Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta para evaluar factibilidad económica de proyectos de software basada en técnicas de Soft Computing.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Orlando Guilarte Cutiño.

Ubesnel Rojas Orozco.

Tutores: MSc. Marieta Peña Abreu.

Ing. Yorgüy Antonio Batista Desdín.

La Habana, junio de 2015.

"Año 57 de la Revolución".

"El éxito es aprender a ir de fracaso en fracaso sin desesperarse."

Winston Churchill

Declaración de Autoría

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

Para que así conste firmo la presente a los __ días del mes de junio del año 2015.

Orlando Guilarte Cutiño

Firma del Autor

Ubesnel Rojas Orozco

Firma del Autor

MSc. Marieta Peña Abreu

Firma del Tutor

Ing. Yorgüy Antonio Batista Desdín

Firma del Tutor

Datos de contacto

Autor:

- **Nombre y apellidos:** Orlando Guilarte Cutiño.
- **Correo electrónico:** ogcutino@estudiantes.uci.cu
- **Institución:** Universidad de las Ciencias Informáticas.
- **Dirección:** Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Autor:

- **Nombre y apellidos:** Ubesnel Rojas Orozco.
- **Correo electrónico:** urojas@estudiantes.uci.cu
- **Institución:** Universidad de las Ciencias Informáticas.
- **Dirección:** Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Tutora:

- **Nombre y apellidos:** MSc. Marieta Peña Abreu.
- **Correo electrónico:** mpabreu@uci.cu
- **Situación Laboral:** Asesora de 5to año, Facultad 3.
- **Institución:** Universidad de las Ciencias Informáticas.
- **Años de Experiencia:** 5
- **Dirección:** Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

.

Tutor:

- **Nombre y apellidos:** Ing. Yorgüy Antonio Batista Desdín.
- **Correo electrónico:** yabatista@uci.cu
- **Situación laboral:** Profesor.
- **Institución:** Universidad de las Ciencias Informáticas.
- **Años de Experiencia:** 1
- **Dirección:** Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Dedicatoria

Quisiera dedicarle esta tesis a mi mamá, por ser el pilar fundamental de mi vida, por ser mi guía en toda la trayectoria de mi vida.

A Ernestina y Luis, por ser mis segundos padres y estar pendiente de mí, siempre guiándome por el buen camino.

A mis hermanos por siempre estar ahí cuando los necesite, por todo su apoyo durante la trayectoria de mi carrera.

A Mailín y Yarenis por ser dos hermanas más en mi vida, por ser mi ejemplo estudiantil a seguir.

A mis abuelos Luis, Ana María y Toña, por su amor y dedicación.

A mis sobrinos Jonathan e Isabela por ser parte de mi felicidad en los últimos años.

A mi novia Liannet por toda su paciencia, comprensión y apoyo durante estos últimos 3 años de la carrera.

De Orlando Guilarte Cutiño.

Quisiera dedicarle esta tesis a mi mamá por haberme traído al mundo y por guiarme siempre por el camino correcto en el poco tiempo que estuvo conmigo, estés donde estés siempre estarás en mi corazón mamita.

A mi papá por ser ejemplo de luchador inalcanzable y darme apoyo en todos estos años.

A mi tía Micel por ser esa madre que desde los 7 años no tuve, y hacer de mi lo que soy de hoy, un hombre de bien.

A mi hermana Yadira por darme siempre su amor y cariño, y estar siempre conmigo en las buenas y las malas.

A mi gran amor Elizabeth, por darme todo su amor y hacerme el hombre más feliz que existe en la tierra.

A todas las personas que de una forma u otra han contribuido en mi formación.

De Ubesnel Rojas Orozco.

Agradecimientos

En especial quiero agradecer a mi mamá por darme todo el amor del mundo, por todo su apoyo en los buenos y malos momentos, por ser esa persona que nunca se da por vencida, por ser la luz que guía mi camino y por hacerme de mí quien soy.

Gracias a mi papá por todo su apoyo.

A mis hermanos por todo su amor, su apoyo y por hacer de mí hoy una mejor persona.

A Ernestina y Luis por ser mis segundos padres, por acogerme como un hijo, por estar siempre pendiente de mis estudios, y por siempre aconsejarme hacia el buen camino.

A Yarenis y Mailín por ser mi modelo de estudiante a seguir, por su amor y por acogerme como un hermano más en la familia.

A mis sobrinos por brindarme tanto amor y momentos de felicidad en los últimos años.

A mis abuelos y abuelas, por su amor incondicional, y cuidarme desde donde están hoy.

A gollita por toda su preocupación, por su amor y por apoyarme en los momentos más difíciles de mi vida.

A mi padrastro Luis por su voluntad para que fuera una persona mejor, gracias a su apoyo he logrado una de las metas de mi vida.

A mis Tíos y Tías por su granito de arena, en especial a Jorge, Roberto, Rey, Dennis y Dionnis.

A Ubesnel, más que un compañero de tesis un gran amigo.

A Alexander Peña, Chiqui, Siuly, Yanais y Yorgüy por ser más que mis amigos mis hermanos, mi apoyo sobre todo en los momentos difíciles y sobre todo porque no tiene precio su amistad.

Gracias a mi novia Liannet por su apoyo y dedicación en los últimos años de mi carrera, por soportarme como soy y por hacerme ver el mundo de un punto de vista diferente.

A mis suegros por acogerme como un hijo más, por su amor y apoyo.

A todos mis compañeros de aula, en especial a Liset, Yadira, Dayani, Roberto, Joel y Gendry, que han demostrado su apoyo en todos los momentos.

A la profesora Dailin Benavides por todos sus consejos y apoyo en los momentos más difíciles en la universidad.

A todos los que de una forma u otra aportaron su granito de arena para que hoy fuera quien soy, Alberto, Aymé, Anidey, Yamilka, Charli, Eliza y Fidel.

A mi tutora Marieta por su esfuerzo, dedicación y paciencia.

De Orlando Guilarte Cutiño.

En especial quiero agradecer a mi madre porque a pesar de no tenerla junto a mí desde los 7 años, en todo este tiempo siempre ha estado conmigo guiándome por el camino correcto, gracias mamita estés donde estés, estoy seguro que estás muy orgullosa de tu niño.

Quiero agradecer a mi papá por ser ese hombre luchador de la vida, porque a pesar de perder a su mujer y quedarse con dos niños solo, supo enfrentar la vida, y nunca darse por vencido, y hacer de mi hermana y de mí las personas que somos hoy.

A mi tía Micel por ser mi segunda madre en la vida, gracias tía por estar siempre conmigo y darme tu amor como si fuera tu hijo.

A mi novia y dentro de poco esposa Eliza por darme todo el amor del mundo, apoyarme siempre, por confiar siempre en mí. Te amo mi amor.

A la familia de mi novia por acogerme con tanto cariño…los quiero…

A mi hermana Yadira por ser una hermana maravillosa, y brindarme todo su apoyo y cariño y darme las sobrinas más lindas del mundo.

A mi cuñado por darme su apoyo y por querer y hacer feliz a mi hermana.

A mis abuelos y abuelas por ese ejemplo de tenacidad y fortaleza que los caracteriza.

A mis tías y tíos por dar su granito de arena.

A todos mis primos y primas por apoyarme siempre y quererme mucho.

A mis tutores por su dedicación y apoyo durante el desarrollo de la tesis.

A mis compañeros de aula por compartir estos años de estudio con ellos, en especial a mi compañero de tesis Orlando por sus horas y dedicación incondicional en el desarrollo de la tesis.

A mis compañeros del equipo de beisbol de la Facultad 3.

A mis compañeros Alumnos Ayudantes.

A mis estudiantes de 7mo grado en la Secundaria Básica 14 de Junio del municipio La Lisa.

A todos los que de una forma u otra permitieron que hoy estuviera escribiendo estas letras.

De Ubesnel Rojas Orozco.

Resumen

Debido a la necesidad de una distribución óptima de los recursos con los que cuenta la Universidad de las Ciencias Informáticas para desarrollar los proyectos de software y a las deficiencias que existen en el proceso de contratación de proyectos, resulta necesario realizar estudios de factibilidad de cada uno de ellos. La realización de estos estudios permitirá determinar las probabilidades de éxito de los proyectos de software, facilitando mayor información al decidir su aceptación y garantizando un mejor uso de los recursos disponibles de acuerdo a las necesidades y características que presente la institución en dicho momento. El presente trabajo tiene como objetivo desarrollar una herramienta informática para el análisis de la factibilidad económica de proyectos de software basada en técnicas de Soft Computing tolerando la incertidumbre presente en la información. Dicha herramienta contribuirá a un mejor desarrollo de software en la universidad, facilitando la realización de estudios de factibilidad económica en entornos de incertidumbre presente en la información, sirviendo de apoyo a la toma de decisiones de los principales gerentes y niveles de dirección de los centros productivos.

Palabras claves: factibilidad económica, Soft Computing.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Introducción

Los estudios de factibilidad constituyen la base para el desarrollo de cualquier proyecto. Siendo una guía que permite establecer el enfoque, alcance y los diferentes aspectos que se deben considerar para efectuar el análisis y evaluación económica del mismo. Estos estudios recopilan información, sirviendo de base para la toma de decisiones referentes a su continuidad o no. Para realizar la ejecución, es necesario analizar los factores económicos que intervienen en él para alcanzar los resultados esperados. Se analiza la proporción existente entre costo y beneficio, se comparan los costos y beneficios esperados para asegurarse de que los beneficios sean mayores que los costos.

Con el avance de las Tecnologías de la Información y las Comunicaciones ha evolucionado la gestión de proyectos. En el mercado cada día aumenta la competencia en la comercialización, lo que exige una mayor calidad de los procesos de producción y el ahorro de los recursos para evitar el fracaso de los proyectos (Lamadrid, 2012). Según el Chaos Report de Standish Group, los proyectos que alcanzan sus objetivos y que podrían ser catalogados como exitosos son minoría. Las causas de que un proyecto no se culmine con éxito suelen ser múltiples e interrelacionadas, algunas causas son: falta de planificación, mala gestión de los recursos humanos, aumento de los costos, mala gestión de los riesgos, cambios frecuentes en los requerimientos y especificaciones (Proyectos, 2014). El análisis económico pretende determinar cuál es el costo de los recursos económicos necesarios para la realización del proyecto (Herrera, 2015). La permanente búsqueda en reducción de costos ha llevado a las empresas a generar proyectos de alta criticidad y exigencia con el fin de que los mismos aporten algún beneficio tangible a la organización que necesita aumentar ingresos y bajar costos (Proyectos, 2013).

Para determinar la factibilidad económica de un proyecto es necesario utilizar los diferentes métodos económicos existentes; estos permiten determinar cómo responderá la economía en un escenario determinado. "Un método económico es una teoría la cual representa un proceso económico bajo determinadas variables y estableciendo una secuencia lógica" (Económicos, 2015). Los métodos económicos tradicionales más conocidos en la actualidad son: flujo de caja, valor actual neto, período de recuperación y tasa interna de retorno (Abreu, 2012).

Los métodos económicos tradicionales son fáciles de calcular y aplicar, presentando un conocimiento preciso, completo y exacto. "El Soft Computing es una rama de la inteligencia artificial centrada en el diseño de sistemas inteligentes capaces de manejar adecuadamente la información incierta, imprecisa y/o incompleta" (Computing, 2015). Las técnicas que componen el Soft Computing son: la neurocomputación, el razonamiento probabilístico y la lógica difusa (Zadeh, 1996).

