - Varianza: Media aritmética del cuadrado de las desviaciones de cada dato respecto a la media de esa muestra.
    - Desviación típica o estándar: Valor igual a la raíz cuadrada de la varianza.
    - Coeficiente de variación: Se define como el cociente entre la desviación típica y la media multiplicada por 100, para dar el resultado en porciento. [12]
    - Percentiles: Indica el porciento de valores que están por debajo de un valor determinado.

1.5 Herramientas existentes para la gestión de riesgos

En el mundo del software existen disímiles herramientas para gestionar de alguna manera los riesgos en los proyectos, a continuación se mencionan algunas de las más importantes y utilizadas:

- Chinchón: Es una herramienta para analizar cuantitativamente los riesgos de un sistema de información. Elaborada por el Dr. José Antonio Mañas, Profesor de la Escuela Técnica Superior de Ingenieros de Telecomunicaciones de la Universidad Politécnica de Madrid. La herramienta sigue el modelo Magerit 1.0 (metodología de análisis y gestión de riesgos de los sistemas informáticos), cuyo objetivo general es garantizar la seguridad en los sistemas de información, identificando problemas y definiendo protocolos que los eviten.
- Acertus: Es una herramienta propietaria de soluciones de software; permite a las organizaciones identificar, medir, gestionar y mitigar los riesgos. Acertus es una empresa, basada en la web, de solución de software para la gestión de riesgos diseñada para mejorar la forma en que las organizaciones evalúan los riesgos, la seguridad y el cumplimiento regulatorio.

- SCRAM99: Es un software propietario que permite evaluar el nivel del calendario y los riesgos de costo de un proyecto, al identificar las áreas del proyecto que se beneficiarían con una mayor atención de gestión. Define así las tendencias de riesgos con el propósito de prevenir los problemas antes de que se produzcan. Establece, además, el plan de contingencia y analiza sus posibles repercusiones.
- Active Risk Manager: Es un software propietario muy reconocido a nivel mundial -más de 25 000 usuarios-, cuya estrategia se basa en asegurar que la información de riesgos se recopile en todos los niveles jerárquicos de la organización para ser analizada y administrada correctamente.
- RiskTrak: Permite a las empresas evaluar mejor las incertidumbres en su organización. La herramienta de software RiskTrak ayuda en la identificación, definición, estimación y análisis de las incertidumbres.
- WelcomRisk: Es una herramienta que proporciona una estructura a la gestión proactiva de los medios, el seguimiento y la reducción de riesgos de proyectos y de la organización. Se asegura del éxito empresarial y del proyecto mediante la mejora de la toma de decisiones. Aumenta el control sobre el mismo y ayuda a la reducción de riesgos. WelcomRisk proporciona una solución completa para la gestión de riesgos. [13]

1.5.1 Herramientas existentes para la gestión de riesgos en la UCI

En la UCI se ha tratado a través de proyectos y de trabajos de diploma indagar en el tema de la gestión de riesgos abordando las diferentes fases de desarrollo, los métodos y herramientas a aplicar para obtener un resultado satisfactorio en la realización del mismo.

Se mencionan a continuación trabajos que se han realizado sobre dicho tópico:

- Análisis y gestión de riesgos: Sistema desarrollado en la Facultad 2, el cual apoya el proceso de gestión de la seguridad informática y posibilita la gestión eficiente de los riesgos sobre los activos y bienes informáticos de una organización en cada una de sus áreas de desarrollo o dominios, aplicaciones en desarrollo, entre otros. Además, brinda las indicaciones precisas u orientaciones objetivas de qué se debe hacer ante estos riesgos y cómo enfrentarlos exitosamente.
- Estrategia para la gestión de riesgos: Sistema desarrollado por la Facultad 8, utilizándose la herramienta TRAC para la publicación de la documentación generada. Este software promovió la

idea de crear un área para la gestión de riesgos dentro del TRAC, y las tareas a realizar en ella se describirían en el sistema de tickets que contiene el propio TRAC.

- Herramienta para la identificación de riesgos de proyectos productivos: Trabajo de diploma de la Facultad 5 del curso 2008/2009, centrado en la primera etapa de la gestión de riesgos -identificación de riesgos-, constituye solamente un solo proceso del ciclo de trabajo. Además es una extensión al TRAC, aplicación que actualmente no se utiliza en la UCI para realizar la gestión de los proyectos. Por todas estas razones no satisface las necesidades de la Universidad.
- Sistema para el análisis cuantitativo de riesgos de los proyectos productivos del Centro de Desarrollo de Informática Industrial (CEDIN): Trabajo de diploma de la Faculta 5 del curso 2009/2010, desarrollada para ser una aplicación web aislada, que no se integra a ningún gestor de proyectos. Solo logra un solo proceso de la gestión de riesgos -análisis cuantitativo de riesgos- y con poco desarrollo en el mismo. Queda demostrado que tampoco satisface las necesidades reales de la UCI.
- Análisis cualitativo de riesgos dentro del Redmine: Herramienta incluida dentro del gestor de proyectos Redmine que logra realizar el segundo proceso de la gestión de riesgos -análisis cualitativo de riesgos-. Este permite registrar los riesgos identificados previamente por el líder de proyecto, pero no está automatizado el proceso de identificación. Altamente útil y utilizado actualmente por toda la comunidad productiva de la UCI.

Las herramientas mencionadas anteriormente ofrecen en la UCI una gran ayuda para gestionar los riesgos de un proyecto; aunque, específicamente para la fase de análisis cuantitativo de riesgos resultan muy primitivas. Además ninguna de desarrollo para adjuntarla al Redmine - aplicación actual para la gestión de proyectos utilizada en la comunidad productiva de la UCI - salvo la de análisis cualitativo de riesgos dentro del Redmine.

1.6 Herramientas y tecnologías a utilizar para el desarrollo

La base para cualquier fase del proyecto dígase: diseño, maquetación, implementación, pruebas, entre otras; la constituye la selección de las herramientas y tecnologías a utilizar durante el desarrollo de una aplicación.

1.6.1 Sistema operativo: GNU/Linux

GNU/Linux es el término empleado para referirse al Sistema Operativo (SO) Unix que utiliza como base las herramientas de sistema de GNU y el núcleo Linux. Su desarrollo es uno de los ejemplos más prominentes de software libre ya que todo el código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la Licencia General Pública (GPL) de GNU y otras licencias libres. Hay que tener en cuenta que se logra la independencia tecnológica al posibilitar la libertad de uso y distribución de los programas, en aras de compartir el conocimiento sin tener problemas con la licencia u otros asuntos legales.

Se eligió para desarrollar la presente aplicación dado que el servidor de la UCI del gestor de proyectos Redmine -para el cual se desarrolló la aplicación- está administrado sobre este SO.

1.6.1.1 Distribución: Ubuntu 10.04

Ubuntu 10.04 es un SO libre que utiliza el núcleo Linux, pero la mayor parte de las herramientas básicas provienen del proyecto GNU. Se tuvo en cuenta que Ubuntu 10.04 es una distribución de desarrollo muy estable, por lo que los paquetes que se desarrollen en él y quieran ejecutarse utilizando cualquier distribución, siempre serán estables. A diferencia de otras distribuciones, las aplicaciones no requieren ser compiladas en la máquina que las esté usando.

Esta distribución es una de las más usadas en la UCI y no presenta problemas de compatibilidad con cualquier otra distribución sobre la cual este instalado el servidor del gestor de proyectos Redmine.

1.6.2 Metodología de desarrollo de software: Open UP 1.0

La metodología Open UP 1.0 -en inglés: Open Unified Process 1.0- es una parte del marco de desarrollo Eclipse. Esta metodología es un proceso unificado que aplica propuestas de gestión ágil, tratando de ser manejable en relación con el Proceso Unificado de Desarrollo de Software (RUP), ya que mantiene sus características esenciales: centrado en la arquitectura, dirigido por Casos de Uso (CU) y desarrollo iterativo e incremental dentro del ciclo de vida de un proyecto de software.

Es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por lo tanto, no provee lineamientos para todos los elementos que se manejan en un proyecto, pero tiene los componentes básicos que pueden servir de base a procesos específicos. La mayoría de los elementos de Open UP 1.0 están declarados para fomentar el intercambio de información entre los

equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances. [14]

En un proyecto organizado sobre Open UP 1.0, los esfuerzos personales se convierten en micro incrementos. Estos proveen un ciclo de retroalimentación relativamente corto que permite flexibilidad y mejor adaptación a las decisiones tomadas dentro de cada iteración.

Esta metodología divide el proyecto en iteraciones que son planeadas sobre intervalos de tiempo definidos en semanas, dichas iteraciones se centran dando cumplimiento a los objetivos definidos previamente en el plan para cada una por parte del equipo de desarrollo, donde cada ciclo iterativo debe concebir como resultado un demo o un ejecutable con funcionalidades específicas. Open UP 1.0 en cada iteración del ciclo de vida, incrementa progresivamente los objetivos de las iteraciones anteriores, añadiendo nuevas funcionalidades a las versiones estables del software que se tiene hasta el momento. Dentro del ciclo de vida existen 4 fases: Inicio, Elaboración, Construcción y Transición. (Ver Ilustración 4)

Se elige esta metodología para el desarrollo del modelo porque en la UCI se han obtenido resultados relevantes en cuanto a la agilización de producción de software para equipos de desarrollo y, además, la propuesta a desarrollar no constituye un software de alta complejidad ni está determinado por procesos críticos donde se necesite poseer una amplia documentación sobre su ciclo de desarrollo.

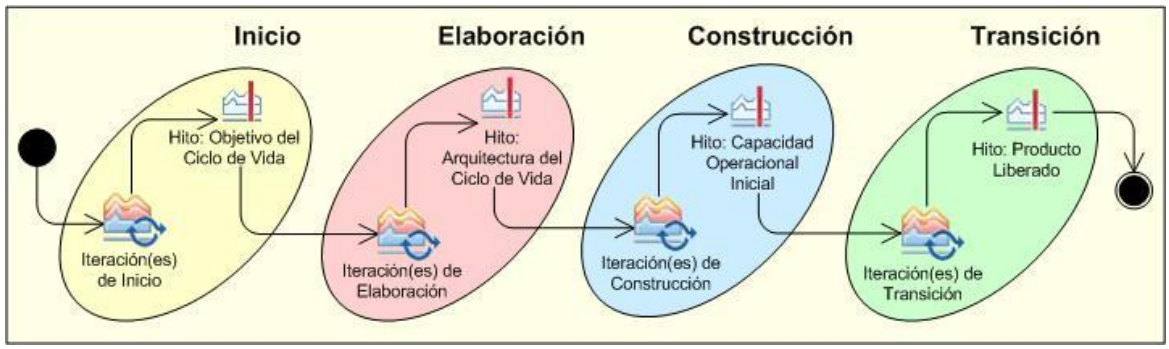


Ilustración 4: Fases en Open UP 1.0.

1.6.3 Lenguaje de Modelado Unificado

Se utiliza Lenguaje de Modelado Unificado (UML) como lenguaje de modelado para el apoyo de la metodología debido a que es un lenguaje que permite modelar, construir y documentar los elementos que forman un producto de software; el cual responde a un enfoque orientado a objetos y puede también

considerarse como un lenguaje de representación visual que permite una abstracción del sistema y sus componentes.

Por otro lado, UML se ha convertido en el estándar internacional e industrial para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación orientada a objetos. Puede utilizarse en todo el ciclo de vida de desarrollo de un software, lo que garantiza el modelado de todas las etapas de desarrollo.

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas de Ingeniería de Software Asistida por Ordenador (CASE) orientadas a objetos. Para ello era necesario definir una notación y semántica común. Es menester tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación.

Fue escogido para modelar la aplicación por las ventajas que brinda para especificar o para describir métodos o procesos del sistema a desarrollar.

1.6.4 Herramienta para el diseño del software: Visual Paradigm for UML 6.4

Visual Paradigm for UML 6.4 es una herramienta CASE potente y fácil de utilizar, que permite el modelado visual UML. Esta herramienta no está diseñada para apoyar una metodología de desarrollo en específico, pero gracias a su flexibilidad se integra perfectamente con Open UP 1.0.

Entre sus características más importantes sobresalen:

- Brinda soporte para varias versiones de UML.
- Brinda una versión gratuita y otra comercial.
- Permite la ingeniería inversa (código a modelo, código a diagrama).
- Permite la generación de código (modelo a código, diagrama a código).
- Permite exportar el diseño de la base de datos para varios gestores.
- Es multiplataforma.

1.6.5 Lenguaje de desarrollo: JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. JavaScript

supera a otros lenguajes del lado del cliente en cuanto a portabilidad, rendimiento, facilidad de uso y documentación.

1.6.6 Lenguaje de desarrollo: Ruby 1.8.6

Ruby 1.8.6 es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su implementación oficial es distribuida bajo una licencia de software libre. La elegante sintaxis que posee se siente natural al leerla y es fácil de escribir. Es interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995.

Ruby 1.8.6 sigue el "principio de la menor sorpresa", lo que significa que el lenguaje debe comportarse de tal manera que minimice la confusión de los usuarios experimentados. Además ha sido descrito como un lenguaje de programación multiparadigma: permite programación procedural (definiendo funciones y variables fuera de las clases, haciéndolas parte del objeto raíz Object), con orientación a objetos (todo es un objeto), o funcionalmente (tiene funciones anónimas, clausuras o closures, y continuations; todas las sentencias tiene valores, y las funciones devuelven la última evaluación).

Este lenguaje tiene muchas virtudes que lo han convertido en la opción de muchos programadores debido a que:

- Brinda facilidad de aprendizaje del lenguaje.
- Posee capacidad de conexión con la mayoría de los manejadores de bases de datos que se utilizan en la actualidad destacando su conectividad con PostgreSQL 8.4.
- Posee capacidad de expandir su potencial utilizando la enorme cantidad de módulos.
- Es multiplataforma.
- Posee una amplia documentación, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en archivos de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Logra una trilogía de perfecta integración: Apache 2.0 - Ruby 1.8.6 - PostgreSQL 8.4.
- No depende de un único proveedor de servicios.
- Su código fuente es abierto y gratuito.

- Existe una gran comunidad de desarrolladores, lo que garantiza que los fallos de funcionamiento se encuentren y reparen rápidamente.

1.6.6.1 Framework: Ruby on Rails 2.3.5

Ruby on Rails 2.3.5 es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo-Vista-Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros framework y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

1.6.7 Sistema Gestor de Base de Datos: PostgreSQL 8.4

PostgreSQL 8.4 es un Sistema Gestor de Base de Datos (SGBD) relacional, orientada a objetos y libre que lleva más de una década de desarrollo y cuenta con una arquitectura probada; ganándose en este tiempo una sólida reputación de confiabilidad e integridad de datos. Funciona en los principales sistemas operativos, incluyendo Linux, UNIX y Windows. También es compatible con el almacenamiento de objetos binarios incluyendo imágenes, sonidos o videos; y utiliza el Lenguaje de Consulta Estructurado (SQL), lenguaje utilizado por todas las Base de Datos (BD) relacionales.

Es considerado como el SGBD de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. Es más eficiente cuando el tamaño de la base de datos es grande. Además, siendo la eficiencia un requisito fundamental en el desarrollo de una Aplicación Web, es fácil entender su selección como SGBD para la aplicación a desarrollar.

1.6.8 Chart Server 1.0.3

Chart Server 1.0.3 es un producto que permite incluir gráficos estadísticos en una página web y otros productos de forma sencilla y segura. Es altamente útil ya que de manera fácil te permite elaborar cualquiera de los gráficos más usados sin tener que implementarlos manualmente.

Entre los tipos de gráficos que te permite generar están:

- Gráficos de pastel.
- Gráficos de barras.
- Gráficos de líneas.
- Gráficos de curvas.
- Gráficos de puntos.
- Gráficos mixtos.

Se decidió utilizarlo en la herramienta ya que actualmente está prestando servicio desde un servidor dentro de la comunidad universitaria, con un equipo de soporte altamente calificado. Y aunque actualmente está en fase de pruebas, ha prestado una asistencia de gran calidad a todos los usuarios que lo han utilizado.

1.6.9 Entorno de Desarrollo Integrado: NetBeans 6.8

NetBeans 6.8 es un IDE visual de código abierto. Su aprendizaje se ha convertido en asunto fundamental para quienes están interesados en el desarrollo de aplicaciones multiplataforma. La programación mediante NetBeans 6.8 se realiza a través de componentes de software modulares, también llamados módulos. Fue fundado en junio del 2000 por la compañía Sun MicroSystems que continúa siendo el patrocinador principal. [15]

Fue elegido como IDE para desarrollar esta aplicación por ser un software con todos los beneficios disponibles en forma gratuita, el cuál ha sido examinado por una comunidad de desarrolladores. Este enfoque de bienes comunes creativos ha permitido una mayor capacidad de uso y ha proporcionado a los desarrolladores mayor flexibilidad. Además posee facilidades para el desarrollo de aplicaciones en Ruby 1.8.6 y Ruby on Rails 2.3.5. Presenta mejoras sobresalientes en las últimas versiones y posibilita con eficiencia desarrollar aplicaciones con el SGBD PostgreSQL 8.4.