La lógica difusa ha tomado gran auge a nivel mundial, siendo una de las más utilizadas en la toma de decisiones. "Es aconsejable utilizarla para resolver procesos muy complejos, es decir, cuando se carece de un modelo matemático simple o para procesos no lineales. De esta forma las características más

importantes de la lógica difusa son: la flexibilidad, la tolerancia con la incertidumbre, la capacidad para moldear problemas no lineales y su fundamento en el lenguaje de sentido común" (Difusa, 2015).

"La combinación de los métodos económicos tradicionales con la lógica difusa permite obtener resultados con una mayor calidad y disminución de las tasas de errores provocados por los humanos" (Abreu, 2012). Es recomendado utilizarla cuando el volumen de datos es muy elevado, donde el razonamiento humano puede llevar a conclusiones erróneas en entornos de incertidumbre.

Luego de entrevistar a varios jefes de proyectos, especialistas y económica de la dirección de comercialización y negocios de la universidad, para conocer cómo se realizan los estudios de factibilidad económica de proyectos de software se encontraron las siguientes deficiencias:

1. El número de proyectos que se desarrollan es elevado, existiendo en ocasiones déficit de personal calificado en temas de factibilidad económica para trabajar en un determinado proyecto, así como de recursos materiales.
2. Se decide la ejecución de algunos proyectos sin un estudio de factibilidad económica profundo.
3. Ausencia de una herramienta que informatice y agilice el proceso de análisis de factibilidad económica y que tenga en cuenta los errores cometidos por los humanos.
4. Los jefes de proyecto carecen de conocimientos de la existencia de modelos y herramientas para el estudio de factibilidad, trayendo consigo que temas como la factibilidad económica no sean tratados adecuadamente.
5. Los análisis de factibilidad en la universidad, algunos se realizan a través del GESPRO, quien solamente trabaja los métodos económicos de forma tradicional.

Por tal motivo surge el interés de establecer un conjunto de métodos que permitan evaluar la factibilidad económica adaptable a las personas que toman decisiones de la ejecución de un proyecto o no, apoyados por una herramienta informática. Actualmente existen varias herramientas para evaluar la factibilidad de proyectos de software pero son privativas y sus costos dificultan su adquisición.

Por lo anteriormente expuesto se plantea el siguiente **problema a resolver:** ¿Cómo contribuir a determinar la factibilidad económica de proyectos de software tolerando la incertidumbre presente en la información?

Basado en el problema planteado se determina como **objetivo general** desarrollar una herramienta informática para el análisis de la factibilidad económica de proyectos de software mediante el uso de técnicas de Soft Computing tolerando la incertidumbre presente en la información.

Con la finalidad de dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos:**

1. Construir el marco teórico conceptual para fundamentar la investigación sobre la factibilidad económica de proyectos de software.
2. Realizar el análisis y diseño de la solución propuesta, mediante la metodología de desarrollo de software seleccionada.
3. Implementar todas las funcionalidades definidas para obtener la herramienta, mediante las herramientas y tecnologías seleccionadas.
4. Validar la herramienta propuesta a partir de la confección de estudios de casos, pruebas de caja blanca, caja negra y de aceptación.

Para el desarrollo del presente trabajo se parte de la **idea a defender:** el desarrollo de una herramienta para evaluar proyectos de software basada en técnicas de Soft Computing, permitirá evaluar la factibilidad económica de proyectos de software tolerando la incertidumbre presente en la información.

Los siguientes métodos teóricos sustentan el trabajo de investigación:

Histórico lógico: se realizó un análisis de la problemática enunciada, revisando los beneficios y deficiencias de las herramientas para la evaluación de factibilidad proyectos de software.

Sistémico: se plantea el problema y su solución como un todo, realizando un estudio de cada uno de los componentes de la evaluación de proyectos de software, estableciendo dependencias y conexiones entre cada una de las fases para lograr un resultado integral e instaurar así una herramienta sostenible.

Hipotético deductivo: se realiza un análisis hipotético deductivo ya que a partir de la problemática existente hoy en la UCI[1] se plantean los objetivos generales y específicos, proponiendo la idea a defender.

Inductivo deductivo: se realiza un análisis inductivo deductivo ya que a partir de numerosos casos particulares por inducción, se pueden confirmar teorías y de estas teorías se logran deducir conclusiones sobre casos particulares que son verificadas posteriormente en la práctica.

Modelación: con el objetivo de realizar abstracciones para dar una explicación de la realidad existente. Su condición principal es la relación entre el modelo y el objeto que se modela, que en este caso sería la herramienta en cuestión. Se evidencia el uso de este método en el modelo de dominio realizado, diagrama de despliegue y modelo de datos.

[1] **UCI:** Universidad de las Ciencias Informáticas.

Métodos empíricos:

Entrevista: en la primera parte de la investigación se realizaron varias entrevistas para determinar cómo se realiza la evaluación de proyectos en la universidad y la existencia de alguna herramienta que informatice este proceso.

La presente investigación está estructurada en tres capítulos:

- **Capítulo 1: Fundamentos teóricos.**

Se hace referencia a los principales conceptos para la investigación. Se realiza un análisis del estado del arte. Se realiza un estudio de las técnicas de Soft Computing, los métodos económicos y las fases de la ingeniería de requisitos. Se fundamentan las herramientas, metodología y tecnologías para el desarrollo de la solución propuesta. Se realiza un estudio de los patrones de diseño, patrón arquitectónico, métricas para validar el diseño y las pruebas de caja blanca, caja negra y pruebas de aceptación.

- **Capítulo 2: Análisis y diseño de la solución propuesta.**

En este capítulo se muestra el modelo de dominio. Se exponen los diferentes artefactos generados en las fases de planificación y diseño de la metodología de desarrollo de software seleccionada. Se identifican los patrones arquitectónicos y de diseño a utilizar para el desarrollo de la herramienta. Se muestran los resultados de validar el diseño mediante las métricas Tamaño Operacional de Clases y Relaciones entre Clases.

- **Capítulo 3: Implementación y pruebas.**

En este capítulo se exponen los diferentes artefactos generados durante la fase de implementación, así como los estándares de codificación y el tratamiento de la seguridad en la herramienta. Se realizan las pruebas unitarias, de aceptación y de sistema para validar funcionalmente la propuesta de solución. Se valida que la propuesta de solución cumple con el objetivo planteado.

Capítulo 1: Fundamentos teóricos

1.1 Introducción

En este capítulo se abordan los principales conceptos utilizados en la investigación. Además se analizan los diferentes modelos y herramientas existentes en el ámbito nacional e internacional para el estudio de factibilidad. Se analizan las diferentes técnicas de Soft Computing y métodos económicos. Se fundamenta la metodología de desarrollo de software seleccionada. Se realiza una breve caracterización de las herramientas utilizadas para la solución del problema planteado. Se describen las fases de la ingeniería de requisitos, las métricas para la validación del diseño y las pruebas de caja blanca, caja negra y pruebas de aceptación.

1.2 Principales conceptos

Para facilitar la comprensión de la investigación se deben entender primeramente los términos que se emplean durante el desarrollo de la misma.

Proyecto: el PMBOK[2] define un proyecto como un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. El término temporal se refiere a que cada proyecto define un comienzo y un final al cual se llega cuando se han cumplido todos los objetivos del proyecto o cuando queda claro que no se han alcanzado o no se alcanzarán los objetivos. Un proyecto crea producto, servicio o resultado único mediante una elaboración gradual, lo que significa que se va desarrollando en pasos el proyecto y este va aumentando cada vez (PMI, 2008).

Evaluación de proyectos: proceso por el cual se determina el establecimiento de cambios generados por un proyecto a partir de la comparación entre el estado actual y el estado previsto en su planificación (Serrano, 1999).

Factibilidad económica: los objetivos de esta área son ordenar y sistematizar la información de carácter monetario que proporcionan los análisis técnicos y comerciales, elaborar cuadros analíticos y antecedentes adicionales para valorar su rentabilidad (Acevedo, 2010).

Soft Computing: especie de consorcio o sociedad entre la lógica difusa, la neurocomputación y el razonamiento probabilístico, en el que la última incluiría los algoritmos genéticos, el razonamiento evidencial y los sistemas caóticos (Zadeh, 1996).

Lógica difusa: constituye un sistema lógico que está dedicado a la formalización de modos de razonamiento que son aproximados y no exactos (Zadeh, 1996).

[2] **PMBOK:** Acrónimo en inglés de Project Management Body of Knowledge.

1.3 Modelos para la evaluación de proyectos de software

A partir de la problemática planteada se realizó un estudio de los diferentes modelos existentes a nivel nacional e internacional relacionados con el tema de la investigación. Varios autores proponen distintos modelos para definir y documentar la evaluación de proyectos de software, a continuación se muestran los modelos estudiados:

1. Metodología de preparación y evaluación de proyectos (Gónzalez, 2002).
2. Propuesta de síntesis para la evaluación y selección de proyectos de nuevos productos (Vázquez, 2010).
3. Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones (Tamayo, 2010).
4. Método de evaluación de proyectos para decidir su aceptación (Castro, 2010).
5. Modelo para análisis de factibilidad en la evaluación de proyectos de software (Abreu, 2012).

Estos modelos van dirigidos a desarrollar métodos y procedimientos para evaluar económicamente proyectos, facilitando así la toma de decisiones y la posibilidad del éxito de los proyectos evaluados. Luego de analizados los modelos estudiados se determinó que presentan un conjunto de deficiencias. En su mayoría son deterministas debido a que no tratan la incertidumbre presente en la información, trabajan con números exactos y no combinan los resultados de los análisis económicos, técnicos y comerciales. El modelo propuesto por (Abreu, 2012) recibe como entrada al sistema de inferencia borroso, los resultados de los métodos económicos tradicionales, siendo números exactos y dejando márgenes de errores provocados por los humanos. De ahí la necesidad de desarrollar una herramienta que permita evaluar proyectos de software teniendo en cuenta la factibilidad económica a partir de la combinación de los métodos económicos tradicionales con la lógica difusa. Los modelos estudiados tienen algunos puntos en común con la herramienta que se desea desarrollar. Ejemplo de ello:

- Tienen como objetivo apoyar a los jefes de proyectos en la toma de decisiones referente a la selección y aceptación de los proyectos de software.
- Analizan los criterios económicos que existen para la evaluación de la factibilidad económica de proyectos (FC[3], VAN[4], TIR[5], PR[6]).
- Determinan el orden de prioridad de los proyectos y su aceptación.

1.4 Herramientas informáticas para la evaluación de factibilidad de proyectos

En la actualidad existen varias herramientas informáticas dedicadas a facilitar la toma de decisiones y la evaluación de factibilidad de proyectos de software. Para la evaluación de factibilidad económica se encuentran software como: Managerial Analyzer, EasyPlanEx, EvalAs, Expert Choice y Gespro. Para el

[3] **FC:** Flujo de Caja
[4] **VAN:** Valor Actual Neto.
[5] **TIR:** Tasa Interna De Retorno.
[6] **PR:** Período de Recuperación.

estudio de las herramientas se definieron 6 criterios de comparación, estos son: métodos económicos que trabajan, tratamiento de la incertidumbre, sistema operativo sobre el que corren, propiedad, formato en que realizan los reportes y costo USD. A continuación se realiza un análisis de cada una de ellas.

1.4.1 Managerial Analyzer

Es una herramienta financiera creada para el análisis económico-financiero empresarial, único en su género, que le facilitará la confección, interpretación y realización del estudio y análisis total de la situación económico-financiera de cualquier empresa. Podrá ver y analizar todo un proyecto de inversión, totalmente desglosado por años y con los cálculos de la TIR, VAN, PR, lo que le indicará la viabilidad o inviabilidad de la misma. No tiene en cuenta el tratamiento de la incertidumbre presente en la información. Funciona en las plataformas Windows XP, Windows Vista, Windows 7, Windows 8 y MacOS con virtualización de Windows. Permite la exportación de informes a diferentes formatos como Pdf, Excel, HTML[7], Word. Es una herramienta privativa que tiene un costo de uso de 1.325,00 USD/mes (Analyzer, 2014). Posee otras características como (Analyzer, 2014):

- Los datos los podrá introducir el especialista directamente, o bien, a través de la importación de datos directamente desde "Excel".
- Puede controlar hasta 999 empresas, su funcionamiento es en red o mono puesto.

1.4.2 EasyPlanEx

Permite la evaluación y optimización de proyectos de inversión. Genera automáticamente gráficos y una documentación completa y consistente del proyecto, calcula la probabilidad de ocurrencia de un resultado como el VAN y la TIR. Tiene entre sus objetivos realizar análisis de sensibilidad, medir el riesgo del proyecto considerando la incertidumbre utilizando la técnica de Montecarlo[8]. Es compatible solo con el sistema operativo Windows. Permite importación y exportación de datos de plantillas electrónica. Es una herramienta privativa que tiene un costo de uso de 1.960,00 USD/año, además posee una versión de prueba de 15 días sin costo de licencia (Easyplanex, 2014). Posee otras características como (Easyplanex, 2014):

- Determina el origen de cualquier cambio introducido en la evaluación de un proyecto.
- Reutiliza proyectos tipo.
- Genera automáticamente una documentación completa y consistente del proyecto.

1.4.3 EvalAs

Es una herramienta para la evaluación de proyectos de inversión productivos. A partir de los datos ingresados, calcula la Matriz de FC y luego los principales indicadores financieros: VAN, PR, TIR, TIR modificada. Realiza el manejo de la incertidumbre y análisis de sensibilidad y escenarios. Es compatible

[7] **HTML:** Acrónimo en inglés de Hyper Text Markup Language.

[8] **Técnica de Montecarlo:** es una herramienta de investigación y planeamiento; básicamente es una técnica de muestreo artificial, empleada para operar numéricamente sistemas complejos que tengan componentes aleatorios.

solo con el sistema operativo Windows. Exporta todo el contenido del proyecto que se está trabajando, a un archivo de formato HTML. Es una herramienta privativa que tiene un costo de uso de 1.680,00 USD/mes (EvalAs, 2014). Posee otras características como (EvalAs, 2014):

- Permite el ingreso de información de proyectos de producción, datos de productos y producción, inversiones, costos fijos, costos variables e impuestos.
- A partir de la versión 1.3 los períodos pueden indicarse en meses.
- Permite la comparación con otro proyecto para determinar cuál alternativa es la más rentable.

1.4.4 Expert Choice

Es un sistema para el análisis, síntesis, justificación de decisiones y evaluaciones complejas. Realiza la evaluación de un proyecto de inversión utilizando los indicadores financieros: FC, VAN, TIR, PR. No tiene en cuenta el tratamiento de la incertidumbre presente en la información. Es compatible solo con el sistema operativo Windows. Es una herramienta privativa que tiene un costo de uso de 2.500,00 USD/mes (Choice, 2015). Posee otras características como (Choice, 2015):

- Permite la integración con herramientas como Microsoft Project, Microsoft Excel y el gestor de bases de datos Oracle.
- Permite especificar los roles que participan en el proceso y una jerarquía alternativa.

1.4.5 GESPRO

Es un ecosistema para el desarrollo y la innovación en gestión de proyectos. Realiza los estudios de factibilidad de proyectos de software mediante los indicadores financieros: FC, VAN, TIR, PR. No tiene en cuenta el tratamiento de la incertidumbre presente en la información. Funciona en las plataformas Windows XP y Linux. El uso de la suite GESPRO es de libre costo para el equipo de desarrollo. "El GESPRO abarca varias áreas de la gestión de proyecto, las cuales están presentes en el sistema mediante funcionalidades y módulos. Posibilita no sólo la informatización de las organizaciones, sino que contribuye a la mejora integral de los procesos de la organización alineado con las mejores prácticas para la dirección estratégica" (Gespro, 2015).Posee otras características como (Gespro, 2015):

- Permite el control y seguimiento de proyectos a diferentes niveles: nivel de proyecto, nivel de sucursal y nivel gerencial de una corporación.
- Incorpora para el control y seguimiento de proyectos un cuadro de mando integral y un sistema de indicadores que permite la toma de decisiones en cascada, dosificando la información y permitiendo llegar en caso requerido hasta detalles del funcionamiento de las organizaciones y sus proyectos.
- Permite el control y seguimiento de programas de proyectos y de sucursales de una organización.
- Permite el control y seguimiento de los proyectos desde la red y tomando como modelo teórico la metodología de gestión de proyectos establecida en la UCI.

1.4.6 Valoración de las herramientas

A continuación se muestra una tabla comparativa entre las diferentes herramientas estudiadas para la evaluación de proyectos, para la comparación se definieron varios indicadores de acuerdo a las necesidades de equipo de trabajo:

Herramientas	Métodos económicos	Tratamiento de incertidumbre	Sistema operativo	Propiedad	Formato de reporte	Costo USD/tiempo
Managerial Analyzer	VAN, TIR, PR	No	Windows	Privativa	Pdf, Excel	$ 1.325,00 /mes
EasyPlanEx	VAN, TIR	Si	Windows	Privativa	Pdf	$ 1.960,00 /año
EvalAs	FC, VAN, TIR, PR	SI	Windows	Privativa	HTML	$ 1.680,00 /mes
Expert Choice	FC, VAN, TIR, PR	No	Windows	Privativa	-	$ 2.500,00 /mes
GESPRO	FC, VAN, TIR, PR	No	Windows/ Linux	Libre-Costo	Pdf, CSV	Libre-Costo

Tabla 1: Comparación entre las herramientas para la evaluación de proyectos.

Luego de analizadas estas herramientas se determinó que presentan un conjunto de ventajas para la toma de decisiones, haciendo este proceso más sencillo y eficaz para las personas que la utilizan. Como se pudo apreciar de las herramientas estudiadas, 2 de ellas no tratan los 4 métodos económicos tradicionales (FC, VAN, TIR, PR). En su mayoría son dependientes de una plataforma (Windows), no tratan la incertidumbre presente en la información y son herramientas privativas lo que implica un costo de uso por su licencia por encima de los $ 1.000,00 USD/mes. Siendo el formato Pdf el más utilizado para realizar los reportes. Estas herramientas aunque tienen puntos en común con la herramienta a desarrollar, realizan el cálculo de los métodos económicos de acuerdo a sus características propias. Son insuficientes de forma individual para responder a la problemática planteada en al presente investigación. Por lo anteriormente planteado se decide la implementación de una herramienta web que combine la lógica difusa con los métodos económicos tradicionales para tratar la incertidumbre presente en la información.

1.5 Técnicas de Soft Computing

El Soft Computing es la rama incluida dentro del campo de la Inteligencia Artificial (IA) que estudia la aproximación subsimbólica del conocimiento. Plantea técnicas para la resolución de problemas computacionalmente complejos aplicando metodologías y técnicas que toleran entornos similares a los encontrados en la realidad, a diferencia de los encontrados en modelos abstractos (Álvarez, 2014).

Las técnicas contenidas en el marco del Soft Computing abordan los problemas no basándose en datos exactos y conjuntos completos como en la IA tradicional, sino generando conclusiones, razonamientos y comportamientos inteligentes a partir de conjuntos inexactos de datos (Álvarez, 2014). La clasificación más usada de dichas técnicas en la IA son: la neurocomputación, el razonamiento probabilístico y la lógica difusa (Zadeh, 1996).

1.5.4 Lógica difusa

Es una técnica de inteligencia computacional que permite trabajar con información con alto grado de incertidumbre, en esto se diferencia de la lógica convencional que trabaja con información precisa. Además se puede aplicar cuando ciertas partes de un sistema a controlar son desconocidas y no pueden medirse de forma confiable y cuando el ajuste de una variable puede producir el desajuste de otras. No es recomendable utilizar la lógica difusa cuando algún modelo matemático ya soluciona eficientemente el problema, cuando los problemas son lineales o cuando no tiene solución (Zadeh, 1996).

Los números difusos constituyen una herramienta valiosa para representar cantidades estimadas u observadas en el contexto de la lógica difusa. En particular los números difusos triangulares son los más usados en la práctica por su relativa comodidad de manipulación. La forma triangular se muestra en la siguiente ilustración (Morales, 2014):

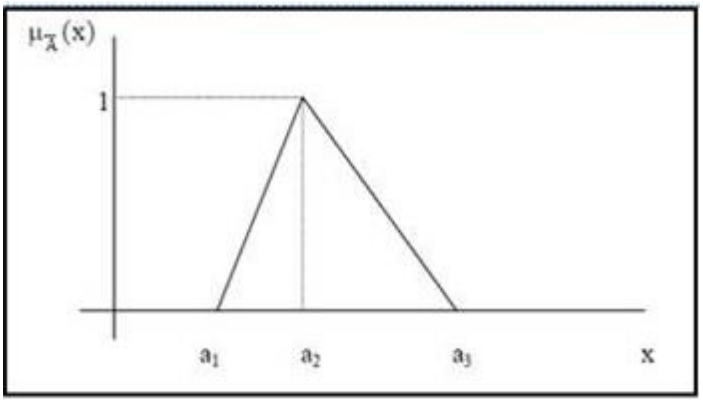


Ilustración 1: Número difuso triangular.

La función de pertenencia para este número triangular viene dado por:

$$\mu_{\tilde{A}}(x) = \begin{cases} \dfrac{x - a_1}{a_2 - a_1} & si\ a_1 < x \leq a_2 \\ \dfrac{a_3 - x}{a_3 - a_2} & si\ a_2 < x \leq a_3 \\ 0 & en\ otro\ caso \end{cases}$$

Ilustración 2: Función de pertenencia para un número difuso triangular.

Las operaciones pueden calcularse siguiendo las reglas que se explican a continuación, a partir de los siguientes números difusos triangulares (Morales, 2014):

$B = [a; b; c]$ y $E = [d; e; f]$

Suma

$a \leq b \leq c$ y $d \leq e \leq f$

$B + E = [a+d; b+e; c+f]$

Resta

$a \leq b \leq c$ y $d \leq e \leq f$

$B - E = [a-f; b-e; c-d]$

Multiplicación

$B * E = [a; b; c] * [d; e; f] = [g; h; i]$

Donde $g = \min(ad, af, cd, cf)$; $h = be$ y $i = \max(ad, af, cd, cf)$

División

Donde $g = \min\left(\frac{a}{d}, \frac{a}{f}, \frac{c}{d}, \frac{c}{f}\right)$; $h = \frac{b}{e}$ e $i = \max\left(\frac{a}{d}, \frac{a}{f}, \frac{c}{d}, \frac{c}{f}\right)$

Ilustración 3: Operaciones con números difusos triangulares.

De las técnicas estudiadas, la neurocomputación se utiliza para el trabajo con las redes neuronales artificiales, siendo eficaz para la creación de sistemas de aprendizaje. El razonamiento probabilístico permite utilizar las redes neuronales para hacer más efectivo el modelado del sistema de razonamiento, además es utilizada en los algoritmos genéticos. La lógica difusa se utiliza para el tratamiento de la imprecisión, la incertidumbre y el razonamiento aproximado. Teniendo en cuenta que el problema planteado va enfocado a tolerar la incertidumbre presente en la información se utilizará la lógica difusa. Para tener una mayor comprensión de los estudios de factibilidad económica se utilizarán los números difusos triangulares en combinación con los métodos económicos tradicionales.