1.6.10 Servidor web: Apache 2.0

Un servidor web es un programa que corre sobre el servidor que escucha las peticiones HTTP, éstas le llegan desde el cliente, que son en este caso los navegadores. Dependiendo del tipo de la petición, el servidor web buscará una página web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al navegador que realizó la petición. [16]

El mundo está dividido por dos grandes grupos de servidores web: el Servidor de Información de Internet (IIS) de Microsoft, y el Apache 2.0, un proyecto libre de la Fundación Apache, gratuito y de código abierto.

El servidor web Apache 2.0 es uno de los más potentes del mercado, ofreciendo una perfecta combinación entre estabilidad y sencillez. Hoy en día es el más utilizados, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software multiplataforma y que se distribuye prácticamente con todas las distribuciones de Linux. También es el servidor que utiliza la herramienta de gestión de proyecto Redmine 0.9.2.
Por todas estas fundamentaciones se consideró utilizarlo para acoplar la aplicación a desarrollar.

1.6.11 Herramienta de Gestión de Proyectos: Redmine 0.9.2

Redmine 0.9.2 es una flexible aplicación web para la gestión de proyectos de código abierto bajo la licencia GPL. Se desarrollo usando el framework Ruby on Rails 2.3.5. Es multiplataforma y permite una eficiente vinculación con el SGBD PostgreSQL 8.4. Permite la administración de documentos y archivos, soporta múltiples proyectos, múltiples lenguajes y múltiples BD. Posee una alta seguridad y sus actividades son manejadas con el tiempo. Además fue diseñado con una arquitectura que permite incorporarle extensiones adjuntas con facilidad. Contiene todos los elementos para que un equipo pueda coordinarse y avanzar en los diferentes proyectos. Estos elementos son:

- Vistazo: Visión general del proyecto.
- Roadmap: Muestra el avance del proyecto y el porciento que queda para terminar un hito o versión.
- Peticiciones: Son las unidades de trabajo que pueden ser tareas, errores, mejoras, etc. Estas se asignan a personas y se puede ir siguiendo su evolución, tiempo dedicado, comentarios, etc.
- Noticias del proyecto: Permite gestionar noticias.
- Documentos: Permite gestionar documentación.
- Wiki: Permite crear documentación.
- Repositorio: Accede a conectarse a un repositorio de código.
- Mi página: Permite ver todo lo relacionado con nuestras tareas pendientes, las que hemos asignado, calendario, etc.
- Configuración del proyecto: Permite personalizar los proyectos con campos específicos y otros temas.

En la UCI se ha estado utilizando desde hace 2 años y aunque todavía está en una etapa de prueba ha demostrado trabajar con eficacia y calidad.

***

1.7 Conclusiones del capítulo

En este capítulo se efectuó un análisis general del proceso de gestión de riesgos abordando sus características y conceptos, enfocando en el análisis cuantitativo de riesgos y en las técnicas para su modelado. Además se abordaron las tendencias actuales de las herramientas que se utilizan para la gestión de riesgos, realizándose una valoración de las tecnologías a utilizar, las cuales por ser libres y por su disponibilidad, viabilizan el diseño y la implementación del sistema de manera efectiva.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción del capítulo

En el presente capítulo se exponen las características y propiedades que debe tener la aplicación, definiéndose los requisitos funcionales y no funcionales. Basado en estos requerimientos se explica brevemente la propuesta de sistema. Se analiza, además, el diagrama de CU del sistema y la descripción de cada uno de estos.

2.2 Visión del Sistema

En todos los proyectos productivos de la UCI, el equipo de desarrollo debe tomar decisiones sobre factores de incertidumbre. Para lograr planificar el futuro en estas circunstancias, es necesario que el equipo se enfrente con situaciones adversas, sacrificándose más de lo cotidiano en pos de preparar los escenarios no deseados. Por tal motivo, no basta con hacer simples análisis de sensibilidad para evaluar cómo afectan las variables críticas a los objetivos del proyecto, sino que es necesario utilizar herramientas de base científica que aporten un mayor soporte técnico a las proyecciones futuras.

A partir de estas circunstancias, surge como necesidad desarrollar una herramienta de base científica que permita una óptima toma de decisiones por parte de los directivos de proyectos después de un correcto proceso de análisis cuantitativo de riesgos. Proceso que no está actualmente automatizado en la UCI. (Ver **¡Error! No se encuentra el origen de la referencia.**)

Esta investigación se propone responder a las demostradas necesidades actuales, implementando una herramienta que permita realizar el análisis cuantitativo de riesgos a través de una extensión funcional al gestor de proyectos Redmine; que finalmente le reporte al usuario una salida altamente útil de información de la cual puedan beneficiarse los directivos del proyecto a la hora de tomar decisiones.

2.3 Selección de la propuesta

Para desarrollar el proceso de análisis cuantitativo de riesgos de forma automatizada se propuso realizar un componente para el Redmine. Por lo que se hizo necesario realizar un estudio de las técnicas existentes para este proceso y proponer una a aplicar en el componente o extensión a desarrollar.

Realizada la investigación, primeramente se separaron las técnicas en dos grupos:

- Técnicas de recopilación y representación de datos.
- Técnicas de análisis cuantitativo de riesgos y de modelado.

Se propusieron por grupo tres y cuatro técnicas respectivamente, para comparar y seleccionar las que más se ajustaban a las características que debía tener la aplicación a desarrollar. Los criterios escogidos para establecer la comparación fueron:

- Grado de especificación: Capaz de proveer un conjunto de soluciones lo más clasificadas y detalladas posibles, para una mayor comprensión de los efectos.
- Capacidad de aceptación: Una técnica que haya arrojado relevantes resultados a favor de la prevención de los riesgos y por ende posea una mayor difusión en cuanto a su uso en el mundo.
- Adaptación a la aplicación: Las características de la técnica deben ajustarse a las necesidades de la herramienta.

De las técnicas de recopilación y representación de datos se analizó:

- Juicio de expertos: Ha sido una de las técnicas más utilizadas para obtener la opinión de personas conocedoras de la materia -internas o externas-, como expertos en ingeniería y en estadística, que validen los datos referentes a los riesgos identificados; sin embargo, se basa en el conocimiento que tengan determinadas personas de los mismos, que por lo general no son muy abarcadores, quedando en la mayoría de los casos la expectativa de no haber analizado todas las categorías y el posible comportamiento de los riesgos. [13]
- Entrevista: Es menos propensa a utilizar que la experiencia en el personal, con la diferencia que cuando se analizan los riesgos se realiza con un número de personas mayor y concentrándose en un mismo lugar, donde se tiene la posibilidad de obtener una mayor variedad de respuestas por parte de los participantes. A su vez, requiere de la disposición de las personas a participar en la técnica, porque de lo contrario no se obtendrá el efecto deseado.
- Distribuciones de probabilidad: Método o función estadística que describe como se espera que varíen los resultados, asignando a cada suceso o factor definido la probabilidad de que ocurra. La distribución de probabilidad está definida sobre el conjunto de todos los eventos en un rango de valores. Dado que estas distribuciones se ocupan de las expectativas, son modelos de gran utilidad para hacer inferencias y tomar decisiones en condiciones de incertidumbre; teniendo gran

repercusión y uso a nivel mundial, ilustran de forma detallada la repercusión de los factores que influyen en el comportamiento de un determinado riesgo. [13]

De las técnicas de análisis cuantitativo de riesgos y de modelado se analizó:

- Análisis de sensibilidad: Ayuda a determinar qué riesgos tienen el mayor impacto posible sobre el proyecto. Este método examina en qué medida la incertidumbre de cada elemento del proyecto afecta al objetivo que está siendo examinado, cuando todos los demás elementos inciertos se mantienen con sus valores de línea base. Posee, no obstante, un inconveniente, ya que resulta menos óptimo si se usa para un número grande de variables con un grado de incertidumbre marcado. No es valorativamente adaptable a la aplicación que se quiere desarrollar.
- Análisis del valor monetario esperado: Es un concepto estadístico que calcula el resultado promedio de los valores analizados cuando el futuro incluye escenarios que pueden ocurrir o no, es decir, realiza análisis con incertidumbre. No se recomienda cuando se desean obtener resultados que estén poco sujetos a errores, por lo que no es recomendable para la herramienta a desarrollar. [13]
- Análisis mediante árbol de decisiones: Normalmente se estructura usando un diagrama de árbol de decisiones que describe una situación que se está considerando, las implicaciones de cada una de las opciones disponibles y los posibles escenarios. Incorpora el coste de cada opción disponible, las probabilidades de cada escenario posible y las recompensas de cada camino lógico alternativo. Su uso se vuelve difícil cuando se posee un modelo con muchos factores relacionados que influyen en un mismo riesgo.
- Modelado y simulación: Proceso de construcción de un modelo matemático o lógico que representa un sistema con el fin de experimentar con él, de tal forma que permita ver su comportamiento y ayude a tomar una decisión. Las simulaciones normalmente se realizan usando la técnica de Montecarlo. En una simulación, el modelo del proyecto se calcula muchas veces -iteraciones-, utilizando valores de entrada seleccionados al azar de una función de distribución de probabilidad de cada variable. [13] Se recomienda el uso del modelado y la simulación para el análisis de los riesgos de costes y del cronograma, pues son más efectivos y están menos sujetos a errores de aplicación que el análisis del valor monetario esperado.

Dado que:

- Las distribuciones de probabilidad son la forma más realista de describir el comportamiento que puede tener un factor o variable con incertidumbre.
- Poseen una excelente integración ya que la simulación se basa en distribuciones de probabilidad para determinar en cada iteración el comportamiento de dicho factor en el modelo que se desea simular.
- La simulación de Montecarlo puede incluir todas las combinaciones posibles de las variables que afectan los resultados de un proyecto.
- La simulación de Montecarlo no sólo brinda el comportamiento más probable del riesgo, sino también su distribución de probabilidad, de modo que todos los resultados posibles pueden ser analizados.
- El número de factores influyentes en el riesgo, que pueden ser considerados en el análisis, es muy grande. Todas las combinaciones posibles de los estados de dichos factores se incluyen en el problema, suministrando un método de análisis muy riguroso y amplio.

Es importante mencionar además que la simulación de Montecarlo permite una serie de ventajas sobre el análisis determinista:

- Resultados probabilísticos: Los resultados muestran no sólo lo que puede suceder, sino la probabilidad de un resultado.
- Resultados gráficos: Gracias a los datos que genera una simulación de Montecarlo, es fácil crear gráficos de diferentes resultados con las posibilidades de que estos sucedan. Esto es importante para comunicar las soluciones a otras personas interesadas.
- Análisis de sensibilidad: Con sólo unos pocos datos, en los análisis deterministas es más difícil ver las variables que más afectan el resultado. En la simulación de Montecarlo, resulta más fácil ver qué variables introducidas tienen mayor influencia sobre las soluciones finales.
- Análisis de escenario: En los modelos deterministas resulta muy difícil modelar diferentes combinaciones de valores de entrada, con el fin de ver los efectos de situaciones verdaderamente diferentes. Usando la simulación de Montecarlo, los analistas pueden ver exactamente los valores que tiene cada variable en correspondencia con ciertos resultados, lo cual es muy valioso para profundizar en los análisis.

- Correlación de variables de entrada: En la simulación de Montecarlo es posible modelar relaciones entre diferentes variables de entrada. Esto es importante para averiguar con precisión la razón real por la que, cuando algunos factores suben, otros suben o bajan paralelamente. [13]

Después de realizar la valoración de cada una de las técnicas; observando sus características, ventajas y desventajas, se llegó a la conclusión de que las más factibles para el desarrollo de la herramienta son: de las técnicas de recopilación y representación de datos, las distribuciones de probabilidad; y de las técnicas de análisis cuantitativo de riesgos y de modelado, la de modelado y simulación a través de la simulación de Montecarlo. (Ver Ilustración 5)

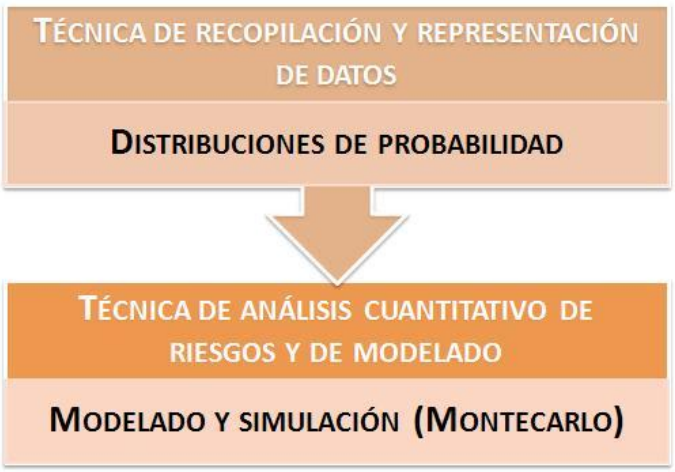


Ilustración 5: Técnicas más competentes para desarrollar la aplicación.

A partir de todos los análisis anteriormente realizados, se ha decidido implementar la herramienta siguiendo como guía los argumentos estudiados en el capítulo anterior; quedando vigente la siguiente propuesta:

La aplicación contará con la capacidad en el Redmine de brindarle al usuario -registrado como gestor de riesgos- la posibilidad de efectuar el análisis cuantitativo de riesgos a partir de la realización satisfactoria de los dos procesos anteriores -identificación de riesgos y análisis cualitativo de riesgos-; pues la entrada de la aplicación está dada por el registro de riesgos -resultado de los procesos anteriores-. A partir de esto el usuario deberá seleccionar solo un riesgo para seguir avanzando, una vez hecho esto, podrá insertarle

variables de incertidumbre al mismo; haciéndose uso de la técnica distribuciones de probabilidad, ya que cada variable ajusta su comportamiento a una distribución.

Luego se creará el registro de variables del riesgo, el cual es también modificable; y se deberá insertar una fórmula que defina la evaluación del riesgo. Entonces se podrán desarrollar las iteraciones con los valores aleatorios, donde cada simulación arrojará resultados que se modelarán estadísticamente mediante un histograma, una tabla de distribución de frecuencias, y los estadígrafos más conocidos; resultados que serán de gran valor al usuario para ayudarlo en la toma de decisiones ante la incertidumbre.

2.4 Especificación de los requisitos del software

Los requisitos son una representación documentada de las condiciones o capacidades que debe alcanzar un sistema o componente de un sistema para satisfacer un contrato estándar, u otro documento impuesto formalmente; que tiene como propósito, establecer un entendimiento común entre el usuario y el proyecto de software sobre los requisitos que solicita el usuario, los cuales serán abordados en dicho proyecto.

Se agrupan en dos grandes categorías:

- Requisitos funcionales.
- Requisitos no funcionales.

A continuación se muestran los que han sido definidos para la realización y desarrollo del sistema que se propone. La distribución de los mismos se ha definido a partir de las funcionalidades que componen dicho sistema y de los servicios que este debe ofrecer.

2.4.1 Requisitos funcionales

RF 1 El sistema debe permitirle al gestor de riesgos la posibilidad de seleccionar el riesgo a analizar cuantitativamente a partir de la lista de riesgos priorizados.

RF 2 El sistema debe ofrecerle al gestor de riesgos la posibilidad de gestionar las variables de los riesgos priorizados.

> **RF 2.1** El sistema debe brindarle al gestor de riesgos la posibilidad de adicionar las variables de los riesgos priorizados.
>
> **RF 2.2** El sistema debe permitirle al gestor de riesgos la posibilidad de modificar las variables de los riesgos priorizados.

RF 2.3 El sistema debe ofrecerle al gestor de riesgos la posibilidad de eliminar las variables de los riesgos priorizados.

RF 2.4 El sistema debe brindarle al gestor de riesgos la posibilidad de ver un listado de las variables de los riesgos priorizados.

RF 3 El sistema debe permitirle al gestor de riesgos la posibilidad de gestionar la fórmula que propiciarán los resultados de la simulación.

RF 3.1 El sistema debe admitirle al gestor de riesgos la posibilidad de insertar la fórmula que propiciarán los resultados de la simulación.

RF 3.2 El sistema debe permitirle al gestor de riesgos la posibilidad de modificar la fórmula que propiciarán los resultados de la simulación.

RF 4 El sistema debe permitirle al gestor de riesgos la posibilidad de iniciar la simulación de Montecarlo.

RF 4.1 El sistema debe permitirle al gestor de riesgos la posibilidad de almacenar todos los resultados de la simulación.

RF 5 El sistema debe ofrecerle al gestor de riesgos la posibilidad de observar desde varios puntos de vista estadísticos los resultados de la simulación.

RF 5.1 El sistema debe ofrecerle al gestor de riesgos la posibilidad de observar resultados de cada iteración.