1.6 Métodos económicos tradicionales

Antes de comenzar la ejecución de un proyecto de software es necesario analizar la posible rentabilidad y sobre todo si es factible o no (González, 2011). Para realizar estudios de factibilidad económica se tiene en cuenta los métodos económicos, estos permiten realizar una valoración de la situación económica y financiera existente en un período de tiempo determinado. Los métodos económicos utilizados son FC, VAN, TIR y PR, a continuación se muestra un resumen de cada uno de ellos.

1.6.1 Flujo de caja

Consiste en la diferencia entre los cobros y pagos generados por el proyecto en cada uno de sus años de operación, desde que se efectúa el primer gasto de inversión hasta que la misma es liquidada o sustituida al final de su vida útil (Mesa, 2006). Los flujos de caja se pueden estructurar en tres flujos fundamentales (Abreu, 2012):

1. Flujos iniciales: dentro de éste flujo se encuentran las entradas asociadas a las compras de activos fijos y el capital de trabajo. Los activos fijos están relacionados con las inversiones fijas que se ejecutan para desarrollar el proyecto como puede ser tecnología, terreno, suministros entre otras. Dentro del capital de trabajo se tiene en cuenta salario y seguridad social de los trabajadores, adiestramiento de la fuerza de trabajo.
2. Flujos operacionales: dentro de éste flujo se encuentra el efectivo recibido o expendido como resultado de las actividades económicas.
3. Flujos finales: efectivo recibido o expendido como resultado de actividades financieras, tales como recepción o pago de préstamos, emisiones o recompra de acciones.

La secuencia de pasos para obtener los flujos totales es:

$+ costos\ y\ gastos$

$+ ventas\ o\ ingresos$

$+ depreciación$

$= \ utilidad\ antes\ de\ impuesto$

$+ impuestos$

$= \ utilidad\ después\ de\ impuestos$

$+ readición\ de\ la\ depreciación$

$= flujos\ operacionales\ totales$

$+ total\ de\ flujos\ iniciales$

$+ flujos\ finales\ totales$

$= flujos\ totales$

Donde:

$costos\ y\ gastos$ = $mano\ de\ obra$ + $materiales$

$impuestos$ = $utilidad\ antes\ de\ impuesto$ * $(0{,}3)$

$readición\ de\ la\ depreciación$ = $depreciación$ * (-1)

$flujos\ finales\ totales$ = $recuperación\ del\ capital\ de\ trabajo$ + $valor\ de\ salvamento$

$recuperación\ del\ capital\ de\ trabajo$ = - $capital\ de\ trabajo$

$valor\ de\ salvamento$ = $(\sum compra\ activos\ fijos$ - $\sum depreciación)$ * (-1)

1.6.2 Valor actual neto

Es el valor actual de los flujos de caja esperados. Una definición más explícita correspondería, entonces, a la que lo define como el valor actualizado del saldo entre el flujo de ingresos y egresos en efectivo generados por un proyecto durante su vida útil (Mesa, 2006). De forma general se puede expresar como:

$$VAN = -I + \sum_{j=1}^{n} \frac{FCj}{(1+k)^{n}}$$

Dónde:

I = Costo inicial de inversión.

FCj = Flujo de Caja en el año j.

k = Tasa de descuento.

n = Período de vida útil.

Siendo el criterio de selección:

Si el VAN > 0 entonces se acepta el proyecto.

Si el VAN = 0 entonces se selecciona otro criterio de evaluación.

Si el VAN < 0 entonces no se acepta el proyecto.

1.6.3 Tasa interna de retorno

Se define como aquella tasa de actualización o descuento r, que hace cero la rentabilidad absoluta neta de la inversión. Es aquella tasa de descuento que iguala el valor actual de la corriente de cobros con el valor actual de la corriente de pagos (Mesa, 2006). Analíticamente el criterio de la TIR se expresa como:

$$r = k1 + \frac{VANp(k2 - k1)}{VANp + |VANn|}$$

Dónde:

r = Tasa interna de rendimiento.

$k1$ = Tasa de actualización en que el VAN es positivo.

$VANp$ = Importe del VAN positivo a la tasa de actualización $k1$.

$k2$ = Tasa de actualización en que el VAN es negativo.

$VANn$ = Importe del VAN negativo a la tasa de actualización $k2$.

La TIR representa la rentabilidad general del proyecto y el criterio de selección corresponderá al proyecto que tenga la mayor TIR. Sin embargo, esta será sólo una condición necesaria, requiriéndose como criterio de suficiencia que el valor de la TIR obtenida sea mayor que k, es decir $r > k$.

1.6.4 Período de recuperación

Se define como el período de tiempo en que mediante los ingresos generados por el proyecto se recupera la inversión realizada (Mesa, 2006). El criterio de selección sería el proyecto con menor período de recuperación. Para el cálculo del mismo se establece la siguiente fórmula:

$$PR = Tn + \frac{|SA1|}{|SA1| + SA2} - m$$

Dónde:

PR = Período de Recuperación de la inversión.

Tn = Número de años con saldo acumulado negativo desde el primer gasto anual de inversión.

$SA1$ = Valor absoluto del último año con efecto negativo en el saldo acumulado.

$SA2$ = Valor absoluto del primer año con efecto positivo en el saldo acumulado.

m = Período de construcción y montaje.

Los métodos económicos estudiados sustentan su trabajo sobre el cálculo con números exactos, no tratando temas como la incertidumbre presente en la información. Debido a esto, el equipo de desarrollo en conjunto con el cliente definió combinar los métodos económicos con los números difusos triangulares, para no obtener como resultado un número exacto sino un rango de valores teniendo en cuenta la incertidumbre presente en la información.

1.7 Ingeniería de requisitos

La ingeniería de requisitos se define como “el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo que satisfaga las necesidades del usuario.” (Pressman, 2005).

Llevar a cabo de manera adecuada el proceso de ingeniería de requisitos disminuye la probabilidad de fracaso de un proyecto. Los requisitos bien definidos permiten conocer de un modo conciso qué debe ser capaz de realizar el software a desarrollar. Orienta las actividades, recursos y esfuerzos de manera eficiente permitiendo la disminución de costos y retrasos. Todo ello provoca una mejora en la calidad del software, puesto que los requisitos bien especificados podrán probarse de manera efectiva, y por tanto cumplirse (Fernández, 2012).

Existen varias técnicas para llevar a cabo el proceso de captura de requisitos. Es tarea del equipo de desarrollo determinar cuál o cuáles son las más factibles a utilizar para la herramienta a desarrollar. Algunas de estas técnicas son (Fernández, 2012):

Entrevistas: es un medio tradicional de obtención de requisitos. La entrevista es un método muy efectivo que permite conocer los problemas de los clientes y encontrar requisitos generales. Para aplicar este método es necesario conocer la forma en que se debe de realizar una entrevista para lograr una buena comunicación entre el entrevistador y el entrevistado.

Prototipos: es la representación o visualización de parte de la herramienta. Es una herramienta valiosa para clarificar requisitos confusos. Proveen a los usuarios un contexto para entender mejor qué información necesitan proporcionar.

Tormenta de ideas: esta técnica es usada para generar nuevas ideas y encontrar la solución a cuestiones específicas. Es muy común en los comienzos del proceso de ingeniería de requisitos.

Observación: este método consiste en la identificación de requisitos cuando se observan a las personas haciendo su trabajo diario. Es muy usado para encontrar requisitos adicionales cuando el usuario es incapaz de explicar los requisitos que necesita para la nueva herramienta.

Después de analizar las definiciones de cada una de las técnicas de captura de requisitos anteriormente citadas se puede pensar que son todas factibles si se emplean combinaciones de ellas, ya que por sí solas no son suficientes para realizar un buen proceso de captura de requisitos.

Existen varias técnicas para validar los requisitos, las cuales se aplican con el objetivo de examinar las especificaciones para asegurar que todos los requisitos han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones y que los errores detectados hayan sido corregidos (Pressman, 2005). Algunas de estas técnicas son (Fernández, 2012):

Revisiones de verificabilidad: los requisitos son analizados sistemáticamente por un equipo de revisores, en busca de anomalías y/u omisiones.

Prototipos de interfaz: se muestra un modelo ejecutable de la herramienta a los usuarios finales y los clientes, para que puedan ver si dicho modelo cumple con sus necesidades reales.

Generación de casos de pruebas: la elaboración de casos de pruebas puede revelar los problemas con que puede contar un requisito y es parte fundamental de la programación externa (Sommerville, 2005).

1.8 Ambiente de desarrollo

Seguidamente se realiza un análisis de la metodología de desarrollo de software y los lenguajes a utilizar.

1.8.1 Metodología de desarrollo

Para regir los procesos de desarrollo están definidas varias metodologías. Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las llamadas metodologías tradicionales y las metodologías ágiles. Las tradicionales son recomendadas para los proyectos de grandes dimensiones y con grandes equipos de desarrollo. En tanto las metodologías ágiles son recomendadas para proyectos de poca duración y equipos pequeños (Letelier, 2012). Debido a la necesidad de llevar a cabo el desarrollo de la propuesta de solución en un corto período de tiempo, manteniendo el intercambio constante con el cliente y garantizar los riesgos a los posibles requisitos cambiantes, se hace necesario optar por una metodología ágil de desarrollo de software en lugar de una metodología tradicional.

Según lo planteado por (Letelier, 2012), los principales valores de las metodologías ágiles son:

- Los individuos y las interacciones entre ellos lo más importante son más importante que los procesos y herramientas empleadas.
- Desarrollar un software que funcione es más importante que conseguir una buena documentación.

- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Entre las metodologías ágiles se encuentran Programación Extrema (XP[9]), SCRUM, Cristal Methodologies, Proceso Unificado Ágil (AUP). Cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos (Letelier, 2012). Debido a las características de la herramienta a desarrollar, no siempre se definen completamente los requisitos de la aplicación antes de comenzar la implementación, por lo que se hace necesario el diálogo constante entre el cliente y el equipo de desarrollo. De esta manera metodologías como XP o SCRUM establecen una forma de trabajo en la que la comunicación fluida garantiza qué si se produce un cambio en los requisitos el equipo de desarrollo será capaz de adaptar la aplicación en un tiempo razonable.

Programación extrema

Más que una metodología, XP se considera una disciplina, la cual está sostenida por valores y principios propios de las metodologías ágiles. Existen cuatro valores que cumplen su papel como pilares en el desarrollo de las metodologías livianas (Tobón, 2007).

- **La comunicación:** entre los usuarios y los desarrolladores.
- **La simplicidad:** al desarrollar y codificar los módulos de la herramienta.
- **La retroalimentación:** concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.
- **El coraje:** el equipo de desarrollo debe estar preparado para enfrentarse a los continuos cambios que se presentarán en el transcurso de la actividad.

A partir de los valores se plantea una serie de prácticas que sirven de guía para los desarrolladores en esta metodología. Uno de los aspectos más importantes para XP son las 12 reglas que se plantean, las cuales se caracterizan por su grado de simplicidad, además de que cada regla se complementa con las demás. A continuación se enuncian las reglas (Tobón, 2007):

1. El desarrollo está dirigido por pruebas.
2. El juego de la planificación.
3. Cliente in-situ.
4. Programación en parejas.
5. Entregas pequeñas.
6. Refactorización sin piedad.
7. Integración continúa con el código.
8. Diseño simple.

[9] **XP:** Acrónimo en inglés de eXtreme Programming.

9. Utilización de metáforas del sistema.
10. Propiedad colectiva del código.
11. Convenciones de código.
12. Las semanas son de 40 horas.

Scrum

Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. Scrum hace patente la eficacia relativa de las prácticas de gestión de producto y de desarrollo, de modo que puedan ser mejoradas. Scrum se fundamenta en la teoría de que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Emplea una aproximación iterativa e incremental para optimizar la predictibilidad y controlar el riesgo (Trigas, 2011).

La implementación del control empírico de procesos se basa en tres pilares (Trigas, 2011):

- **Transparencia:** todo lo que afecta al resultado del trabajo debe ser conocido y visible para todo el mundo.
- **Inspección:** es necesario realizar un seguimiento de la evolución del trabajo para detectar desviaciones con respecto a lo previsto.
- **Adaptación:** en el momento en el que se detecta alguna desviación o se tienen indicios de que esta se puede producir, es necesario actuar en consecuencia para adaptarse a las nuevas circunstancias.

Scrum plantea la necesidad de conocer profundamente cada uno de los procesos que intervendrán en el desarrollo del software. En el caso de la presente investigación el equipo de desarrollo no cuenta con suficiente experiencia en el trabajo con las tecnologías y los procesos asociados a la construcción de la herramienta que se propone. Teniendo en cuenta lo antes planteado se escoge XP como metodología de desarrollo, debido a su capacidad de comunicación, simplicidad y reutilización del código, características que facilitarán llevar a cabo el proceso de implementación de la solución propuesta.

1.8.2 Herramienta CASE[10] de modelado

Las herramientas CASE son herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software. Visual Paradigm (VP) 8.0 es una de las herramientas que utiliza UML[11] como lenguaje de modelado, está considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java, .NET y PHP[12]. Se

[10] **CASE:** Acrónimo en inglés de Computer Aided Software Engineering.
[11] **UML:** Acrónimo en inglés de Unified Modeling Language.
[12] **PHP:** Acrónimo en inglés de HyperText Preprocessor.

integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, Oracle JDeveloper y NetBeans IDE. Proporciona valiosa ayuda a los profesionales visualizando, comunicando y aplicando sus diseños. Es una herramienta multiplataforma, capaz de generar código SQL[13]. Posee perfecta integración con el IDE de desarrollo utilizado y con el gestor de bases de datos PostgreSQL.

1.8.3 Marco de trabajo

Symfony2 es la versión más reciente de Symfony. Es un proyecto PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional, basado en el Modelo Vista Controlador (MVC) (Eguiluz, 2011). Su código y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT[14] de software libre (Symfony, 2014). Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un determinado proyecto. Symfony2 utiliza el ORM[15] llamado Doctrine, el mismo es hogar de varias bibliotecas de PHP, se centra principalmente en el almacenamiento de la base de datos y el mapeo de objetos. Como gestor de plantillas se utilizará Twig, su principal característica se encuentra en la herencia de plantillas, permite separar el código PHP del HTML lo que garantiza que los usuarios pueden modificar el diseño de la plantilla, sin modificar el código (Eguiluz, 2011).

1.8.4 Lenguaje del lado del servidor PHP

Es un lenguaje interpretado del lado del servidor, de propósito general, ampliamente usado y que está diseñado especialmente para el desarrollo web y puede ser embebido dentro del código HTML. Se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida.

Es un lenguaje multiplataforma con capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, tiene capacidad de expandir su potencial utilizando la enorme cantidad de módulos, llamados extensiones, es libre, por lo que se presenta como una alternativa de fácil acceso para todos, además de que permite la utilización de las técnicas de Programación Orientada a Objetos (PHP, 2014).

1.8.5 Lenguaje del lado del cliente HTML5

HTML es el lenguaje usado para escribir las páginas web, describe la estructura y el contenido usando solo texto y lo complementa con objetos tales como imágenes, flash y otros. Es más ligero al ser más sencillo y simple el código, lo que permite que las páginas escritas en este lenguaje carguen más rápido en el navegador. HTML5 incluye etiquetas orientadas principalmente a los buscadores, para facilitar la comprensión del contenido de las páginas. Otras de las razones es el empleo del micro formato en las páginas web, que algunos son totalmente incompatibles con otros lenguajes por lo que no validan correctamente a no ser que se use HTML5 (HTML5, 2015).

[13] **SQL:** Acrónimo en inglés de Structured Query Language.
[14] **MIT:** Acrónimo en inglés de Massachusetts Institute of Technology.
[15] **ORM**: Acrónimo en inglés de Object Relational Mapping.

1.9 Herramientas de desarrollo

A continuación se describen las herramientas y tecnologías a utilizar.

1.9.1 Sistema gestor de bases de datos

PostgreSQL es un SGBD[16] objeto-relacional, distribuido bajo licencia BSD[17] y con su código fuente disponible libremente. Es el SGBD de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales (PostgreSQL, 2014).

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (PostgreSQL, 2014). Se utilizará el SGBD PostgreSQL debido a que los métodos económicos tradicionales serán implementados en el lenguaje plpgsql.

Algunas de sus características son (PostgreSQL, 2014):

- SGBD multiplataforma compatible con Windows de 32/64 bit, Linux y Unix en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, MAC OS X, SOLARIS).
- Completa documentación.

1.9.2 Servidor de aplicaciones web

Se utilizará como servidor de aplicaciones web Apache 2.0. Este servidor es una tecnología gratuita de código fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable, es decir, se pueden elegir qué características van a ser incluidas en el servidor seleccionando, qué módulos se van a cargar, ya sea al compilar o al ejecutar el servidor (Apache, 2014).

1.9.3 Entorno de desarrollo integrado

Netbeans 8.0 es un entorno de desarrollo integrado de código abierto y libre de costo, escrito en lenguaje Java, lo cual lo convierte en multiplataforma. Tiene todas las herramientas necesarias para crear aplicaciones de escritorio y web profesionales, mediante el uso de diferentes lenguajes de programación como Java, C/C++, PHP, Ruby, JavaScript, HTML y Groovy (NetBeans, 2014). Entre sus características se encuentran: un editor de código muy potente, permite depurar y ejecutar programas, tiene integración con marcos de trabajo de desarrollo de PHP, sus funcionalidades se pueden extender con el uso de complementos o extensiones que se le incorporan según las necesidades del programador (NetBeans, 2014).

Se utilizará para la implementación de la herramienta el entorno de desarrollo integrado Netbeans, porque permite el uso de un amplio rango de tecnologías de desarrollo con aplicaciones web. Tiene gran interacción con Symfony2, siendo este el marco de trabajo a utilizar. Es un editor de código multilenguaje.

[16] **SGBD:** Sistema Gestor de Bases de Datos.
[17] **BSD:** Acrónimo en inglés de Berkeley Software Distribution. Es una licencia de software libre que permite el uso del código fuente en software no libre.

Simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda y estructurando la visualización de manera ordenada. Permite el acceso al gestor de base de datos utilizado, y desde el propio IDE realizar consultas, modificaciones y la visualización de las tablas.

1.10 Patrones de diseño

Los patrones de diseño de software permiten describir fragmentos de diseño y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos. Los patrones más populares son los General Responsibility Assignment Software Patterns (GRASP) y Gang of Four (GOF).

1.10.1 Patrones GRASP

Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objetos esenciales y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades (Larman, 1999).

Entre los patrones GRASP se encuentran (Larman, 1999):

- **Experto:** asigna una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad.
- **Creador:** asigna a la clase *B* la responsabilidad de crear una instancia de clase *A*.
- **Controlador:** asigna la responsabilidad de administrar un mensaje de eventos del sistema a una clase.
- **Bajo acoplamiento:** asigna responsabilidades de modo que se mantenga el bajo acoplamiento entre las clases.
- **Alta cohesión:** asigna responsabilidad de modo que se mantenga la alta cohesión entre las clases.

1.10.2 Patrones GOF

Se dieron a conocer a principios de los años 90 con el libro “Design Patterns. Elements of Reusable Object-Oriented Software”. En dicho libro se hace una recopilación de 23 patrones de diseño comunes, clasificados en tres grupos de acuerdo a su naturaleza (Prieto, 2009):

1. **Patrones creacionales:** inicialización y configuración de objetos.
2. **Patrones estructurales:** separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
3. **Patrones de comportamiento:** más que describir objetos o clases, describen la comunicación entre ellos.

Algunos de los patrones de diseño GOF son:

- **Método fábrica:** define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- **Inicialización perezosa:** es la táctica de retrasar la creación de un objeto, el cálculo de un valor, o algún otro proceso costoso hasta la primera vez que es realmente necesario.

Para desarrollar una herramienta que cumpla con los requerimientos del cliente, es necesario desarrollar un diseño satisfactorio, para esto existen diversos patrones de diseño. Los patrones se enmarcan en dos grandes grupos, los patrones GRASP y los patrones GOF. El uso de los patrones GRASP permitirá asignarle las responsabilidades a las clases que cuentan con la responsabilidad necesaria. El uso de los patrones GOF permitirá obtener un diseño con una mayor calidad.

1.11 Diseño arquitectónico

Para la implementación de la solución se utilizará el estilo en capas (capa de presentación, capa de negocio, capa de acceso a datos y capa de datos) y el patrón Modelo Vista Controlador (MVC). El mismo separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos utilizado por la aplicación (Prieto, 2009). En la siguiente ilustración se puede presenciar el uso de este patrón según Symfony2 (Escobar, 2013).

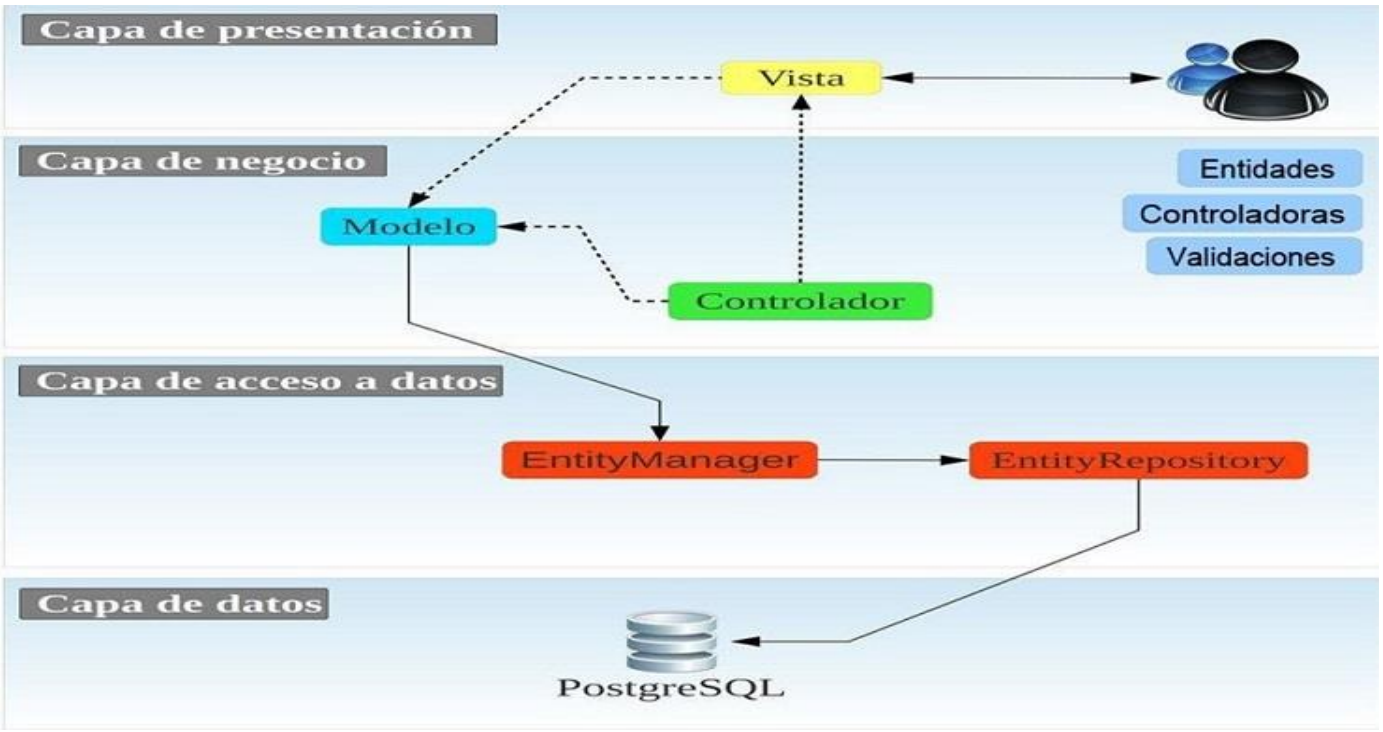


Ilustración 4: Patrón arquitectónico MVC.