RF 5.2 El sistema debe brindarle al gestor de riesgos la posibilidad de observar la tabla de distribución de frecuencias.

RF 5.3 El sistema debe ofrecerle al gestor de riesgos la posibilidad de observar los valores de todos los estadígrafos.

RF 5.4 El sistema debe permitirle al gestor de riesgos la posibilidad de observar la tabla de percentiles.

RF 5.5 El sistema debe brindarle al gestor de riesgos la posibilidad de observar el histograma en gráfico de barras y en gráfico de pastel.

RF 5.6 El sistema debe ofrecerle al gestor de riesgos la posibilidad de exportar todos los resultados en formato PDF.

2.4.2 Requisitos no funcionales

Usabilidad:

- La aplicación luego de instalada deberá visualizarse con calidad en los principales navegadores existentes como son: Mozilla Firefox, Microsoft Internet Explorer, Opera y Google Chrome.

Fiabilidad:

- El sistema debe ser capaz de mantener la calidad de los datos de manera que garantice su integridad durante su aplicación, procesamiento y almacenamiento en la BD.
- Cada vez que se detecta un cambio en la BD se debe replicar de manera automática que se ha efectuado el mismo.

Funcionamiento:

- Los tiempos de respuestas del sistema serán de corta duración, así como la simulación que se efectuará.

Soportabilidad:

- El sistema debe ser de fácil instalación, configuración y puesta en marcha; de capacidad escalable y programado orientado a objeto.
- El sistema debe funcionar sobre un servidor Apache 2.0 y utilizar PostgreSQL 8.4 como Sistema Gestor de BD.

Hardware:

- Para PC Cliente: Procesador Pentium 233 MHz (recomendado 500 MHz o mayor), 512 MB de Memoria de Acceso Aleatorio (RAM), dispositivo de red de al menos 10 Mbits de velocidad de trasmisión.
- Para PC Servidora: Procesador Pentium 500 MHz, 1 GB de RAM, 5 GB de capacidad del disco duro, dispositivo de red de al menos 10 Mbits de velocidad de trasmisión.

Apariencia o interfaz externa:

- La interfaz del sistema será a través de un diseño sencillo, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema. Debe ser además un diseño formal y serio teniendo en cuenta el fin con el que se desarrolla la aplicación, aplicando colores serios y utilizados normalmente en las funcionalidades del Redmine como el blanco y el azul.
- El sistema proporcionará claridad y una correcta organización de la información, permitiendo la interpretación inequívoca de ésta.

- El diseño de la interfaz gráfica deberá garantizar la distinción visual entre los elementos del sistema.

Implementación:

- El sistema debe ejecutarse en diversas plataformas de hardware y software.
- EL sistema se realizará utilizando el lenguaje de programación Ruby 1.8.6 con el framework Ruby on Rails 2.3.5, PostgreSQL 8.4 como SGBD y como herramienta para programación NetBeans 6.8. Para la modelación de los diagramas UML se empleará el Visual Paradigm 6.4 y como metodología de desarrollo de software Open UP 1.0.
- El sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte.

Seguridad:

- El sistema debe garantizar que la información sea registrada, visualizada, eliminada y actualizada únicamente por la persona que posea los privilegios correspondientes.
- El sistema validará rigurosamente todos los datos de entrada y de salida en el intercambio de información con el usuario.

Licencia:

- Uso de licencias de código abierto en los componentes que conforman el desarrollo del sistema cumpliendo con los principios básicos de software libre.

Documentación del sistema:

- Debe existir documentación para el trabajo con el sistema dependiendo de las tareas que realice el usuario.

Normas Aplicables:

- Entorno para la ingeniería de desarrollo de software Open UP.

2.5 Modelo del Sistema

2.5.1 Actor del Sistema

Actor	Justificación
Gestor de riesgos.	Rol con permisos para interactuar con todas las funcionalidades del sistema. Podrá gestionar las variables de todos los riesgos, gestionar las fórmulas de todos los riesgos, realizar la simulación de Montecarlo y

	tener acceso a todos los resultados de dicha simulación. (Ver Ilustración 6)

Tabla 1: Actor del Sistema.

2.5.2 Diagrama de CU del Sistema

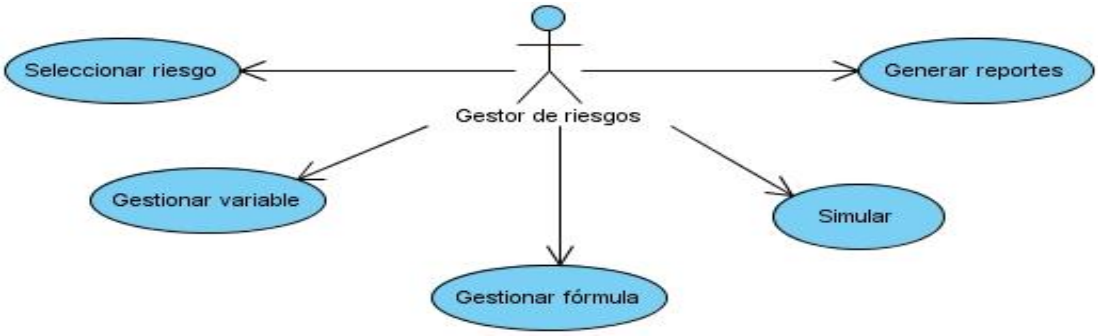


Ilustración 6: Diagrama de CU del Sistema.

2.5.3 Descripción de los CU del Sistema

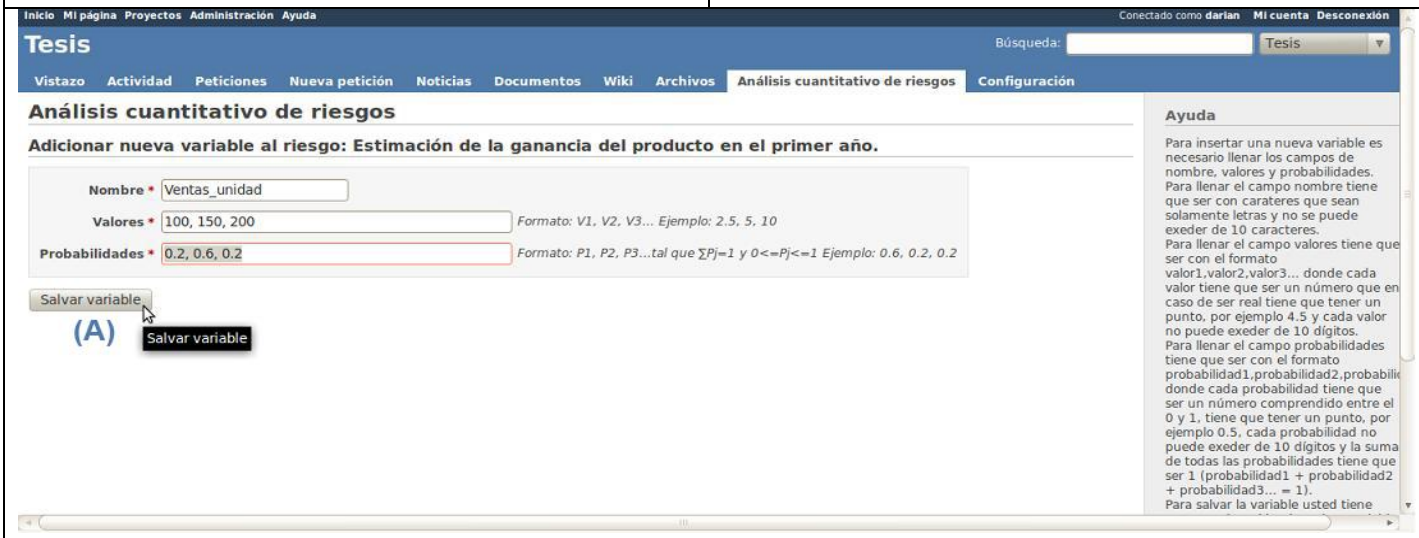
2.5.3.1 Descripción del CU del Sistema Gestionar variable

Caso de uso:	Gestionar variable.
Actor(es):	Gestor de riesgos.
Descripción:	El sistema debe ofrecerle al gestor de riesgos la posibilidad de gestionar las variables de los riesgos priorizados.
Referencias:	RF 2, RF 2.1, RF 2.2, RF 2.3, RF 2.4.
Precondiciones:	El usuario debe estar previamente autenticado. Debe existir al menos un riesgo priorizado y seleccionado. Si desea modificar o eliminar alguna variable, debe existir al menos una en el sistema.
Poscondiciones:	Se insertó, modificó o eliminó alguna variable al riesgo seleccionado, actualizándose la tabla de variables.
Flujo normal de los eventos:	

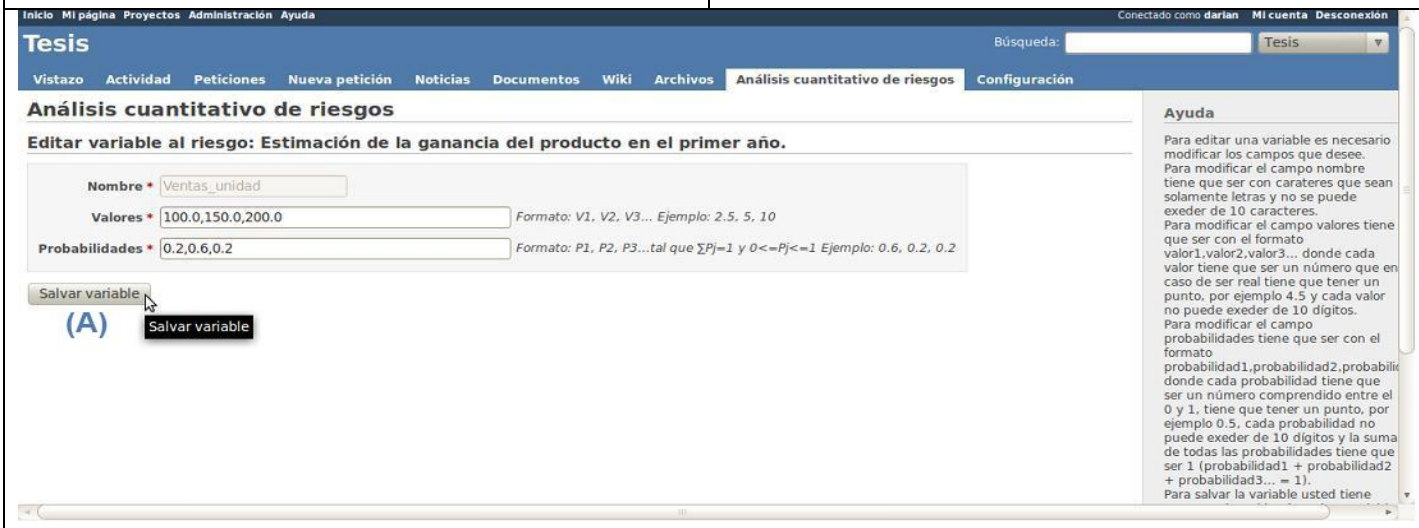
1 El CU se inicia cuando el usuario selecciona la opción “Seleccionar riesgo” en la fila de dicho riesgo.	1.1 El sistema muestra una tabla con la lista de variables del riesgo seleccionado. Si el usuario desea adicionar una variable, selecciona la opción “Nueva variable” (A): ver sección “Adicionar variable”. Si el usuario desea modificar una variable selecciona la opción “Editar variable” en la fila de la variable (B): ver sección “Modificar variable”. Si el usuario desea eliminar una variable, selecciona la opción “Eliminar variable” en la fila de la variable (C): ver sección “Eliminar variable”.
Flujo alterno de los eventos:	
	1.1 El sistema no pudo mostrar la tabla con la lista de variables del sistema y muestra un mensaje informándolo.

Sección Adicionar variable:	
Flujo normal de los eventos de la sección Adicionar variable:	
	2.1 El sistema muestra el formulario para insertar la nueva variable.
3 El usuario introduce los datos de la nueva variable y selecciona la opción “Salvar variable”. (A)	3.1 El sistema valida los datos de entrada.

	3.2 El sistema almacena la información en su BD y muestra un mensaje de confirmación.
Flujo alterno de los eventos de la sección Adicionar variable:	
	3.1 Si los datos no son correctos el sistema muestra un mensaje informándolo.
	3.2 El sistema no pudo almacenar la información en su BD y muestra un mensaje informándolo.



Sección Modificar variable:	
Flujo normal de los eventos de la sección Modificar variable:	
	4.1 El sistema muestra el formulario para modificar la variable.
5 El usuario modifica los datos de la variable y selecciona la opción “Salvar variable”. (A)	5.1 El sistema valida los datos modificados.
	5.2 El sistema almacena la información modificada en su BD y muestra un mensaje de confirmación.
Flujo alterno de los eventos de la sección Modificar variable:	
	5.1 Si los datos modificados no son correctos el sistema muestra un mensaje informándolo.
	5.2 El sistema no pudo almacenar la información

	modificada en su BD y muestra un mensaje de confirmación.
	
Sección Eliminar variable:	
Flujo normal de los eventos de la sección Eliminar variable:	
	6.1 El sistema cuestiona al usuario sobre lo que está a punto de hacer. (A)
7 El usuario reafirma su decisión.	7.1 El sistema elimina la variable en su BD y muestra un mensaje de confirmación.
Flujo alterno de los eventos de la sección Eliminar variable:	
	7.1 El sistema no pudo eliminar la variable en su BD y muestra un mensaje informándolo.

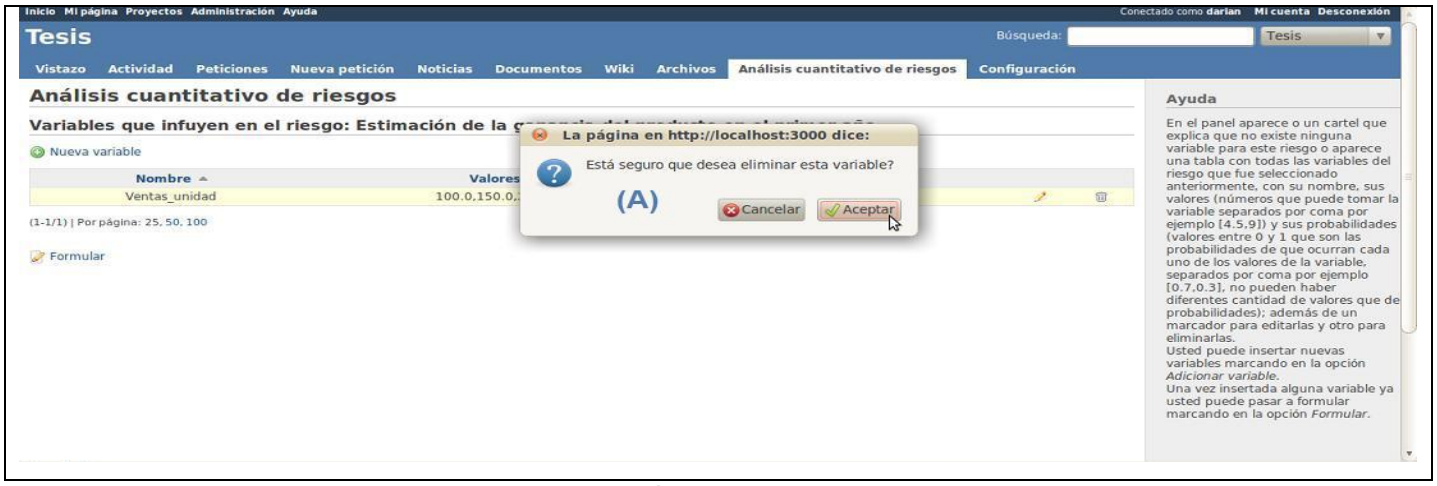
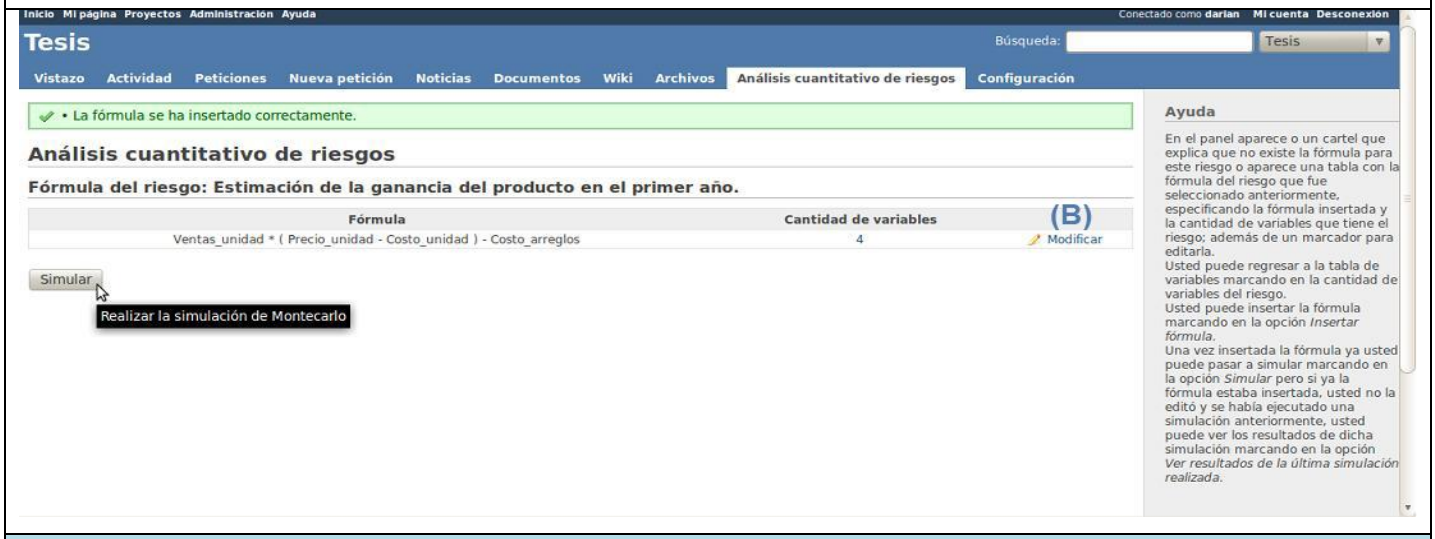


Tabla 2: Descripción del CU Gestionar variable.