1.12 Métricas para validar el diseño

Existen varias métricas para validar el diseño realizado. Es tarea del equipo de desarrollo determinar cuál o cuáles son más factibles a utilizar. Se emplearán las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC), diseñadas para evaluar los siguientes atributos de calidad (Diseño, 2014):

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

La métrica TOC se basa en la cantidad de operaciones que tiene encapsulada una clase y evalúa los siguientes atributos de calidad:

Fuente: elaboración propia.

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

Tabla 2: Atributos de calidad evaluados por la métrica TOC.

Para evaluar la métrica son necesarios los valores de los umbrales para los parámetros de calidad. En la tabla 3 se muestran los umbrales para la métrica TOC.

Fuente: elaboración propia.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<= promedio
	Media	Entre promedio y 2* promedio
	Alta	> 2* promedio

Complejidad de implementación	Baja	<= promedio
	Media	Entre promedio y 2* promedio
	Alta	> 2* promedio
Reutilización	Baja	> 2* promedio
	Media	Entre promedio y 2* promedio
	Alta	<= promedio

Tabla 3: Rango de valores para la métrica TOC.

La métrica RC está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Fuente: elaboración propia

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

Tabla 4: Atributos de calidad evaluados por la métrica RC.

Para evaluar la métrica son necesarios los valores de los umbrales para los parámetros de calidad. En la tabla 5 se muestran los umbrales para la métrica RC.

Fuente: elaboración propia.

Atributo	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	> 2
Complejidad de mantenimiento	Baja	<= promedio
	Media	Entre promedio y 2* promedio
	Alta	> 2* promedio
Reutilización	Baja	> 2 * promedio
	Media	Entre promedio y 2* promedio
	Alta	<= promedio

Cantidad de pruebas	Baja	<= promedio
	Media	Entre promedio y 2* promedio
	Alta	> 2* promedio

Tabla 5: Rango de valores para la métrica RC.

Antes de comenzar la fase de implementación es necesario verificar que el diseño realizado es satisfactorio. Para esto es necesario utilizar las diferentes métricas existentes para la validación del diseño, estas tienen como objetivo mejorar la calidad del producto y del diseño. Chidamber y Kemerer proponen 6 métricas para medir 5 atributos de calidad, los atributos que miden son: acoplamiento, complejidad de una clase, reutilización, cohesión y herencia. Abreu y Melo proponen 6 métricas para evaluar 4 atributos de calidad, los atributos son: encapsulamiento, herencia, polimorfismo y paso de mensajes. Las métricas propuestas por estos dos autores están centradas en la herencia y el polimorfismo, debido a estos no cumple con las características de la herramienta a desarrollar. Por lo anteriormente expuesto se utilizarán las métricas TOC y RC propuestas por Lorenz y Kidd, con el objetivo de evaluar la mayor cantidad de atributos.

1.13 Pruebas

Pressman, declara que las pruebas de software son un elemento crucial para garantizar la calidad del producto y validar las especificaciones, el diseño y la programación. Estas tienen como objetivo, además de descubrir errores, medir el grado en que el software cumple con los requerimientos definidos (Pressman, 2005). La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (Beck, 2000). Además de las pruebas propuestas por la metodología seleccionada se realizarán pruebas de caja negra, para examinar algunos aspectos del modelo fundamentalmente de la herramienta sin tener mucho en cuenta la estructura interna del software.

1.13.1 Pruebas de caja blanca

Es un método de diseño de casos de pruebas que usa la estructura de control del diseño procedimental para obtener los casos de pruebas (Pressman, 2005). Mediante los métodos de pruebas de caja blanca (Pressman, 2005):

- Se garantiza que se recorra por lo menos una vez los caminos independientes de cada módulo.
- Se ejecutan todas las decisiones lógicas en sus opciones verdaderas y falsas.
- Se ejercitan todos los bucles en sus límites.
- Se usan las estructuras internas de datos para asegurar su validez.

Existen varias técnicas de pruebas de caja blanca, algunas de ellas son (Pressman, 2005):

Camino básico: permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de pruebas obtenidos garantizan que se ejecute al menos una vez cada sentencia del programa (Pressman, 2005).

Prueba de condición: es un método de diseño de casos de pruebas que ejercita las condiciones lógicas contenidas en el módulo de un programa. El propósito de esta técnica es detectar no solo errores en las condiciones, sino también otros tipos de errores (Pressman, 2005).

Pruebas de flujos de datos: selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variantes del programa (Pressman, 2005).

Inicialmente puede parecer que una prueba de caja blanca profunda puede llevar a tener programas correctos, definiendo todos los caminos lógicos, generar casos de pruebas que examinen exhaustivamente la lógica del programa. Lamentablemente, estas pruebas incluso para programas el número de caminos lógicos que genera puede ser enorme. Las pruebas de caja blanca no se deben desechar como impracticables, se deben escoger una serie de caminos importantes a ejecutar (Pressman, 2005). La técnica optada fue camino básico, dado que permite obtener una medida de la complejidad lógica del diseño y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

1.13.2 Pruebas de caja negra

La prueba de caja negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas no son una alternativa a las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca (Pressman, 2005). Las pruebas de caja negra intentan encontrar errores en las siguientes categorías (Pressman, 2005):

- Funciones incorrectas o ausentes
- Errores en la interfaz.
- Errores en la estructura de datos.
- Rendimiento.
- Inicialización y terminación.

Para las pruebas de caja negra existen varias técnicas, algunas de ellas son (Pressman, 2005):

Partición equivalente: permite examinar los valores válidos e inválidos de las entradas existentes en el software. Además descubre de forma inmediata una clase de errores que, de otro modo, requerirían la

ejecución de muchos casos antes de detectar el error genérico. Esto permite reducir el número de casos de pruebas a elaborar (Pressman, 2005).

Análisis de valores límites: los errores tienden a darse más en los límites del campo de entrada que en el centro. Esta es una técnica que complementa la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, lleva a elección de casos de pruebas en los extremos de la clase (Pressman, 2005).

1.13.3 Pruebas de aceptación

El objetivo de este tipo de prueba es comprobar si el cliente está satisfecho o no con el producto desarrollado y si este producto cumple con sus expectativas. El usuario debe ser el que realice las pruebas, ayudado por personas del equipo de pruebas, siendo deseable, que sea el mismo usuario quien aporte los casos de pruebas (Moreno, 2005). XP propone pruebas de aceptación o pruebas funcionales y estas son supervisadas por el cliente basándose en los requerimientos tomados de las historias de usuario (Gálvez, 2013). Algunas de las características de las pruebas de aceptación son: la participación activa del usuario, que deje ejecutar los casos de pruebas ayudado por miembros del equipo de pruebas. Están enfocadas a demostrar que no se cumplen los requisitos (Moreno, 2005).

1.14 Conclusiones del capítulo

Después de realizado el estudio para este primer capítulo se puede arribar a las siguientes conclusiones parciales:

- Se construyó el marco teórico de la investigación, permitiendo el esclarecimiento de conceptos y definiciones esenciales para la investigación.
- Se realizó un estudio del arte sobre los modelos y herramientas para la evaluación de proyectos, quedando demostrado que ninguna resuelve la problemática planteada.
- La lógica difusa en combinación con los números difusos triangulares resuelve el problema de la incertidumbre presente en la información.
- Se eligió como metodología de desarrollo de software XP, ya que por sus características resultó ser la apropiada para guiar el proceso de desarrollo de la solución.
- Se definieron las herramientas y tecnologías para el desarrollo de la herramienta.
- Se realizó un estudio de los patrones y métricas para validar el diseño permitiendo realizar un diseño con una mayor calidad.

Capítulo 2: Análisis y diseño de la solución propuesta

2.1 Introducción

En el presente capítulo se describe la estructura y componentes de la herramienta a desarrollar. Se realiza una descripción detallada del desarrollo de la herramienta durante las fases de planificación y diseño que propone la metodología XP. Se describe el estilo y patrón arquitectónico utilizado, así como los patrones de diseño. Se valida el diseño mediante las métricas Tamaño Operacional de Clases y Relaciones entre Clases.

2.2 Propuesta de solución

Utilizando la información recopilada en el capítulo anterior, se propone el desarrollo de una herramienta que combine los métodos económicos con la lógica difusa. Con la herramienta se ayudará a los decisores en la toma de decisiones mediante el análisis de factibilidad económica tolerando la incertidumbre presente en la información. Para un mejor entendimiento del tema en cuestión el equipo de desarrollo definió confeccionar el modelo de dominio. Según (Larman, 1999), un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés.

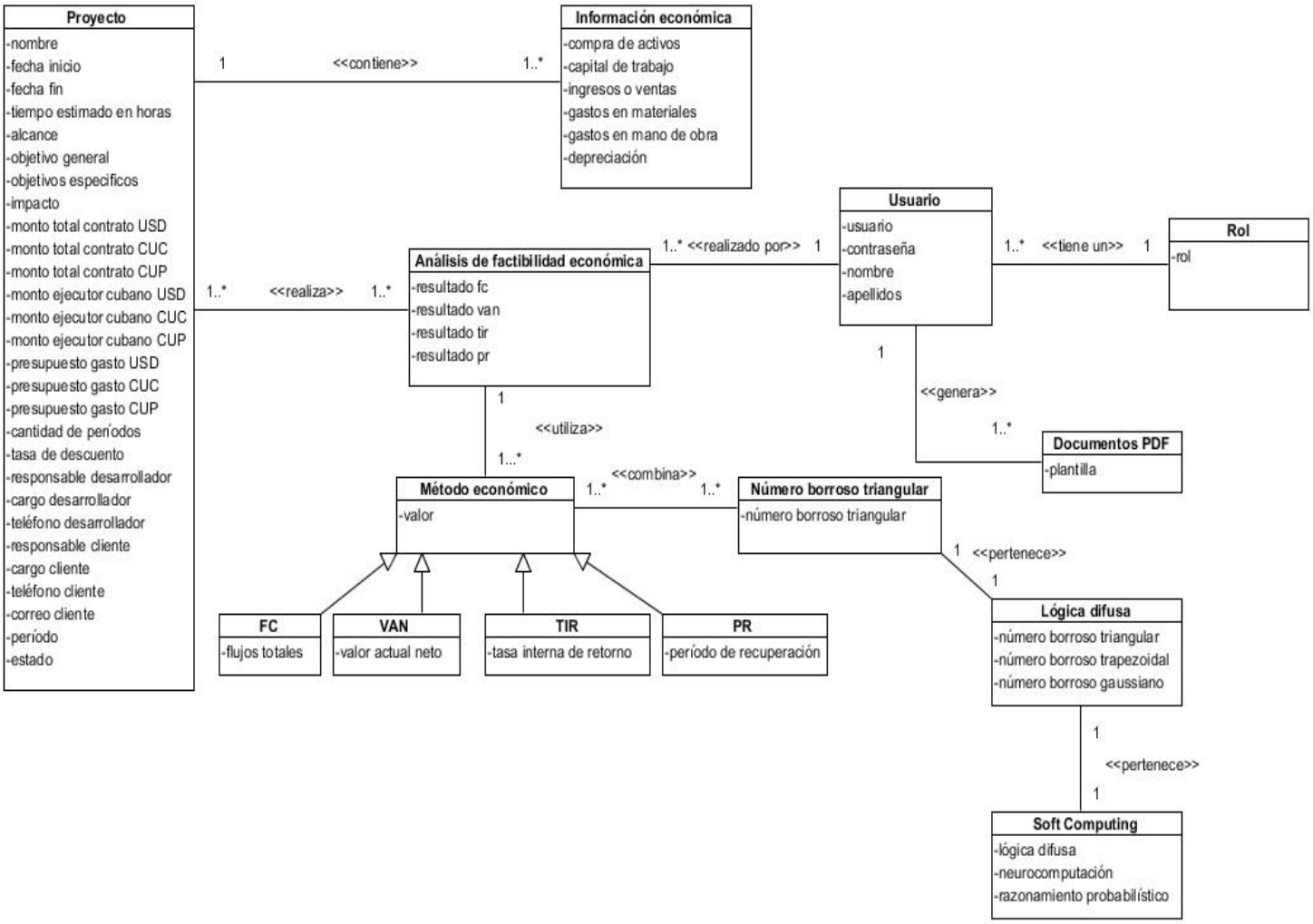


Ilustración 5: Modelo de dominio.

Dentro de la lógica difusa se utilizarán los números difusos triangulares siendo una herramienta potente en entornos de incertidumbre para la toma de decisiones. Un número borroso triangular puede expresarse como un número impreciso: A = (a1, a2, a3) siendo a1≤ a2 ≤ a3, lo que implica simplicidad en el cálculo tanto de los números difusos en sí como de las operaciones entre ellos. Posee dos valores críticos:

1. Un valor central cuyo nivel de confianza α es igual a 1.
2. Dos valores extremos cuyos niveles de confianza α son iguales a cero.

A continuación se muestra un ejemplo del cálculo del VAN con números borrosos triangulares, para un proyecto con vida útil de 5 años:

Datos:

Inversión inicial = 25.000,00 y k = 0.05.

FC1 = 1.000,00; 4.000,00; 6700	FC2 = 2.000,00; 5.000,00; 8.500,00	FC3 = 3.200,00; 6.100,00; 9.300,00
FC4 = 3.400,00; 5.400,00; 7.050,00	FC5 = 2.900,00; 4.500,00; 5.850,00	

$$VAN = -25.000,00 + \frac{1.000,00;\ 4.000,00;\ 6.700,00}{1,05^{1}} + \frac{2.000,00;\ 5.000,00;8500}{1,05^{2}} + \frac{3.200,00;\ 6.100,00;\ 9.300,00}{1,05^{3}} + \frac{3.400,00;\ 5.400,00;\ 7.050,00}{1,05^{4}} + \frac{2.900,00;\ 4.500,00;\ 5.850,00}{1,05^{5}}$$

VAN = -25.000,00 + (10.600,11 + 21.582,51 + 32.508,05)

VAN = (-14.399,89; -3.417,49; 7.508,12)

Este resultado se interpreta de la manera siguiente: el proyecto tendrá un VAN que se encontrará entre una pérdida de $14.399,89 y una ganancia de $7.508,12, siendo lo más posible que se obtenga una pérdida de $3.417,49.

EL cálculo de los métodos económicos tradicionales en combinación con los números difusos triangulares permitirá obtener un mayor sinceramiento de la información, debido a que no se obtendrá un valor exacto sino un rango de valores. Los decisores de proyectos contarán con varias alternativas en vez de una sola a la hora de la selección de un proyecto, teniendo presente el valor con mayor grado de pertenencia.

2.3 Fase de planificación

La metodología XP define como fase inicial la planificación. Esta es una fase corta, en la que el cliente, el jefe de proyecto y los desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario.

También se realizará una estimación del esfuerzo que costará implementar cada una de ellas. El resultado de la fase de planificación es un plan de entregas (Beck, 2000).

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de planificación toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (Letelier, 2012). El artefacto generado en la fase de planificación son las historias de usuario, derivándose de estas la estimación de esfuerzo por historia de usuario y el plan de iteraciones.

2.3.1 Historias de usuario

Técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las historias de usuario es dinámico y flexible. En cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada una de ellas es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en pocas semanas (Letelier, 2012). A continuación se describen 2 historias de usuario de la herramienta a desarrollar de un total de 9, el resto pueden visualizarse en el **Anexo 1**.

Historia de usuario	
Número: HU #02	**Historia de usuario:** Gestionar proyecto.
Modificación de la historia de usuario: ninguna	
Programador: Ubesnel Rojas Orozco y Orlando Guilarte Cutiño.	**Iteración asignada:** primera
Prioridad: alta	**Puntos estimados:** 5 días
Riesgo en desarrollo: bajo	**Puntos reales:** 5 días
Descripción Inicialmente se deben listar todos los proyectos que han sido adicionados con anterioridad. Los proyectos deben mostrar la siguiente información: nombre oficial del proyecto, objetivo general, fecha inicio y fecha fin y el estado. Además debe permitir realizar las siguientes acciones: adicionar, modificar, eliminar, buscar, listar y mostrar información del proyecto.	
Observaciones ➢ Las acciones sobre los proyectos solo pueden ser realizadas por usuarios con rol especialista. ➢ Para modificar, eliminar, buscar, mostrar y listar un proyecto, debe haber sido adicionado al menos un proyecto con anterioridad.	

Tabla 6: Descripción de la historia de usuario gestionar proyecto.

Historia de usuario	
Código: HU # 07	**Historia de usuario:** Realizar análisis de factibilidad económica
Modificación de la historia de usuario: ninguna	
Programador: Ubesnel Rojas Orozco y Orlando Guilarte Cutiño.	**Iteración asignada:** primera
Prioridad: alta	**Puntos estimados:** 20 días
Riesgo en desarrollo: alta	**Puntos reales:** 20 días
Descripción La funcionalidad realizar análisis de factibilidad económica debe permitir calcular cada uno de los métodos económicos (FC, VAN, PR, TIR) para todos los proyectos que han sido adicionados al sistema. Inicialmente deben mostrarse todos los proyectos que han sido adicionados con anterioridad. Además debe permitir realizar las siguientes acciones: calcular FC, calcular VAN, calcular TIR, calcular PR.	
Observaciones ➢ Las acciones sobre los proyectos solo pueden ser realizadas por usuarios con rol especialista. ➢ Debe existir un proyecto adicionado con anterioridad y tener asociado una información económica. ➢ El proyecto debe estar asociado a un proceso de evaluación.	

Tabla 7: Descripción de la historia de usuario realizar evaluación de proyectos.

2.3.2 Requisitos funcionales del software

Según (Sommerville, 2005) los requisitos funcionales son "declaraciones de los servicios que deben proporcionar el sistema de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares". En la tabla 8 se muestran los requisitos funcionales de la herramienta a desarrollar.

Código	**Descripción**
HU#1	**Gestionar usuario**
RF- #01	Adicionar usuario
RF- #02	Modificar usuario
RF- #03	Eliminar usuario
RF- #04	Buscar usuario
RF- #05	Listar usuarios
HU#2	**Gestionar proyecto**
RF- #06	Adicionar proyecto
RF- #07	Modificar proyecto

RF- #08	Eliminar proyecto
RF- #09	Buscar proyecto
RF- #10	Listar proyectos
RF- #11	Mostrar información del proyecto
HU#3	**Gestionar información económica**
RF- #12	Adicionar información económica
RF- #13	Modificar información económica
RF- #14	Eliminar información económica
RF- #15	Mostrar información económica
HU#4	**Gestionar proceso de evaluación**
RF- #16	Adicionar proceso de evaluación
RF- #17	Modificar proceso de evaluación
RF- #18	Eliminar proceso de evaluación
RF- #19	Mostrar proceso de evaluación
RF- #20	Buscar proceso de evaluación
RF- #21	Listar proceso de evaluación
HU#5	**Gestionar proyecto a evaluar durante una evaluación**
RF- #22	Adicionar proyectos al proceso evaluativo
RF- #23	Eliminar proyectos del proceso evaluativo
HU#6	**Gestionar método a evaluar durante una evaluación**
RF- #24	Adicionar método económico al proceso evaluativo
RF- #25	Eliminar método económico del proceso evaluativo
HU#7	**Realizar análisis de factibilidad económica**
RF- #26	Evaluar método flujo de caja con números difusos triangulares
RF- #27	Evaluar método valor actual neto con números difusos triangulares
RF- #28	Evaluar método período de recuperación con números difusos triangulares
RF- #29	Evaluar método tasa interna de retorno con números difusos triangulares
HU#8	**Generar reporte**
RF- #30	Generar reporte de un proyecto
RF- #31	Generar reporte de todos los proyectos
HU#9	**Mostrar análisis de factibilidad**
RF- #32	Mostrar análisis de factibilidad económica

RF- #33	Mostrar análisis de factibilidad del flujo de caja
RF- #34	Mostrar análisis de factibilidad del valor actual neto
RF- #35	Mostrar análisis de factibilidad de la tasa interna de retorno
RF- #36	Mostrar análisis de factibilidad del período de recuperación

Tabla 8: Requisitos funcionales del usuario.

2.3.3 Requisitos no funcionales del software

Según (Sommerville, 2005) un requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad (Sommerville, 2005). En la tabla 9 se muestran los requisitos no funcionales identificados.

Código	Clasificación
	Interfaces de usuario
RNF- #01	La interfaz de usuario en cada una de las páginas debe ser sencilla, intuitiva, que le permita al usuario realizar las acciones en un tiempo menor a 5 segundos.
RNF- #02	Las interfaces tendrán menús que le permitan al usuario acceder desde cualquier lugar de la aplicación a otro con la menor cantidad de acciones.
	Usabilidad
RNF- #03	La herramienta debe presentar un acceso fácil y rápido, para facilitar su uso por usuarios con pocos conocimientos en el campo de la informática.
	Fiabilidad
RNF- #04	Se restringirá el acceso por roles y usuarios a la aplicación, para asegurar la consistencia y fiabilidad de los datos.
	Seguridad
RNF- #05	Para asegurar la integridad y confiabilidad de los datos que se procesan, la aplicación deberá estar protegida mediante una política de roles que no permitan el acceso no autorizado.
RNF- #06	Cada usuario tendrá acceso y se le mostrarán las funcionalidades a las cuales está autorizado a acceder.
	Software
RNF- #07	Para el funcionamiento en la PC cliente será necesario Firefox 26.0 o superior.
	Hardware
RNF- #8	El servidor para aplicaciones web deberá tener las siguientes características mínimas: Procesador Intel Pentium Dual Core a 2.0 GHz, equivalente o superior.
RNF- #9	El servidor para la base de datos deberá tener las siguientes características mínimas: Procesador Intel Pentium Dual Core a 2.0 GHz, equivalente o superior y 1 GB de memoria RAM

	Portabilidad
RNF- #10	El sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.
	Restricciones de diseño e implementación
RNF- #11	Para el desarrollo de la aplicación se deberá utilizar un conjunto de herramientas y tecnologías libres. Como lenguaje de programación se utilizará PHP con el marco de trabajo Symfony2, el IDE Netbeans 8.0, servidor web apache 2.0 y gestor de bases de datos PostgreSQL 9.2. Para regir el proceso de desarrollo de software se deberá utilizar la metodología XP y para el modelado Visual Paradigm. Además como sistema operativo se utilizará Windows 8 o una distribución de GNU/Linux.