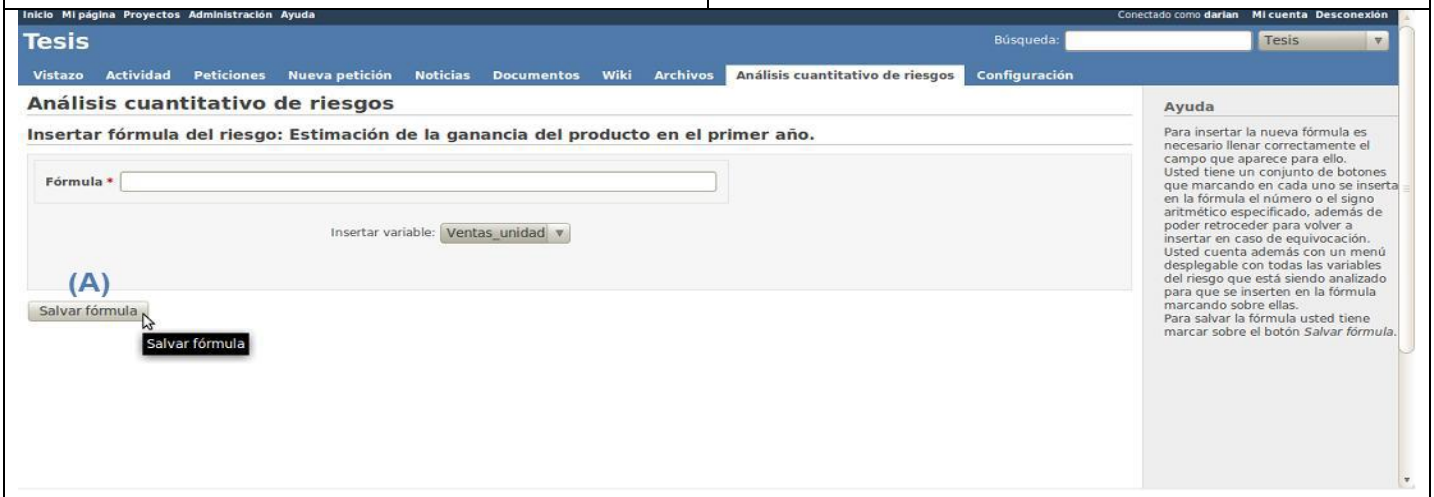
2.5.3.2 Descripción del CU del Sistema Gestionar fórmula

Caso de uso:	Gestionar fórmula.
Actor(es):	Gestor de riesgos.
Descripción:	El sistema debe permitirle al gestor de riesgos la posibilidad de gestionar la fórmula que propiciarán los resultados de la simulación.
Referencias:	RF 3, RF 3.1, RF 3.2.
Precondiciones:	El usuario debe estar previamente autenticado. Debe existir al menos un riesgo priorizado y seleccionado en el sistema. Deben existir al menos una variable en el sistema para el riesgo seleccionado. Si desea modificar la fórmula, debe existir una en el sistema.
Poscondiciones:	Se insertó o modificó la fórmula que retorna los valores de cada iteración.
Flujo normal de los eventos:	
1 El CU se inicia cuando el usuario selecciona la opción “Formular”.	1.1 El sistema muestra la opción “Insertar fórmula” si no existe la fórmula para el riesgo seleccionado o la o la opción “Modificar fórmula” si existe la fórmula para el riesgo seleccionado. Si el usuario desea insertar la fórmula selecciona la opción “Insertar fórmula”. (A) Si el usuario desea modificar la fórmula selecciona la opción “Modificar fórmula”. (B)



Sección Insertar fórmula:	
Flujo normal de los eventos de la sección Insertar fórmula:	
2 El usuario selecciona la opción “Insertar fórmula”.	2.1 El sistema muestra el formulario para insertar la fórmula.
3 El usuario introduce la fórmula y selecciona “Salvar fórmula”. (A)	3.1 El sistema valida los datos de entrada.
	3.2 El sistema almacena la información en su BD y muestra un mensaje de confirmación.
Flujo alterno de los eventos de la sección Insertar fórmula:	

	3.1 Si los datos no son correctos el sistema muestra un mensaje informándolo.
	3.2 El sistema no pudo almacenar la información en su BD y muestra un mensaje informándolo.



Sección Modificar fórmula:	
Flujo normal de los eventos de la sección Modificar fórmula:	
4 El usuario selecciona la opción “Modificar fórmula”.	4.1 El sistema muestra el formulario para modificar la fórmula.
5 El usuario introduce la fórmula y selecciona “Salvar fórmula”. (A)	5.1 El sistema valida los datos de entrada.
	5.2 El sistema almacena la información en su BD y muestra un mensaje de confirmación.
Flujo alterno de los eventos de la sección Modificar fórmula:	
	5.1 Si los datos no son correctos el sistema muestra un mensaje informándolo.
	5.2 El sistema no pudo almacenar la información en su BD y muestra un mensaje informándolo.

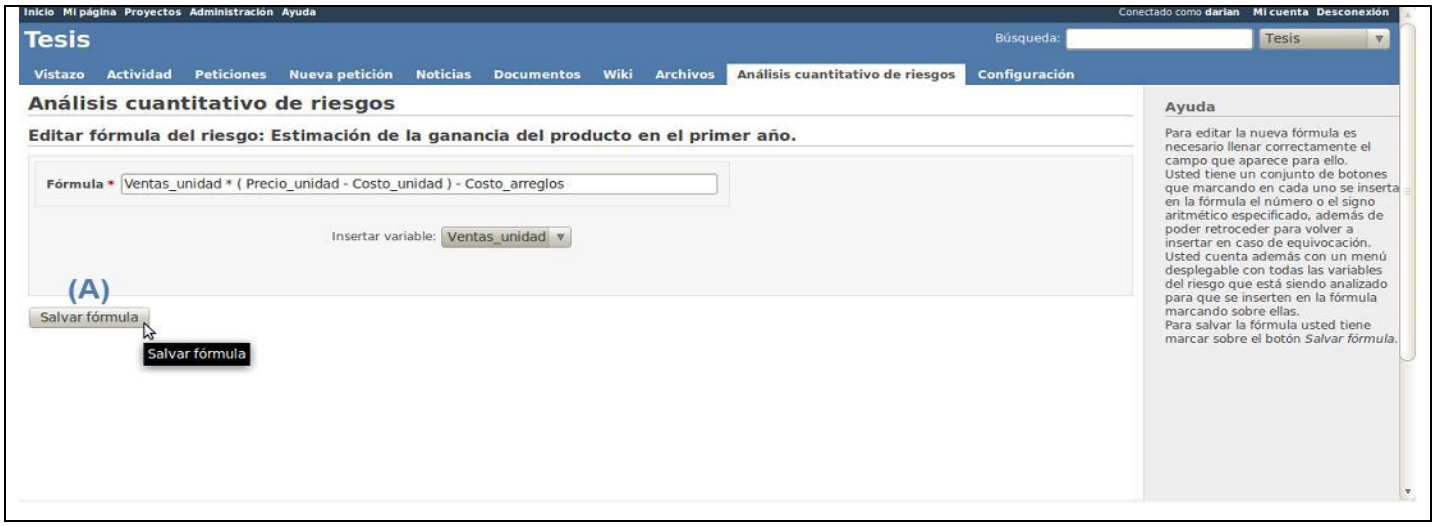


Tabla 3: Descripción del CU Gestionar fórmula.

2.5.3.3 Descripción del CU del Sistema Simular

Caso de uso:	Simular.
Actor(es):	Gestor de riesgos.
Descripción:	El sistema debe permitirle al gestor de riesgos la posibilidad de iniciar la simulación de Montecarlo.
Referencias:	RF 4, RF 4.1.
Precondiciones:	El usuario debe estar previamente autenticado. Debe existir al menos un riesgo priorizado y seleccionado. Deben estar insertadas la fórmula y las variables del riesgo al que se quiere analizar.
Poscondiciones:	Se simuló y el sistema guardó una amplia gama de resultados en la BD.
Flujo normal de los eventos:	
1 El CU se inicia cuando el usuario selecciona la opción “Simular” del menú. (A)	1.1 El sistema simula y crea una nueva tabla en la BD donde almacena los resultados de cada iteración de la simulación.
Flujo alterno de los eventos:	
	1.1 El sistema no ha podido simular, ni crear una nueva tabla en la BD donde almacena los resultados

	de cada iteración de la simulación y muestra un mensaje informándolo.
	

Tabla 4: Descripción del CU Simular.

Las restantes descripciones de CU se encuentran en los anexos. (Ver **¡Error! No se encuentra el origen de la referencia.**)

2.6 Conclusiones del capítulo

En este capítulo se obtuvieron los requisitos funcionales y no funcionales de la aplicación; y tomándolos como base, se planteó la propuesta del sistema. Además, se identificaron y describieron los CU y los actores presentes en el sistema. Los artefactos generados son el punto de partida para el desarrollo del diseño del sistema.

CAPÍTULO 3: ARQUITECTURA Y DISEÑO DEL SISTEMA

3.1 Introducción del capítulo

En este capítulo se explican todos los patrones utilizados como el patrón arquitectónico y los patrones de diseño. Se realiza el diseño del sistema a desarrollar. Para ello se modelan los diagramas de clases y de secuencias del diseño para cada CU. Se diseña también la estructura de la base de datos a utilizar, tomando como base las clases persistentes, con el objetivo de garantizar una adecuada comprensión del diseño.

3.2 Patrones utilizados

Un patrón es un problema-solución, con un nombre, que codifica o estandariza buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

Existen diversas clases de patrones como son:

- Patrones de CU.
- Patrones de arquitectura.
- Patrones de diseño.
- Patrones Generales de Software para la Asignación de Responsabilidades (GRASP).

3.2.1 Patrón de casos de uso: CRUD completo

La experiencia en la utilización de CU ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. Dado un contexto y un problema a resolver, estas técnicas han mostrado ser la solución adoptada en la comunidad del desarrollo de software. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática.

Para realizar el diagrama de CU del sistema se utilizó el patrón de CU: CRUD completo. Este patrón consta de un CU, llamado Información CRUD o Gestionar información, que modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación.

3.2.2 Patrón de arquitectura: Modelo-Vista-Controlador

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. Para diseñar la arquitectura de la aplicación se escogió el patrón Modelo-Vista-Controlador (MVC) que actualmente es uno de los más utilizados para el desarrollo de aplicaciones web y es el utilizado en la arquitectura del Redmine. Este patrón separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. La Vista es la página HTML y el código que provee de datos dinámicos a la página, el Modelo es el SGBD y el Controlador representa la lógica del Negocio (Ver Ilustración 7).

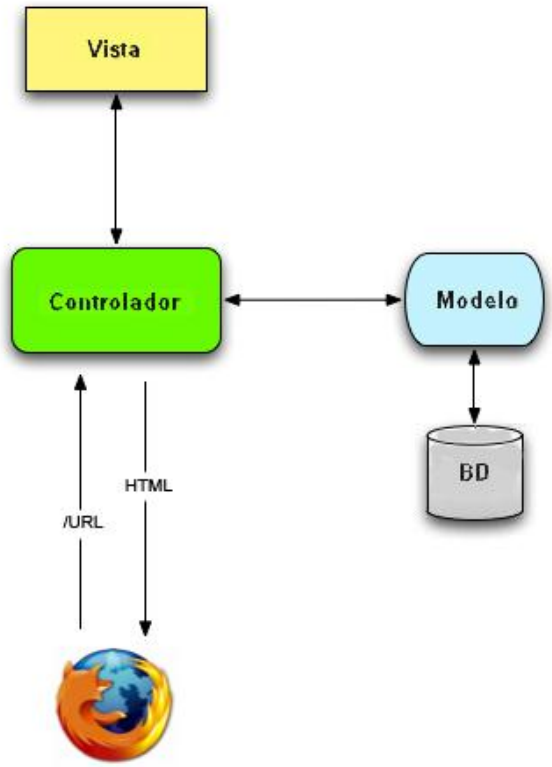


Ilustración 7: Representación esquemática de la arquitectura del patrón MVC.

Para representar detalladamente la arquitectura del patrón MVC utilizaremos el siguiente ejemplo en el cuál un usuario solicita el listado de todos los usuarios en una aplicación web: (Ver Ilustración 8)

***

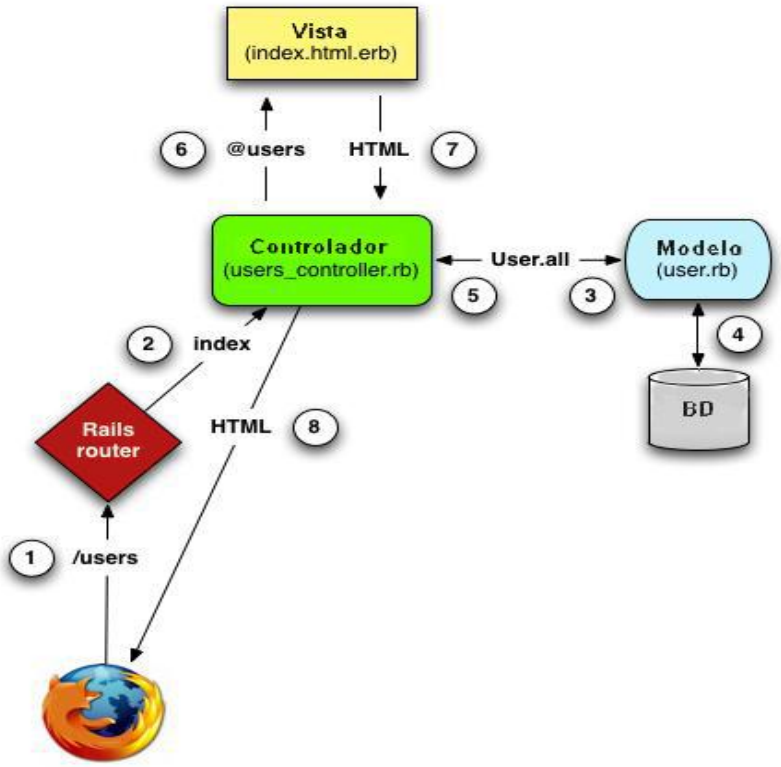


Ilustración 8: Representación detallada de la arquitectura del patrón MVC.

1- El navegador publica una demanda en la URL */users*.
2- Rails encamina */users* a la acción *index* en la controladora *Users*.
3- La acción *index* pide al modelo *User* recobrar todos los usuarios *(User.all)*.
4- El modelo *User* extrae todos los usuarios de la Base de Datos.
5- El modelo *User* envía el listado de los usuarios a la controladora.
6- La controladora captura los usuarios en la variable *@users*, la cuál es pasada a la vista *index*.
7- La vista aprovecha el Ruby incrustado para mostrar las páginas como *HTML*.
8- El controlador pasa el *HTML* al navegador.

Este patrón de arquitectura de software se desarrolla rápidamente y de forma modular. Las funciones de la aplicación se separan en tres componentes distintos: Modelo, Vista y Controlador (Ver Ilustración 9), lo cual hace que la aplicación sea muy ligera y permita hacer cambios en una parte de la aplicación sin que las demás se vean afectadas.

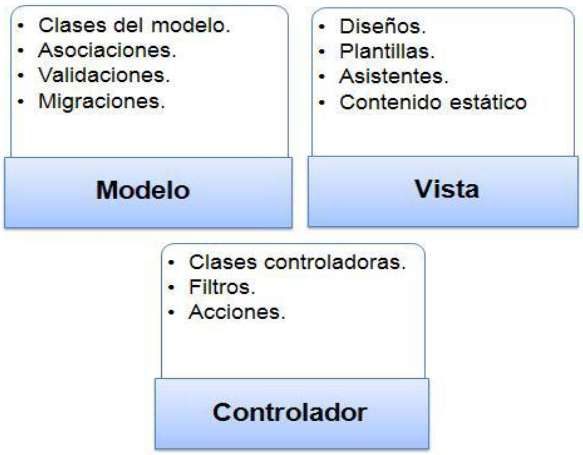


Ilustración 9: Componentes del MVC.

- Modelo: Representa los datos y reglas del Negocio.
- Vista: Permite al usuario interactuar con los objetos del Modelo.
- Controlador: Se encarga de atender las peticiones de los usuarios y realizar los cambios necesarios tanto en el Modelo como en la Vista.

Algunas de las ventajas que proporciona la utilización de este patrón son: [17]

- Sencillez para crear distintas representaciones de los mismos datos.
- Reutilización de los componentes.
- Simplicidad en el mantenimiento de los sistemas.
- Los desarrollos suelen ser más escalables.

3.2.3 Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño estructurado común, que lo hace útil para la creación de diseños orientados a objetos reutilizables. Definen una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariables de todas las soluciones. [18]

Específicamente los aplicados fueron:

- Instancia única: Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Este patrón provee una única instancia global y su utilización

posibilitará tener creado solamente un objeto de la clase que se encargue de establecer la conexión a la BD en todo momento.

- Active record: Plantea que el objeto contenga los datos presentes en una fila de la tabla que represente y encapsule la lógica necesaria para interactuar con la base de datos. De esta forma el acceso a datos se realiza de manera más sencilla y uniforme. Es un patrón principal en el diseño del Redmine.

3.2.4 Patrones Generales de Software para la Asignación de Responsabilidades

Los Patrones Generales de Software para la Asignación de Responsabilidades (GRASP) utilizados son:

- Controlador: Este patrón se evidencia en las clases controladoras, al ellas tener la responsabilidad de recibir o manejar un evento del sistema. Por ejemplo: cuando reciben las peticiones de la Vista. En la aplicación diseñada se dividen los eventos del sistema en un mayor número de clases controladoras para poder aumentar la cohesión y disminuir el acoplamiento.
- Bajo acoplamiento: El bajo acoplamiento se muestra al existir la menor dependencia posible entre clases, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- Alta cohesión: Este es también un principio que se debe tener en cuenta en todas las decisiones de diseño. Se da una alta cohesión cuando los elementos de un componente, una clase por ejemplo, colaboran para producir algún comportamiento bien definido. Una clase de alta cohesión posee un número relativamente pequeño de responsabilidades, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Este patrón mejora la calidad y facilidad con que se puede entender el diseño, genera un bajo acoplamiento y permite fomentar la reutilización.

3.3 Diagramas de Clases del Diseño

3.3.1 Diagrama de Clases del Diseño del CU Gestionar variable

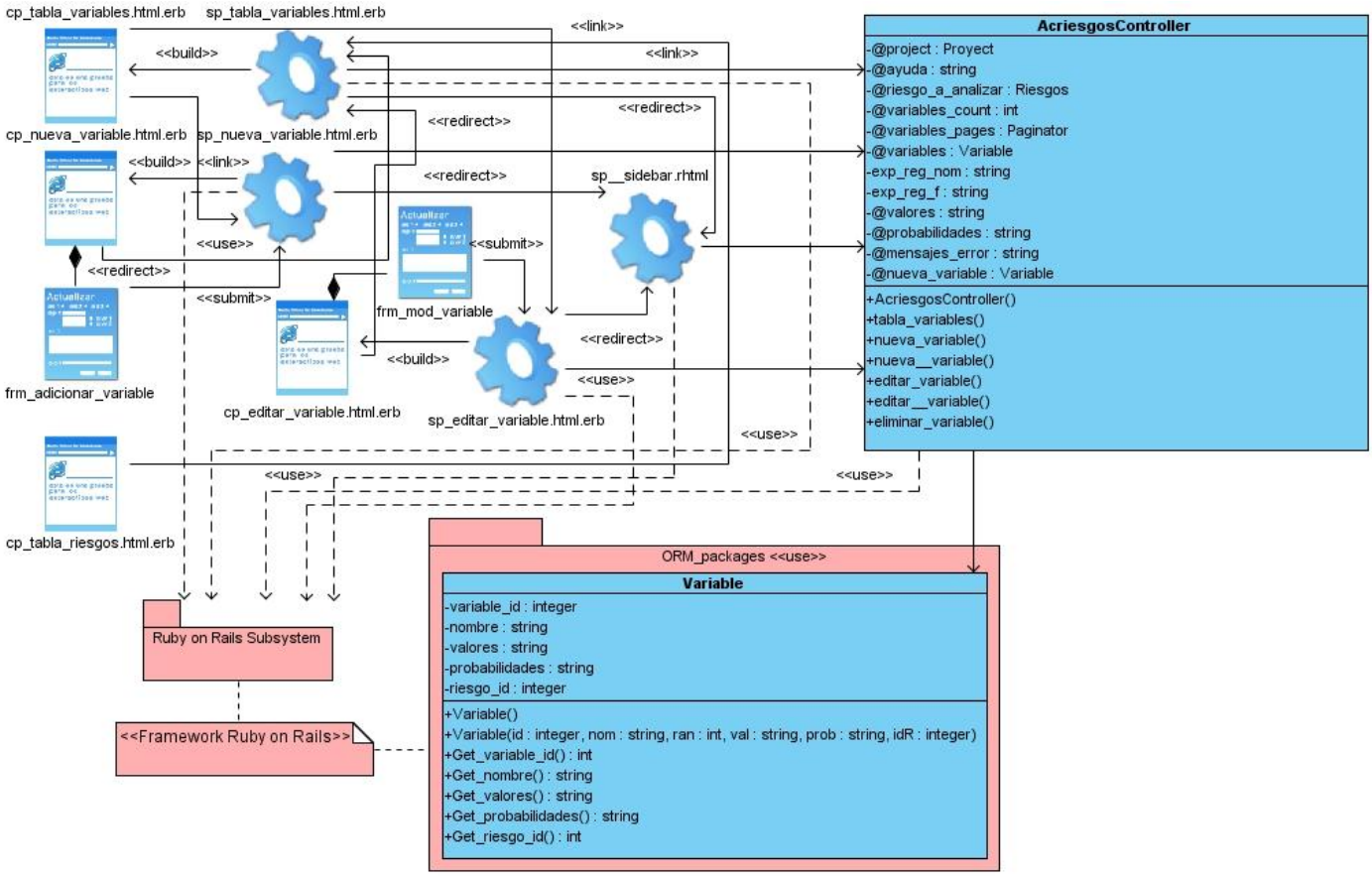


Ilustración 10: Diagrama de Clases del Diseño del CU Gestionar variable.

***

3.3.2 Diagrama de Clases del Diseño del CU Gestionar fórmula

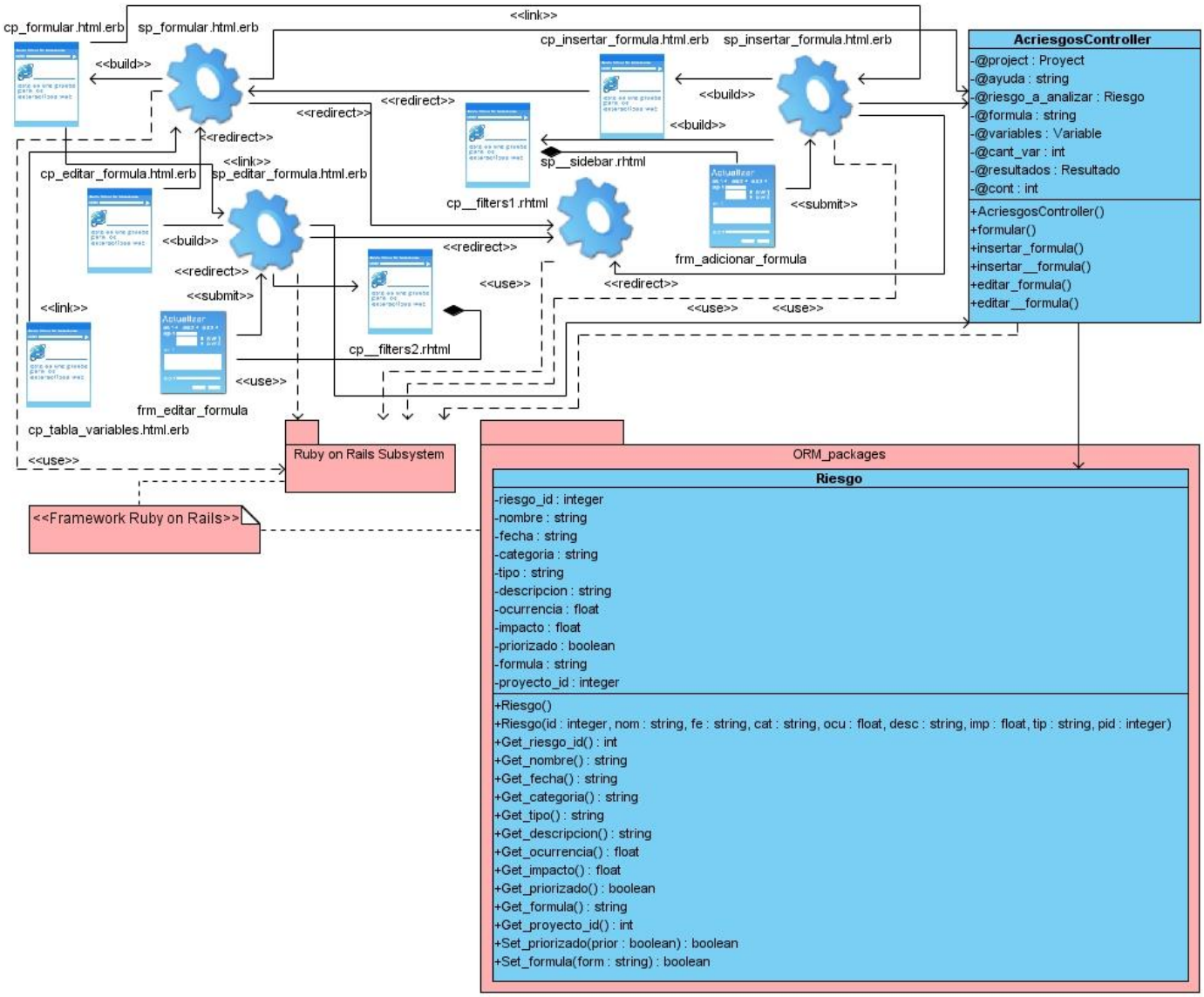


Ilustración 11: Diagrama de Clases del Diseño del CU Gestionar fórmula.

3.3.3 Diagrama de Clases del Diseño del CU Simular

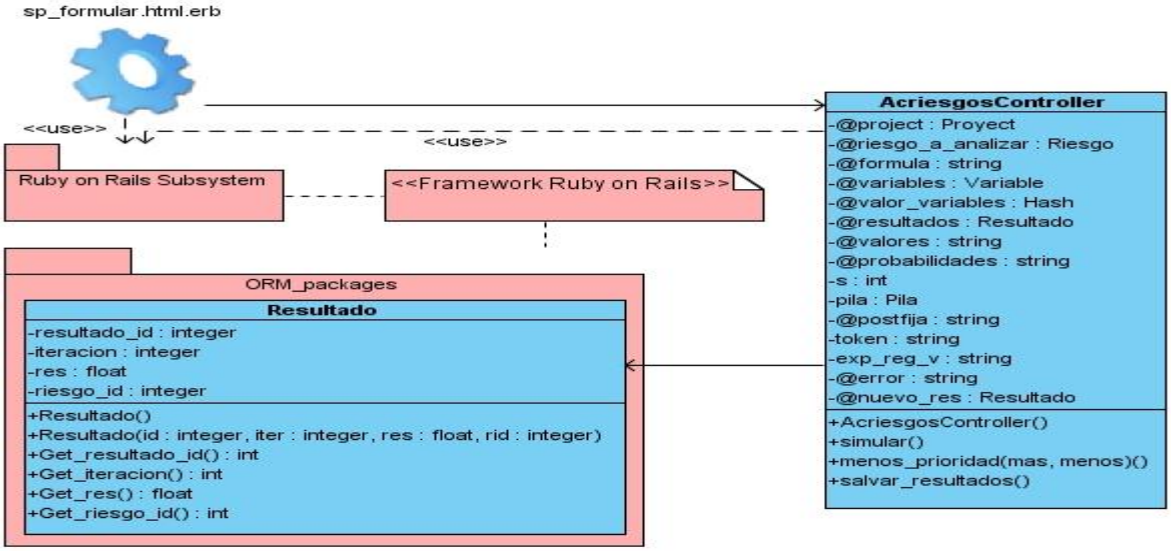


Ilustración 12: Diagrama de Clases del Diseño del CU Simular.

Los restantes diagramas de Clases del Diseño se encuentran en los anexos. (Ver **¡Error! No se encuentra el origen de la referencia.**)

3.4 Diagramas de Secuencia del Diseño

3.4.1 Diagrama de Secuencia del Diseño del CU Gestionar variable

- Escenario Principal:

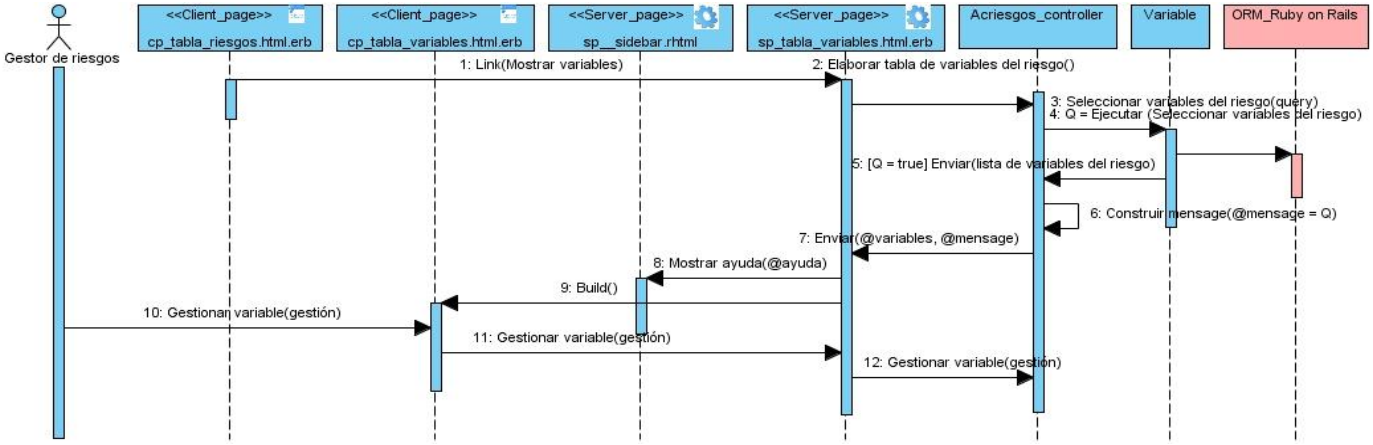


Ilustración 13: Diagrama de Secuencia del Diseño del CU Gestionar variable. Escenario Principal.

- Escenario Adicionar variable:

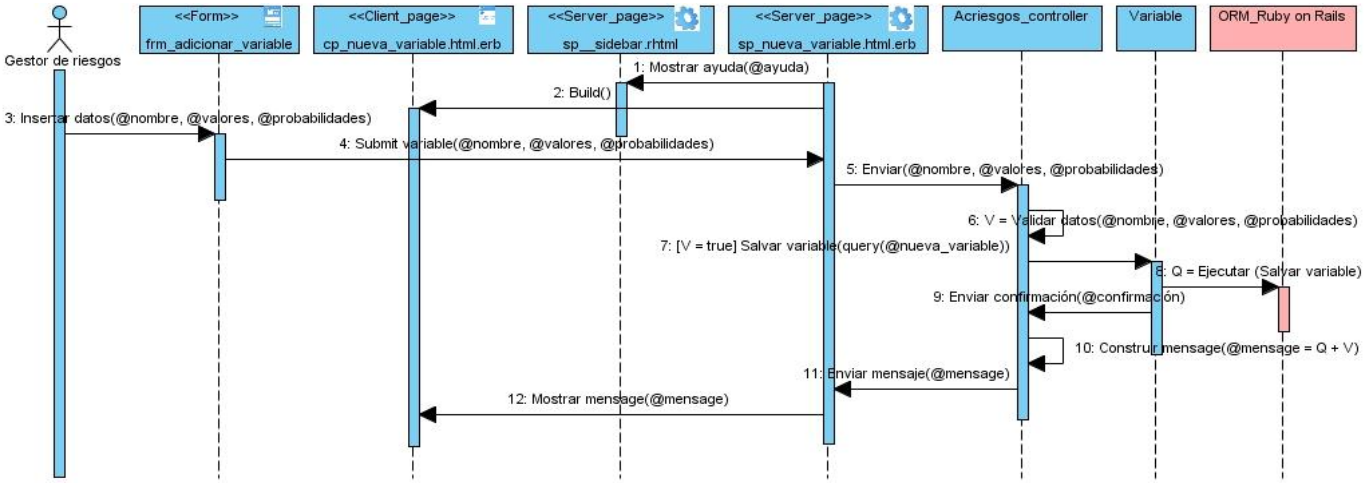


Ilustración 14: Diagrama de Secuencia del Diseño del CU Gestionar variable. Escenario Adicionar variable.

Escenario Modificar variable:

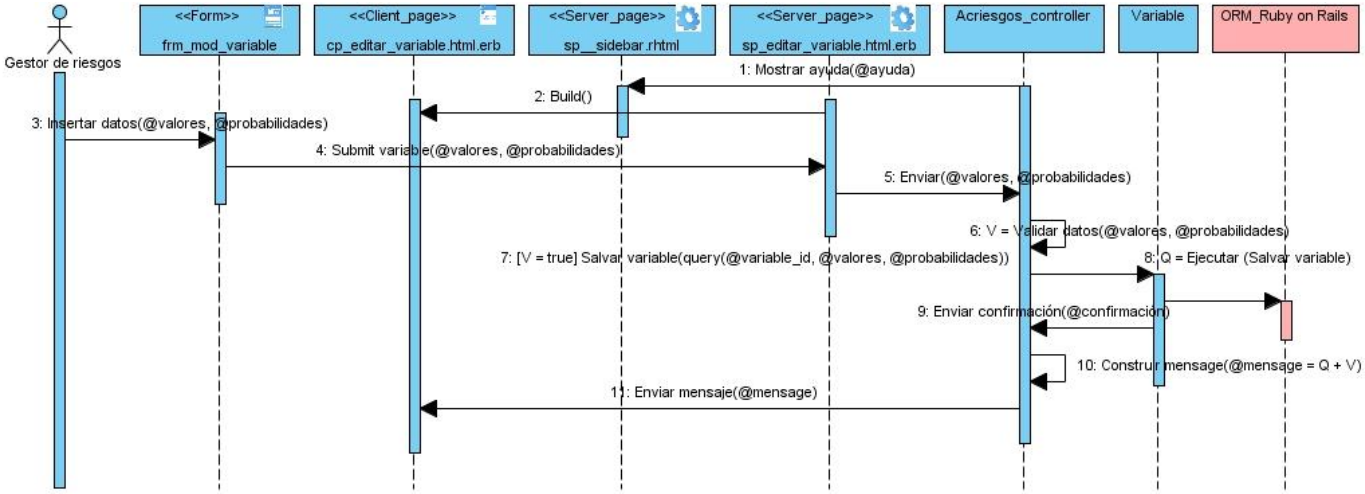


Ilustración 15: Diagrama de Secuencia del Diseño del CU Gestionar variable. Escenario Modificar variable.

3.4.3 Diagrama de Secuencia del Diseño del CU Simular

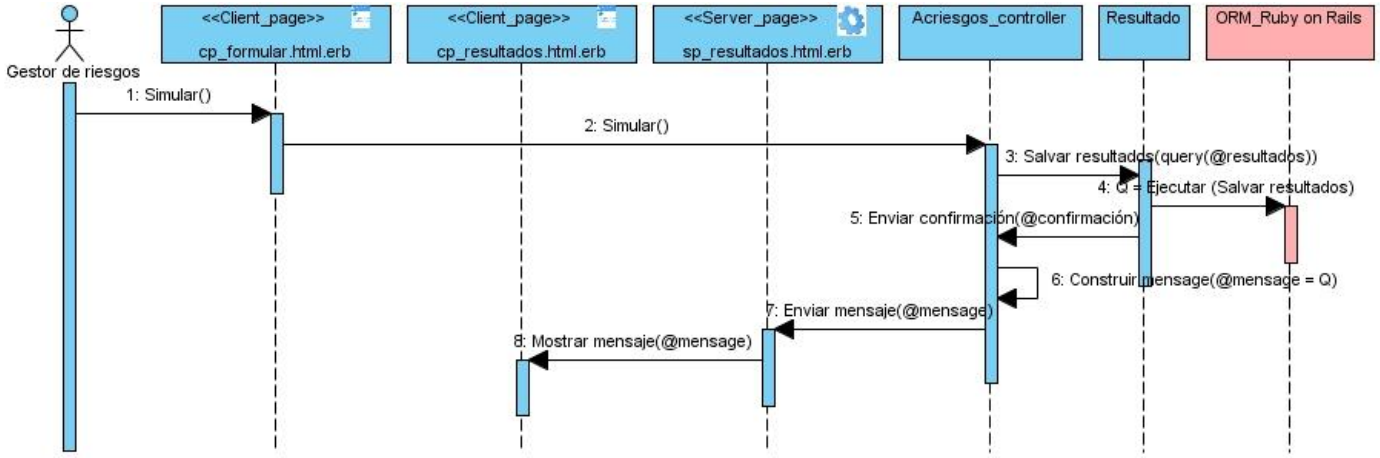


Ilustración 16: Diagrama de Secuencia del Diseño del CU Simular.

Los restantes diagramas de Secuencia del Diseño se encuentran en los anexos. (Ver **¡Error! No se encuentra el origen de la referencia.**)

3.5 Modelo físico de datos

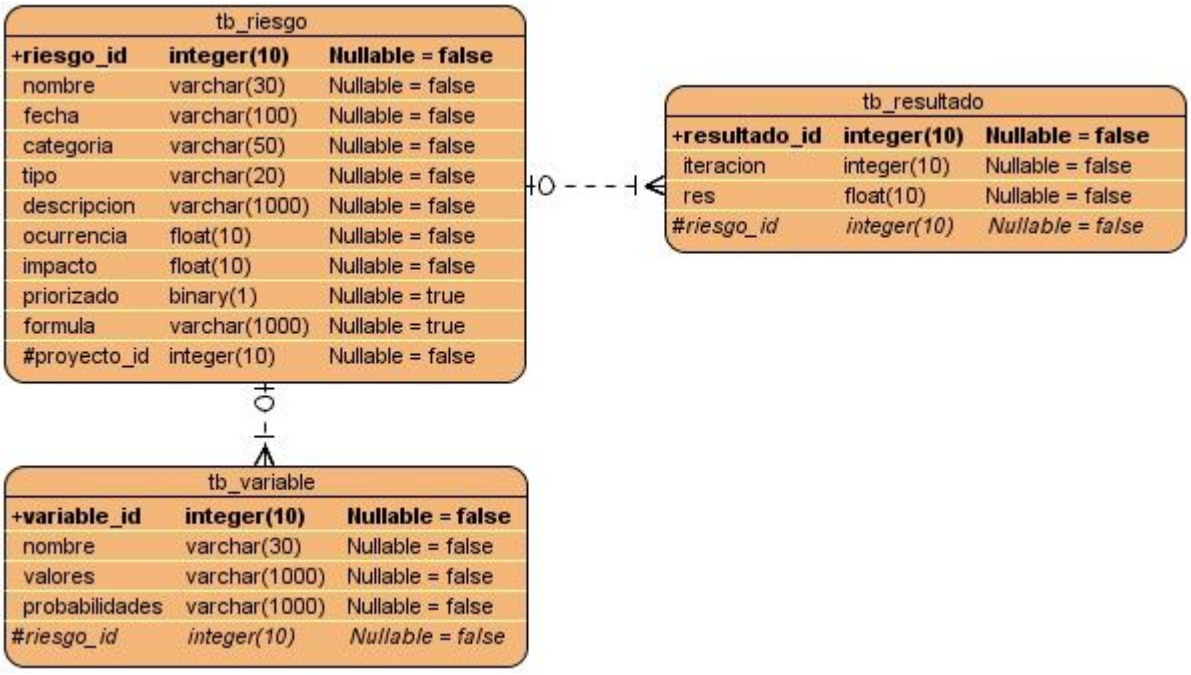


Ilustración 17: Modelo físico de datos.

3.6 Definiciones del diseño de la interfaz

El diseño de la interfaz de una aplicación, el tratamiento de los errores, así como el seguimiento de la seguridad de la misma tienen una gran importancia en el desarrollo y en el éxito de una herramienta. A continuación se describen los principios de diseño seguidos para el desarrollo de la aplicación en cuestión.

- Tratamiento de errores: El tratamiento de errores es de vital importancia en la aplicación, de él depende el buen funcionamiento de la misma. Evita la inserción de datos erróneos que conlleven a un mal funcionamiento. En este se aplica el trabajo con expresiones regulares a alto nivel, y se hace uso de funciones que provee el framework Ruby on Rails para emitir mensajes concisos y claros explicando en qué consiste el error. Ejecutándose todo lo mencionado a nivel del cliente.
- Seguridad: La seguridad es necesaria en todo tipo de aplicación. Se debe autenticar el usuario antes de que pueda realizar cualquier ejercicio sobre el sistema, y se utilizan mecanismos de encriptación de datos que por cuestiones de seguridad no deben viajar a través de la red en texto claro, como es el caso de los datos de la simulación.

- Interfaz: La Interfaz de Usuario (IU) de un programa es un conjunto de elementos hardware y software de una computadora que presentan información al usuario y le permiten interactuar con la información y con el computador. Si la IU está bien diseñada, el usuario hallará la respuesta que espera a su acción.

  Para alcanzar esto, en el desarrollo de la interfaz, se han tenido en cuenta los siguientes aspectos:
  - La herramienta será diseñada para una resolución de pantalla de 1024x768, la cual será la resolución óptima, pero además se ajusta en gran medida a la resolución a que esté configurada la computadora del cliente.
  - Se tiene en cuenta la uniformidad del sitio utilizando plantillas y páginas de estilo que permitirán presentar de la misma manera la presentación de los contenidos. El color predominante es tenue pues resulta neutral, agradable y refrescante a la vista. El idioma empleado es el español.
  - Se aprovechará y optimizará el espacio libre.
- Concepción de la ayuda: Se expondrá una ayuda especializada al usuario sobre el manejo de cada uno de los paneles funcionales del sistema, así como el funcionamiento del mismo como un todo.

3.7 Conclusiones del capítulo

En este capítulo se modelaron los diagramas de clases y de secuencia del diseño de la aplicación con el objetivo de transformar los requerimientos en elementos del sistema. Se analizaron los patrones más representativos utilizados así como la implementación del patrón arquitectónico MVC que realiza Ruby on Rails en el Redmine. Del mismo modo fueron definidas las relaciones existentes entre las entidades de la BD, quedando de esta manera todo listo para comenzar a implementar la aplicación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

4.1 Introducción del capítulo

El presente capítulo está orientado a dos vertientes fundamentales, la primera explicará cómo está concebido el sistema a través de los diagramas de componentes, y la segunda estará dedicada a detectar los posibles errores en el sistema mediante la realización de los casos de prueba. Se modela el diagrama de despliegue, donde se muestra la distribución física de los elementos de hardware que conformarán el sistema y las relaciones entre ellos; y el diagrama de componentes, donde se representa la organización y las dependencias lógicas que existen entre los componentes del software.

4.2 Implementación del Sistema

4.2.1 Diagrama de Despliegue

En un diagrama de despliegue se muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Los nodos no son más que elementos físicos que existen en tiempo de ejecución y representan un recurso computacional que generalmente tienen algo de memoria y capacidad de procesamiento. El diagrama de despliegue valdrá para modelar la topología de hardware sobre la cual se ejecutará el sistema.

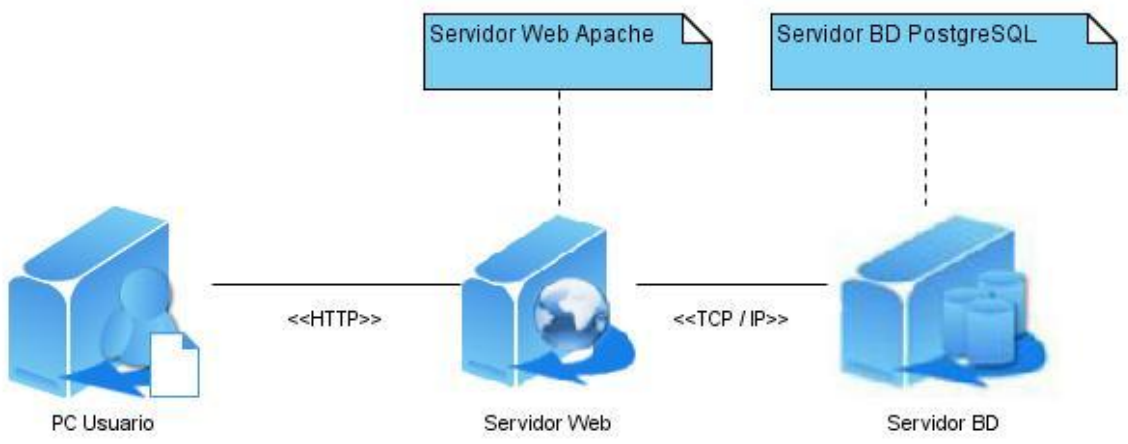


Ilustración 18: Diagrama de Despliegue.

La herramienta de gestión de proyectos Redmine 0.9.2 está instalada dentro de la UCI en un Servidor Apache, utilizando PostgreSQL como SGBD encargado de almacenar los valores con que trabajará la aplicación. Todas las PC clientes podrán acceder a dicho servidor mediante el protocolo de comunicación HTTP, garantizando que en cada estación de trabajo los usuarios tengan acceso a la herramienta.

4.2.2 Diagramas de Componentes

El Diagrama de Componentes muestra las dependencias lógicas entre componentes de software, sean estos componentes fuentes, binarios o ejecutables. Los componentes de software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución.

4.2.2.1 Diagrama de Componentes del CU Gestionar variable

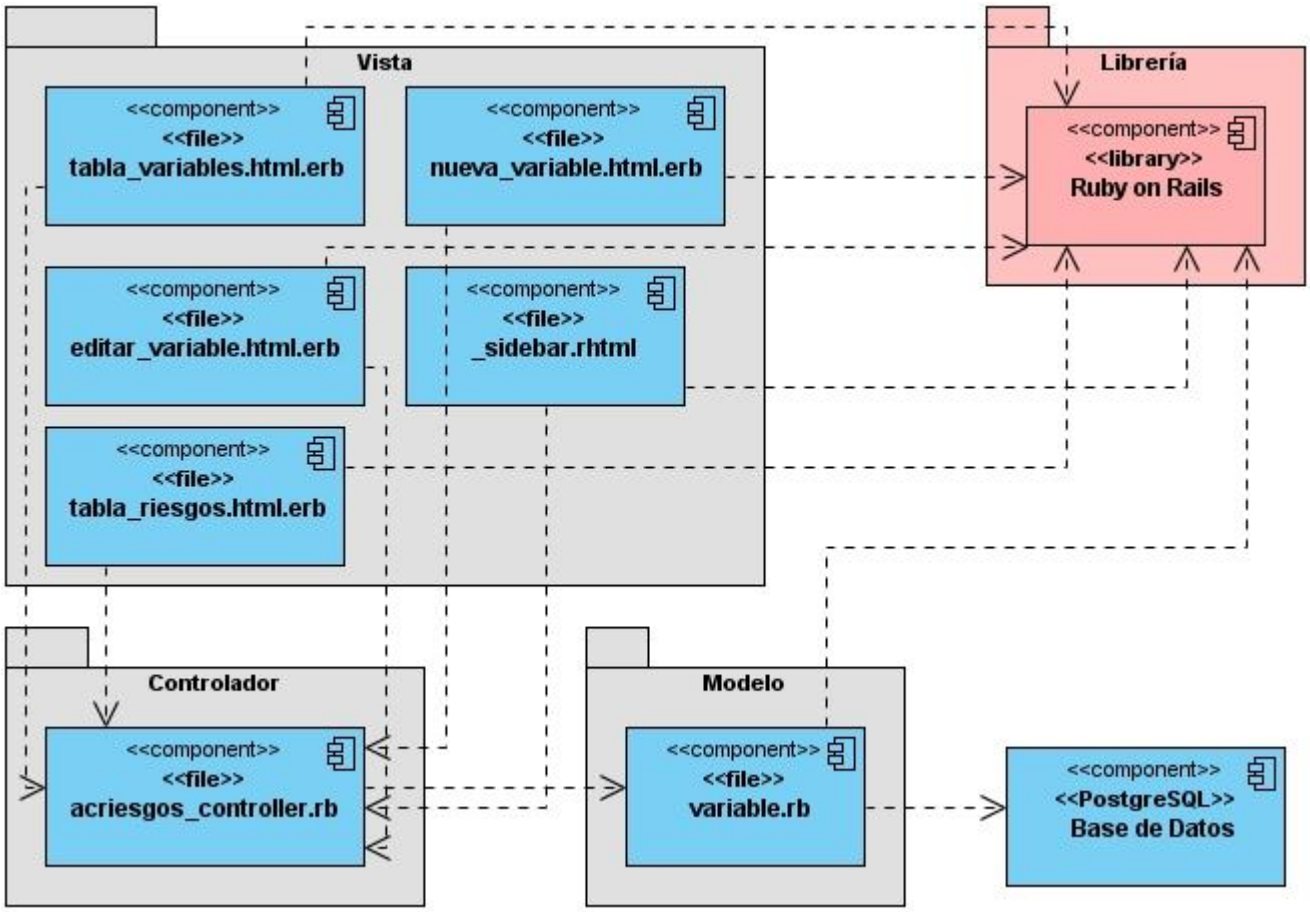


Ilustración 19: Diagrama de Componentes del CU Gestionar variable.

4.2.2.2 Diagrama de Componentes del CU Gestionar fórmula

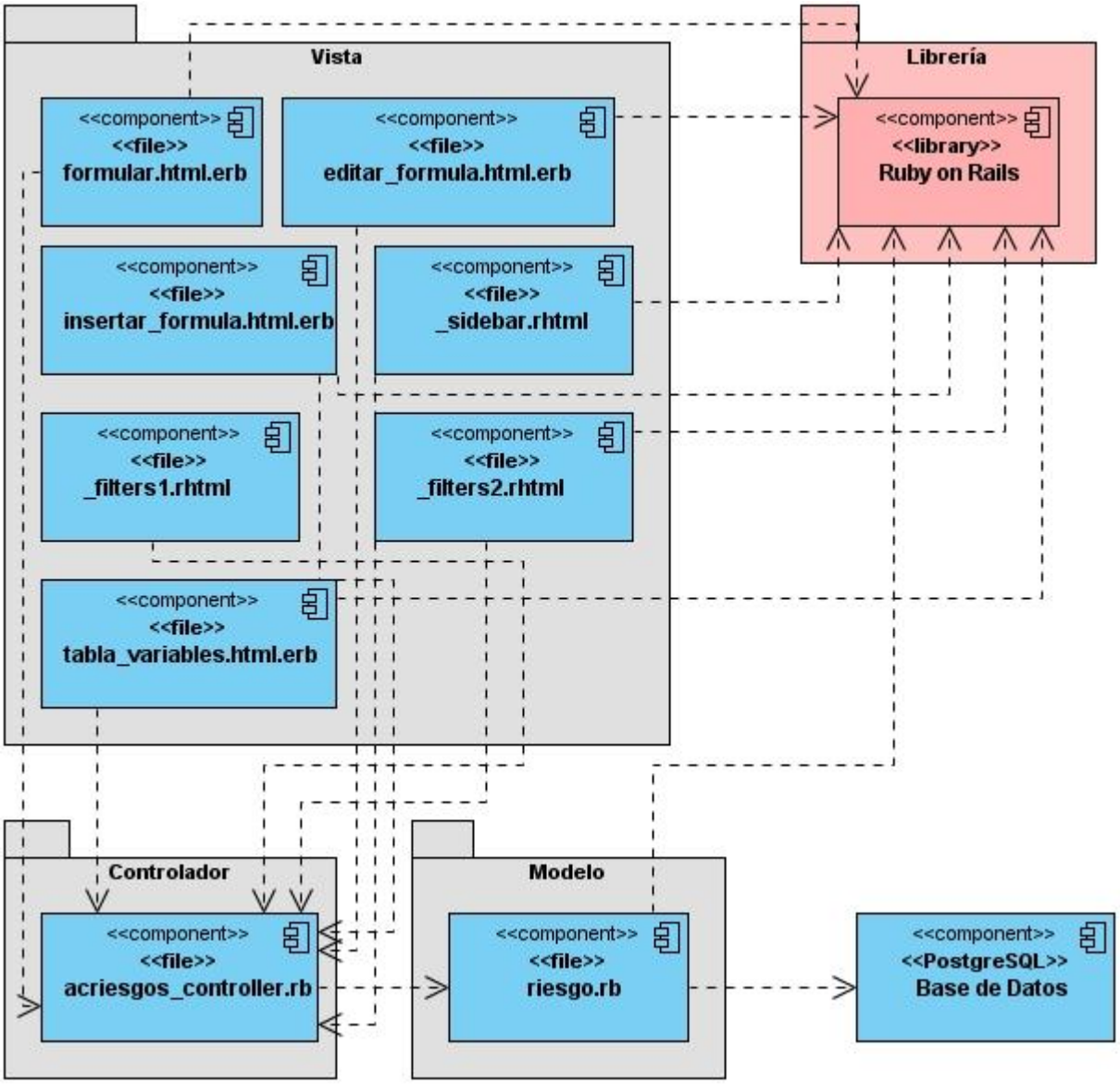


Ilustración 20: Diagrama de Componentes del CU Gestionar fórmula.

4.2.2.3 Diagrama de Componentes del CU Simular

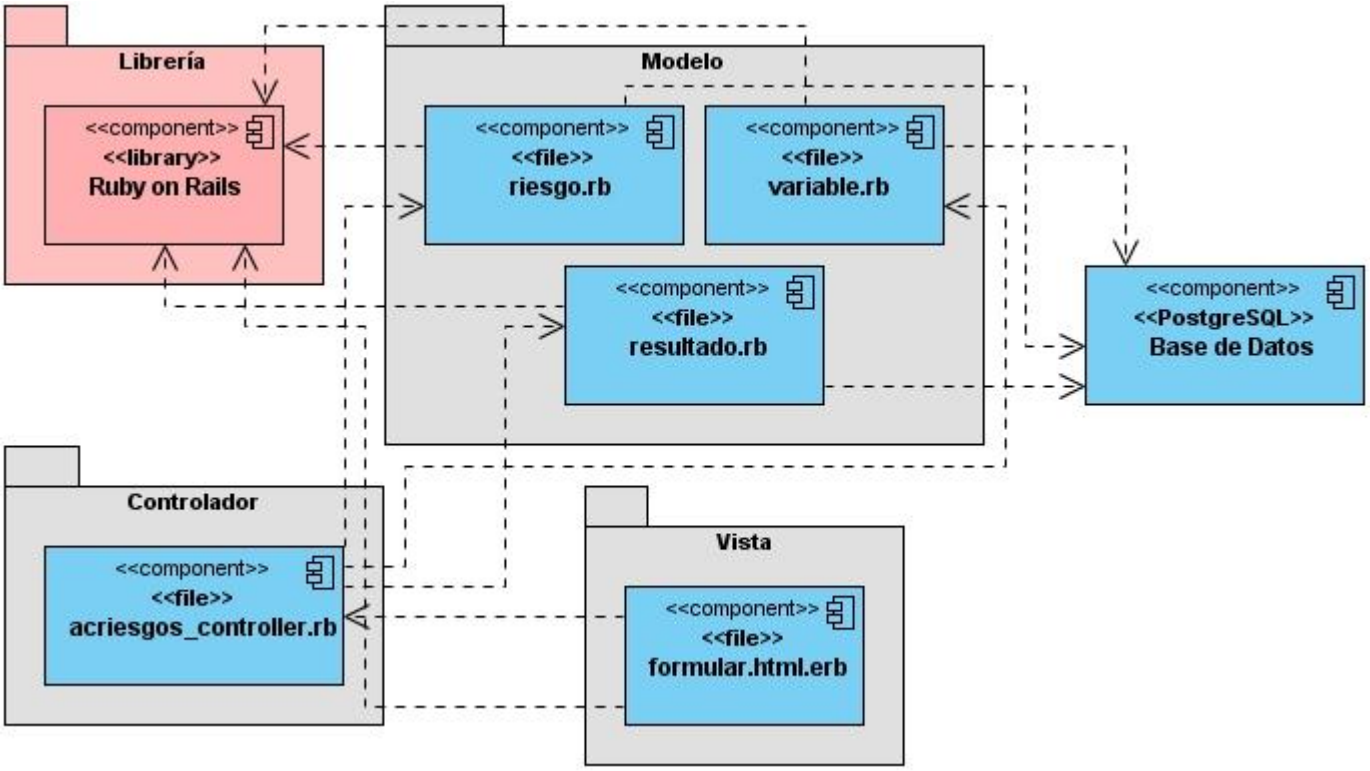


Ilustración 21: Diagrama de Componentes del CU Simular.

Los restantes diagramas de Componentes se encuentran en los anexos. (Ver ¡Error! No se encuentra el origen de la referencia.**)**

4.3 Pruebas del Sistema

El proceso de pruebas se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen Caso de Prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces.

La prueba es el proceso de ejercitar un programa bajo condiciones específicas cuyos resultados deben ser registrados y luego analizados. Es posible hacerlas a diferentes niveles, tales como unidad, integración, sistema y aceptación.

Cualquier producto de ingeniería puede ser probado mediante uno de estos métodos:

- Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa. **Pruebas de caja negra**.
- Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que todas las piezas encajen, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. **Pruebas de caja blanca**.

4.3.1 Plan de Pruebas

- **Nivel de Prueba. Sistema:** A este nivel es verificado que cada elemento encaje de forma adecuada, que sea alcanzado la funcionalidad y el rendimiento del sistema total, que sean validados los requisitos establecidos comparándolos con el sistema construido. El software debe integrarse con los componentes de hardware correspondientes respondiendo de manera funcional acorde a lo propuesto.
- **Tipo de Prueba. Funcionalidad:** Este tipo de prueba asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Verificando la apropiada aceptación de datos y se encuentra enfocada principalmente a los requisitos funcionales.
- **Método de Prueba. Caja negra:** Es denominada prueba de comportamiento, se centra en los requisitos funcionales del software, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa y está referida a las pruebas que se llevan a cabo sobre la interfaz del software examinando algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.
- **Técnica de Prueba. Particiones o Clases de equivalencia:** Divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Constituye una técnica algebraica que trata cada parámetro como un modelo donde unos datos son equivalentes a otros. Se realiza dividiendo un rango excesivamente amplio de posibles valores reales a un conjunto

reducido de clases de equivalencia, entonces es suficiente probar un caso de cada clase, pues los demás datos de la misma clase son equivalentes.

El diseño de casos de prueba según esta técnica consta de dos pasos:

1. Identificar las clases de equivalencia.
2. Identificar los casos de prueba.

- **Ambiente de Prueba:** Para la correcta realización de las pruebas, se contó con una serie de recursos que facilitaron el trabajo de ejecutar cada caso de prueba sobre la herramienta:
  - Recursos de hardware: Las computadoras utilizadas para el desarrollo de las pruebas contaron con microprocesador Intel Core2Duo E4500 con velocidad de 2.20 GHz, motherboard Intel, 512 MB de memoria RAM, capacidad en disco duro de 160 GB y red alámbrica.
  - Recursos de software: El SO en el cual se desarrollaron las pruebas fue Ubuntu 10.04.

4.3.2 Diseño de los Casos de Pruebas

Al sistema desarrollado se le realizaron pruebas de caja negra, las cuales se centran fundamentalmente en los requisitos funcionales del sistema (Ver Ilustración 22). Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

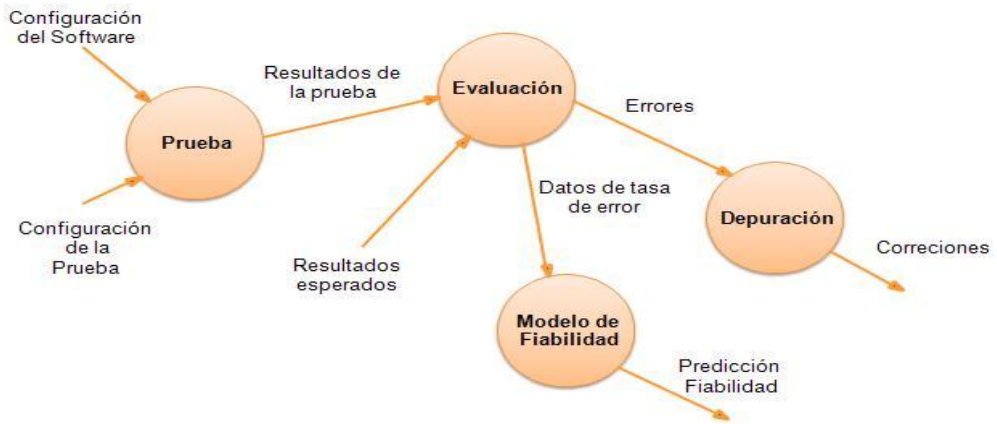
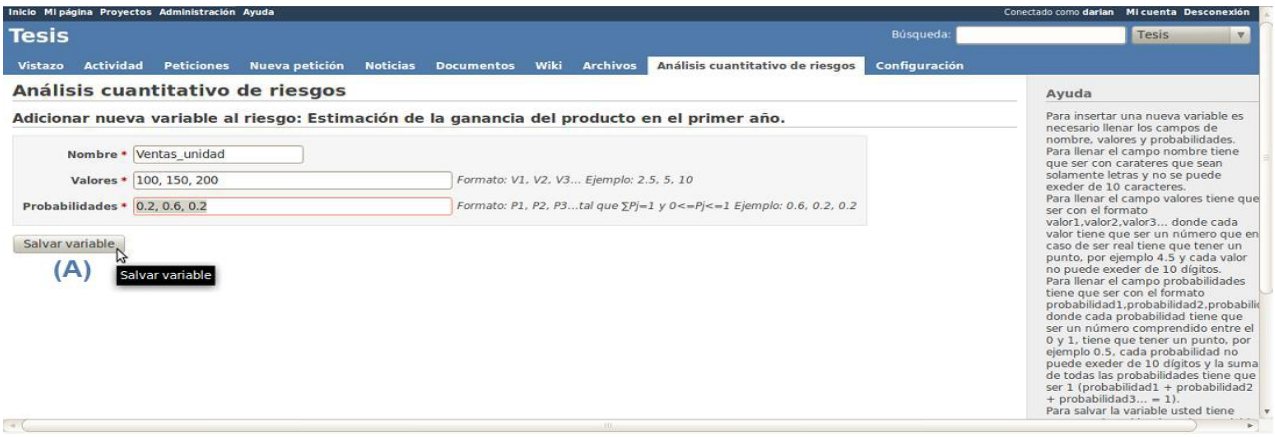


Ilustración 22: Modelo del proceso de Pruebas.

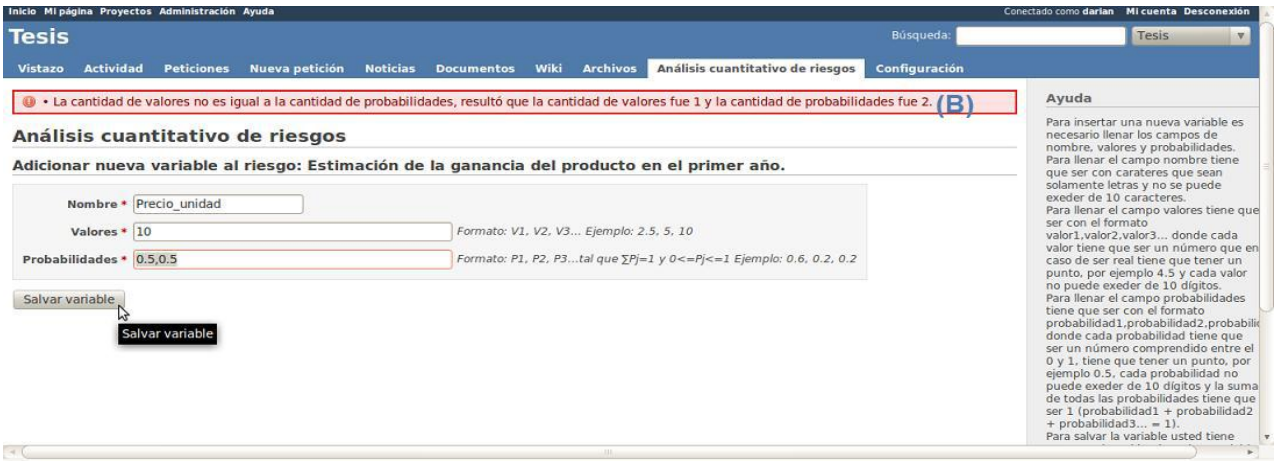
4.3.2.1 Caso de Pruebas Insertar variable

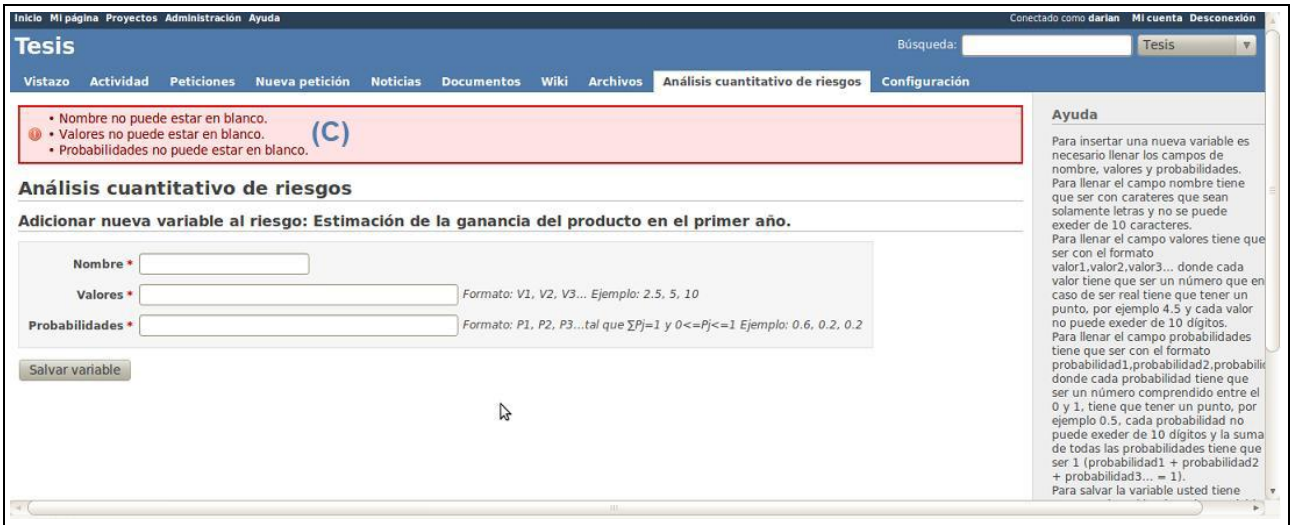
Condición de entrada:	Casos válidos:	Casos no válidos:

El usuario debe introducir el nombre, los valores que puede tomar la variable y las probabilidades de esos valores.	Que el usuario introduzca el nombre, los valores que puede tomar la variable y las probabilidades de esos valores correctamente.	Que el usuario no introduzca alguno de los parámetros o que los introduzca en un formato incorrecto.
Caso de uso:	Gestionar variable.	
Caso de prueba:	Insertar variable.	
Entradas:		
Nombre: Ventas_unidad Valores: 100, 150, 200 Probabilidades: 0.2, 0.6, 0.2		
Resultado:	El sistema inserta los datos de la variable en la BD. (A)	
Condiciones:	Los datos deben estar correctamente validados.	

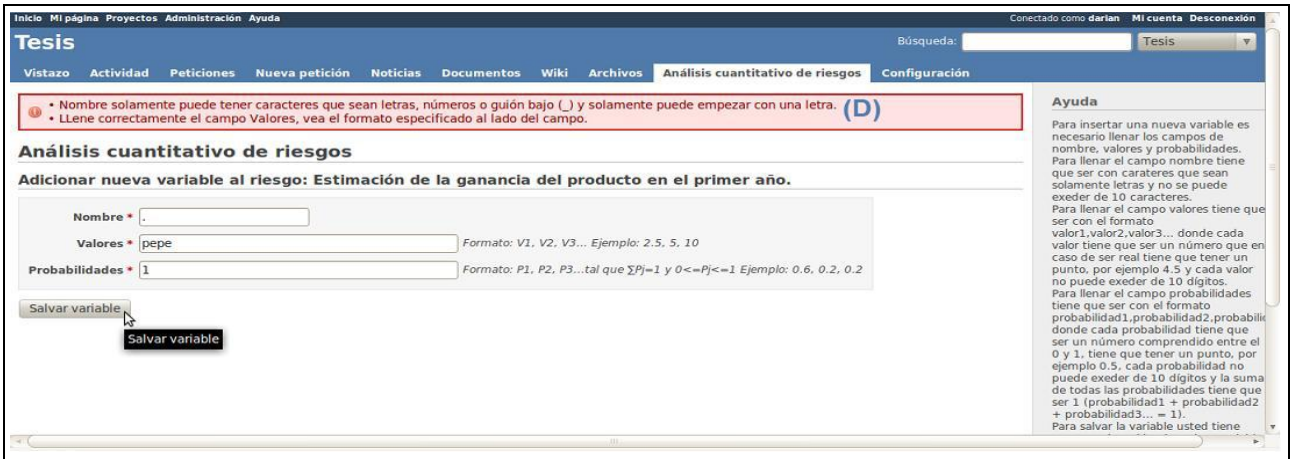


Caso de uso:	Gestionar variable.
Caso de prueba:	Insertar variable.
Entradas:	
Nombre: Precio_unidad	

Valores: 10 Probabilidades: 0.5,0.5	
Resultado:	El sistema muestra un mensaje de error. (B)
Condiciones:	El campo Probabilidades debe contener 1 valor ya que el campo Valores contiene 1 valor.
	
Caso de uso:	Gestionar variable.
Caso de prueba:	Insertar variable.
Entradas:	
Nombre: Valores: Probabilidades:	
Resultado:	El sistema muestra un mensaje de error. (C)
Condiciones:	El campo Nombre no puede estar en blanco, el campo Valores no puede estar en blanco, el campo Probabilidades no puede estar en blanco.



Caso de uso:	Gestionar variable.
Caso de prueba:	Insertar variable.
Entradas:	
Nombre: . Valores: pepe Probabilidades: 1	
Resultado:	El sistema muestra un mensaje de error. (D)
Condiciones:	El campo Nombre solamente puede tener caracteres que sean letras, números o guión bajo (_) y solamente puede empezar con una letra. El campo Valores no puede tener letras, solamente números separados por coma (,).



Caso de uso:	Gestionar variable.
Caso de prueba:	Insertar variable.
Entradas:	
Nombre: Costo_arreglos Valores: 10,20 Probabilidades: 0.5,0.6	
Resultado:	El sistema muestra un mensaje de error. (E)
Condiciones:	El campo Probabilidades la suma de todas las probabilidades tiene que ser igual a 1.

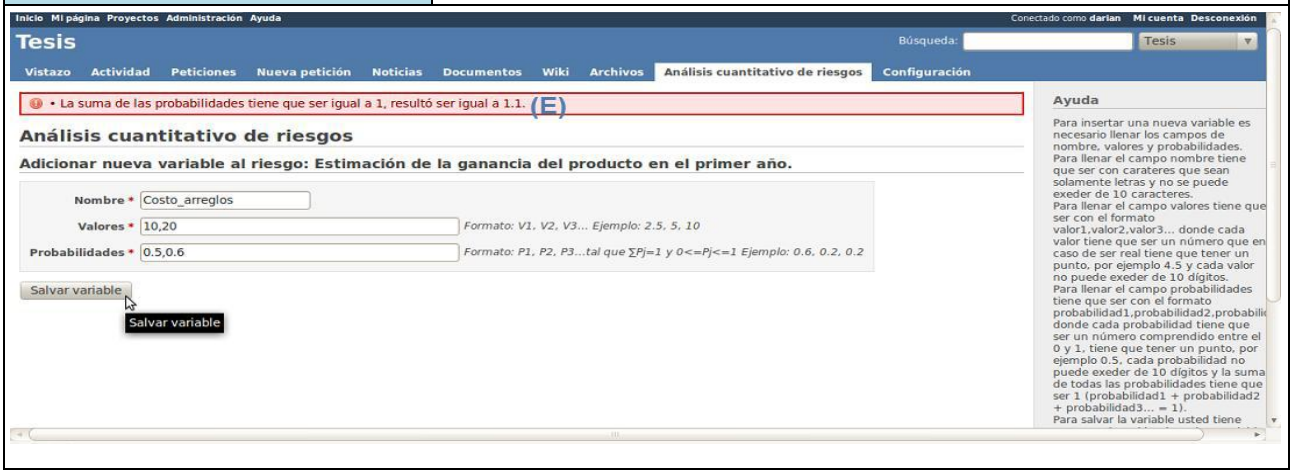


Tabla 5: Caso de Pruebas Insertar variable.

Los restantes casos de pruebas se encuentran en los anexos. (Ver **¡Error! No se encuentra el origen de la referencia.**)

4.3.3 Resumen del proceso de Pruebas

Para el correcto desarrollo del proceso de pruebas se definieron un total de 2 casos de pruebas, realizándose un total de 3 iteraciones, se utilizó el método de caja negra para probar las principales funcionalidades con las que debe cumplir la aplicación, detectándose un total de 6 no conformidades (Ver **¡Error! No se encuentra el origen de la referencia.**). Estos errores detectados tenían una repercusión negativa sobre el funcionamiento del Sistema y la BD; con la supresión de los mismos se alcanzó mejorar y optimizar gradualmente el funcionamiento de la aplicación.

4.4 Conclusiones del capítulo

En este capítulo, como parte del flujo de trabajo de implementación, se modelaron los diagramas de despliegue y de componentes. Por otra parte, se elaboraron y ejecutaron pruebas al sistema, arrojando resultados significativos para el perfeccionamiento del proceso de análisis cuantitativo de riesgos.

CONCLUSIONES GENERALES

A partir de la investigación realizada para el proceso de desarrollo de la herramienta se arribó a las siguientes conclusiones:

- Como respuesta al problema originado se obtuvo una aplicación web adjunta a la herramienta Redmine, respetando la interfaz y arquitectura de la misma, que incorpora el proceso de análisis cuantitativo de riesgos a la gestión de riesgos en los proyectos productivos de la UCI, dando cumplimiento al objetivo de la investigación, que servirá de apoyo en la toma de decisiones por parte de los líderes de proyectos frente a situaciones con incertidumbre.
- La obtención de una amplia documentación será útil para el desarrollo de futuras investigaciones sobre el análisis cuantitativo de riesgos.
- Las técnicas de modelado y simulación, haciendo uso de distribuciones de probabilidad, son las más idóneas para efectuar un óptimo proceso de análisis cuantitativo de riesgos.
- El valor social del sistema se expresa en la contribución a mejorar las condiciones de trabajo, desempeño y equidad de los especialistas en la UCI; pues su utilización reportará una mejora considerable en la calidad y eficiencia de los procesos que se automatizan.
- Un algoritmo de simulación de Montecarlo totalmente genérico, será útil al ser utilizado para otras funcionalidades dentro del Redmine.

RECOMENDACIONES

Al cumplir los objetivos propuestos en esta investigación, surgieron una serie de recomendaciones para futuras versiones:

- Hacer extensiva la aplicación adjunta al Redmine para todos los proyectos de la UCI.
- Incorporar variadas y específicas distribuciones de probabilidad para hacer más flexible y real la simulación del comportamiento de las variables con incertidumbre.
- Incorporar al proceso un análisis de sensibilidad a las variables de los riesgos.

REFERENCIAS BIBLIOGRÁFICAS

[1] SEI, S. I. (2004). Software Engineering Institute SEI.

[2] Díaz, C. (2004). Definiciones de riesgo. Consultado el 26 de Agosto de 2010. Disponible en: http://www.suratep.com.co/articulos/70/

[3] Connell, S. (2002). Gestión de Riesgos. Consultado el 15 de Septiembre de 2010. Disponible en: http://www.ldc.usb.ve/~teruel/ci4713/clases2002/riesgos.html

[4] Fernández, P., & Hernández, A. (2008). Guía para la Gestión de Riesgos a través de RUP.UCI. Ciudad Habana.

[5] Pressman. (2001). Ingeniería de Software. Un enfoque práctico. 5ta edición, McGraw-Hill. Vol. I.

[6] Charette, R. N. (1989). Software Engineering Risk Analysis and Management.

[7] Project Management Institute (PMI). (2004). Guía de los Fundamentos de Dirección de Proyectos. Norma Nacional Americana.

[8] Connell, S. (1997). Desarrollo y Gestión de Proyectos Informáticos. McGraw-Hill Iberoamericana.

[9] Project Management Institute (PMI). (2004). Guía de los Fundamentos de Dirección de Proyectos. Norma Nacional Americana.

[10] Fernández Fernández, Santiago; Cordero Sánchez, José María; Cordoba Largo, Alejandro. (2da edición revisada y actualizada. 2002). Estadística descriptiva. Consultado el 26 de Diciembre de 2010. Disponible en: http://books.google.com.cu/books?id=31d5cGxXUnEC&pg=PA17&dq=estadistica+descriptiva&hl=es&ei=eYgeTJuxD8T48AaxguCVDA&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCYQ6AEwAA#v=onepage&q=estadistica%20descriptiva&f=false

[11] Gómez Barrantes, Miguel. (1998). Elementos de estadística descriptiva. Consultado el 20 de Diciembre de 2010. Disponible en: http://books.google.com.cu/books?id=VJNpI4_U9SYC&printsec=frontcover&dq=estadistica+descriptiva&hl=es&ei=15AeTKXcEcP58AaMo4C_DA&sa=X&oi=book_result&ct=result&resnum=10&ved=0CFAQ6AEwCQ#v=onepage&q=estadistica%20descriptiva&f=false

[12] DDC Matemática Aplicada (UCI). Conferencia 3: Estadística Descriptiva. Distribuciones empíricas. Gráficos. Medidas estadísticas. Consultado el 26 de Diciembre de 2010. Disponible en: http://eva.uci.cu/file.php/69/Tema_2/Act.%207.pdf

[13] Pérez Amigo, Ernesto; Martínez Rodríguez, Yoandry. (2009). Sistema para el Análisis Cuantitativo de los Riesgos de los proyectos productivos del Centro de Desarrollo de Informática Industrial. UCI, Ciudad de la Habana. págs 37, 50, 51, 52.
[14] OpenUP Copyright. What is OpenUP? (2010). Consultado el 20 de Diciembre de 2010. Disponible en: http://epf.eclipse.org/wikis/openup/index.htm
[15] Nieto Doce, Yanet. (2008). Herramienta para la identificación de riesgos de proyectos productivos. UCI. Ciudad de la Habana. pág 31.
[16] Vidal, G. R. (2008). Gestión de la Seguridad Informática. Sistema para la Gestión de Riesgos.UCI, pág 23.
[17] Microsoft. Consultado el 10 de Noviembre de 2010. Disponible en: http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador
[18] Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objeto. S.l. Preason.

BIBLIOGRAFÍA

- Albert, C. J. (2006). Common Elements of Risk.
- Baldají, D. (2007). Propuesta de procedimiento para el desarrollo y aplicación de la Gestión del Riesgo en proyectos de producción de software. UCI. pág. 92.
- Connell, S. Desarrollo y Gestión de Proyectos Informáticos. Traducido por: Development, R. McGraw-Hill.
- Darkiewicz, G., Dhaene, J., & Goovaerts, M. (2004). Risk measures and dependencies of risks. Faculty of Economics and Applied Economics. K.U.Leuven.
- Díaz, C. (2004). Definiciones de riesgo. Consultado el 11 de Diciembre de 2009. Disponible en: http://www.suratep.com.co/articulos/70/
- Epstein, L. G. (2007). Living with Risk.
- Fernández, P., & Hernández, A. (2008). Guía para la Gestión de Riesgos a través de RUP.UCI. Ciudad Habana.
- Gallegos, J. (2006). Análisis del riesgo en la administración de proyectos de tecnología de información.

- García, J. (2005). Patrones de diseño, Diseño de Software Orientado a Objetos. Consultado el 5 de octubre del 2010. Disponible en http://www.ingenierosoftware.com/
- IEEE. Temas varios relacionados con Análisis de Riesgos de Software. Consultado el 25 de Septiembre del 2010. Disponible en: http://ieeexplore.ieee.org/search/searchresult.jspnewsearch=true&queryText=software+Risk+Analisis
- Ingersoll, K. Risk World Software: computer programs for risk assessment and risk management. Consultado el 6 de agosto del 2010, de Risk-Related Software. Disponible en: http://www.riskworld.com/SOFTWARE/SW5SW001.HTM
- OpenUP. Consultado el 26 de Septiembre del 2010. Disponible en: http://epf.eclipse.org/wikis/openup/
- Pandian, C. (2007). Applied Software Risk Management. A Guide for Software Projects Managers.Taylor & Francis Group.
- Pressman.R. (2005). Ingeniería de Software. Un enfoque práctico. Félix Varela. Parte 1 y Parte 2.
- Requerimientos Funcionales y No Funcionales. Consultado el 17 de octubre del 2010. Disponible en: http://www.mitecnologico.com/Main/RequerimientosFuncionalesYNoFuncionales
- SEI, S. I. (2004). Software Ingineering Institute SEI.
- Square, N. (2004). Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK) Tercera Edición, Project Management Institute. pág. 270-276. Pennsylvania 19073-3299 EE.UU.
- Technology, I. (2006). IT Risk Management.Trends through.
- Though, S. (2009). Active Risk Manager. Gerenciamiento de Riesgos. Consultado el 26 de Septiembre del 2010. Disponible en: http://www.overseasbr.com/es/riskmanagement/risksolutons/arm.asp
- University, X. (1992). The University Challenge: Problem-Solving Process User Manual. Stamford, Ct.: Xerox Corporation.
- Vidal, G. R. (2008). Gestión de la Seguridad Informática. Sistema para la Gestión de Riesgos.UCI. pág 23.
- Williams, R. C., Ambrose, K. & Bentrem, L. (2006). Risk Based Diagnostics.
- Williams, R. C. & Sandra, B. (1999). Software Risk Evaluation (SRE) Method Description, V 2.0. Carnegie Mellon University.