Tabla 9: Requisitos no funcionales.

2.3.4 Estimación de esfuerzo por historias de usuario

Luego de terminadas las historias de usuario entre el cliente y el equipo de desarrollo, se pasó a la estimación de esfuerzo por cada una de ellas. Estas estimaciones permiten tener una medida de la velocidad del proyecto y ofrecen una guía a la cual ajustarse. El equipo de desarrollo según la complejidad de implementación definió el tiempo tardado en desarrollar cada historia de usuario. La estimación realizada se encuentra entre una y cuatro semanas. Las estimaciones realizadas a las historias de usuario se muestran a continuación:

Historia de Usuario	Puntos de estimación (semanas)
Gestionar usuario	1
Gestionar proyecto	1
Gestionar información económica	2
Gestionar proceso de evaluación	2
Gestionar proyectos a evaluar durante una evaluación	0.5
Gestionar métodos a evaluar durante una evaluación	0.5
Realizar análisis de factibilidad económica	4
Generar reporte	2
Mostrar análisis de factibilidad económica	1

Tabla 10: Estimación de esfuerzo por historia de usuario

2.3.5 Plan de iteraciones

Para cada entrega fueron escogidas algunas historias de usuario, teniendo en cuenta el orden definido por el cliente y el equipo de desarrollo. Este plan define las historias de usuario que deben ser implementadas en cada iteración y las fechas de liberación. A continuación se muestran 2 iteraciones y las historias de usuario por cada una de ellas.

Primera: en esta iteración se van a implementar las historias de usuario que tienen una prioridad alta para el negocio.

Segunda: en esta iteración se van a implementar las historias de usuario que tienen una prioridad media para el negocio.

Iteración	Orden de la HU a implementar	Duración de cada HU (Días)	Duración total (semanas)
1ra	Gestionar usuario	4	10
	Gestionar proyectos	9	
	Gestionar información económica	5	
	Gestionar proceso de evaluación	6	
	Gestionar proyectos a evaluar durante una evaluación	3	
	Gestionar métodos a evaluar durante una evaluación	3	
	Realizar análisis de factibilidad económica	20	
2da	Generar reporte	10	4,0
	Mostrar análisis de factibilidad económica	10	
Total		70	14,0

Tabla 11: Plan de iteraciones por historias de usuario

2.3.6 Plan de entregas

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas (Joskowicz, 2008). En el momento en que culmina la elaboración de las HU, se inicia el proceso de creación de un plan de entrega. El cual tiene como objetivo fundamental la obtención por parte de los programadores de una estimación detallada del período de tiempo que deben tener en cuenta para la implementación. A continuación se presenta un plan de entregas por módulo:

Historia de usuario	Iteración	Entrega
Gestionar usuario	1	29 de mayo
Gestionar proyectos		
Gestionar información económica		
Realizar análisis de factibilidad económica		
Gestionar proceso de evaluación		
Gestionar métodos a evaluar durante una evaluación		
Gestionar proyectos a evaluar durante una evaluación		
Generar reporte	2	18 de junio
Mostrar análisis de factibilidad económica		

Tabla 12: Plan de entregas de las iteraciones.

2.4 Fase de diseño

La metodología de desarrollo XP plantea prácticas especializadas que accionan directamente en la realización del diseño para lograr un sistema robusto y reutilizable. Los conceptos más importantes de diseño en esta metodología son: simplicidad, soluciones, recodificación y metáforas (Joskowicz, 2008).

2.4.1 Tarjetas CRC

Las tarjetas Clase-Responsabilidad-Colaboración (CRC) permiten ver las clases no como un depósito de datos, sino que permiten conocer el comportamiento de cada una en un alto nivel (Joskowicz, 2008). La metodología XP estipula su uso como un artefacto obligatorio durante el desarrollo de un proyecto, debido a los beneficios que aportan a los desarrolladores. A continuación se muestra la tarjeta de una de las clases que responden a funcionalidades de alta prioridad en el sistema. El resto de las tarjetas pueden visualizarse en el **Anexo 2**.

Clase: proyecto	
Tipo de clase: Entidad	
Nombre:	**Descripción:**
Responsabilidades	**Colaborador**
Obtener nombre oficial.	
Modificar nombre oficial.	
Obtener fecha inicio del proyecto.	
Modificar fecha inicio del proyecto.	
Obtener fecha fin del proyecto.	
Modificar fecha fin del proyecto.	
Obtener tiempo estimado en horas.	
Modificar tiempo estimado en horas.	
Obtener alcance del proyecto.	
Modificar alcance del proyecto.	
Obtener objetivo general del proyecto.	
Modificar objetivo general del proyecto.	
Obtener objetivos específicos del proyecto.	
Modificar objetivos específicos del proyecto.	
Obtener estado del proyecto.	
Modificar estado del proyecto.	
Obtener impacto del proyecto.	
Modificar impacto del proyecto.	
Obtener monto total contrato USD del proyecto.	
Modificar monto total contrato USD del proyecto.	
Obtener monto total contrato CUC del proyecto.	
Modificar monto total contrato CUC del proyecto.	
Obtener monto total contrato CUP del proyecto.	
Modificar monto total contrato CUP del proyecto.	
Obtener monto ejecutor cubano USD del proyecto.	
Modificar monto ejecutor cubano USD del proyecto.	

Obtener monto ejecutor cubano CUC del proyecto.	
Modificar monto ejecutor cubano CUC del proyecto.	
Obtener monto ejecutor cubano CUP del proyecto.	
Modificar monto ejecutor cubano CUP del proyecto.	
Obtener presupuesto gasto USD del proyecto.	
Modificar presupuesto gasto USD del proyecto.	
Obtener presupuesto gasto CUC del proyecto.	
Modificar presupuesto gasto CUC del proyecto.	
Obtener presupuesto gasto CUP del proyecto.	
Modificar presupuesto gasto CUP del proyecto.	
Obtener cantidad de períodos del proyecto.	
Modificar cantidad de períodos del proyecto.	
Obtener tasa de descuento del proyecto.	
Modificar tasa de descuento del proyecto.	
Obtener responsable desarrollador.	
Modificar responsable desarrollador.	
Obtener cargo desarrollador.	
Modificar cargo desarrollador.	
Obtener teléfono desarrollador.	
Modificar teléfono desarrollador.	
Obtener responsable cliente.	
Modificar responsable cliente.	
Obtener cargo cliente.	
Modificar cargo cliente.	
Obtener teléfono cliente.	
Modificar teléfono cliente.	
Obtener correo cliente.	
Modificar correo cliente.	
Obtener tiempo período.	Período
Modificar tiempo período.	Período

Tabla 13: Tarjeta CRC para la clase proyecto.

2.4.2 Patrones de diseño

En la implementación del sistema se utilizaron varios patrones de diseño, a continuación se mencionan y se explica el modo en que fueron utilizados. Los mismos son propuestos por Symfony2, que fue el marco de trabajo utilizado.

Patrones GRASP

Creador: en las clases controladoras se encuentran todas las acciones definidas para el módulo evaluacionEconomica. Con el método createAction () se crean los objetos de las clases que representan las entidades, evidenciando de este modo que las clases controladoras son creadoras de dichas entidades. A continuación se muestra la clase controladora proyectoController:

```php
class ProyectoController extends Controller {

    public function createAction(Request $request) {
        $entity = new Proyecto();
        $form = $this->createForm(new ProyectoType(), $entity);
        $form->bind($request);
        $em = $this->getDoctrine()->getManager();
        $idproyecto = $em->getRepository('EvaluacionBundle:Proyecto')->selectId($entity->getNombreOficial());
        if ($idproyecto != null) {
            $this->get('session')->getFlashBag()->add('existe', null);
            return $this->redirect($this->generateUrl('proyecto_new'));
        }
        if ($this->comprobarFecha($entity->getCantPeriodos(), $entity->getFechaInicio(), $entity->getFechaFin(), $entity->getTiempoPeriodo()) == false) {
            $this->get('session')->getFlashBag()->add('fecha', null);
            return $this->redirect($this->generateUrl('proyecto_new'));
        }
        $entity->setTiempoEstimadoHH($this->calcularTiempoHH($entity->getCantPeriodos(), $entity->getTiempoPeriodo()));
        if ($form->isValid()) {
            $entity->setEstado("Iniciado");
            $em->persist($entity);
            $em->flush();
            $this->get('session')->getFlashBag()->add('add', null);

            return $this->redirect($this->generateUrl('proyecto'));
        }

        return $this->render('EvaluacionBundle:Proyecto:new.html.twig', array(
                    'entity' => $entity,
                    'form' => $form->createView(),
          ));
        }
```

Ilustración 6: Patrón creador.

Experto: se evidencia en las clases entidades, ya que las responsabilidades se le asignan a las clases con la información necesaria para cumplir con dichas responsabilidades.

Bajo Acoplamiento: impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que nos lleve a los resultados negativos que puede producir un acoplamiento alto. El patrón bajo acoplamiento se ve reflejado en las clases controladoras, estas heredan solamente de Controller.php para garantizar que exista la menor cantidad de dependencias entre clases posibles.

```php
class ProyectoController extends Controller {

    /**
     * Lists all Proyecto entities.
     *
     */
    public function indexAction() {
        $em = $this->getDoctrine()->getManager();

        $entities = $em->getRepository('EvaluacionBundle:Proyecto')->findAll();

        return $this->render('EvaluacionBundle:Proyecto:index.html.twig', array('entities' => $entities,));
    }

```

Ilustración 7: Patrón bajo acoplamiento.

Alta Cohesión: Symfony2 permite asignar responsabilidades con una alta cohesión ya que los controladores definen las acciones para las plantillas y colaboran con otras clases para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador: la responsabilidad de controlar el flujo de eventos del sistema se debe asignar a clases específicas. En la aplicación es utilizado a través del controlador frontal app.php, siendo el encargado de manejar todas las peticiones web. El controlador app.php es el punto de entrada único de toda la aplicación.

Patrones GOF

Creacionales:

1. **Método fábrica:** define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
2. **Inicialización perezosa:** es la táctica de retrasar la creación de un objeto, el cálculo de un valor, o algún otro proceso costoso hasta la primera vez que es realmente necesario.

Estructurales:

3. **Adaptador:** se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda.
4. **Compuesto:** sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol.
5. **Decorador:** responde a la necesidad de añadir dinámicamente funcionalidad a un objeto.

Comportamiento:

6. **Mediador:** define un objeto que encapsula cómo interactúan un conjunto de objetos.
7. **Iterador:** declara los métodos necesarios para acceder secuencialmente a un grupo de objetos de una colección.
8. **Método Plantilla:** se caracteriza por la definición, dentro de una operación de una superclase, de los pasos de un algoritmo, de forma que todos o parte de estos pasos son redefinidos en las subclases herederas de la citada superclase.
9. **Estrategia:** determina cómo se debe realizar el intercambio de mensajes entre diferentes objetos para resolver una tarea.

2.4.3 Patrón Arquitectónico

Para la implementación de la herramienta se utilizó el patrón arquitectónico Modelo-Vista-Controlador. Este estilo establece que los componentes de un sistema de software deben organizarse en tres capas distintas según su misión (Symfony, 2012):

- **El modelo:** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- **La vista:** transforma el modelo en una página web que permite al usuario interactuar con ella.
- **El controlador:** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

En la siguiente ilustración se puede presenciar el uso de este patrón en la herramienta a desarrollar:

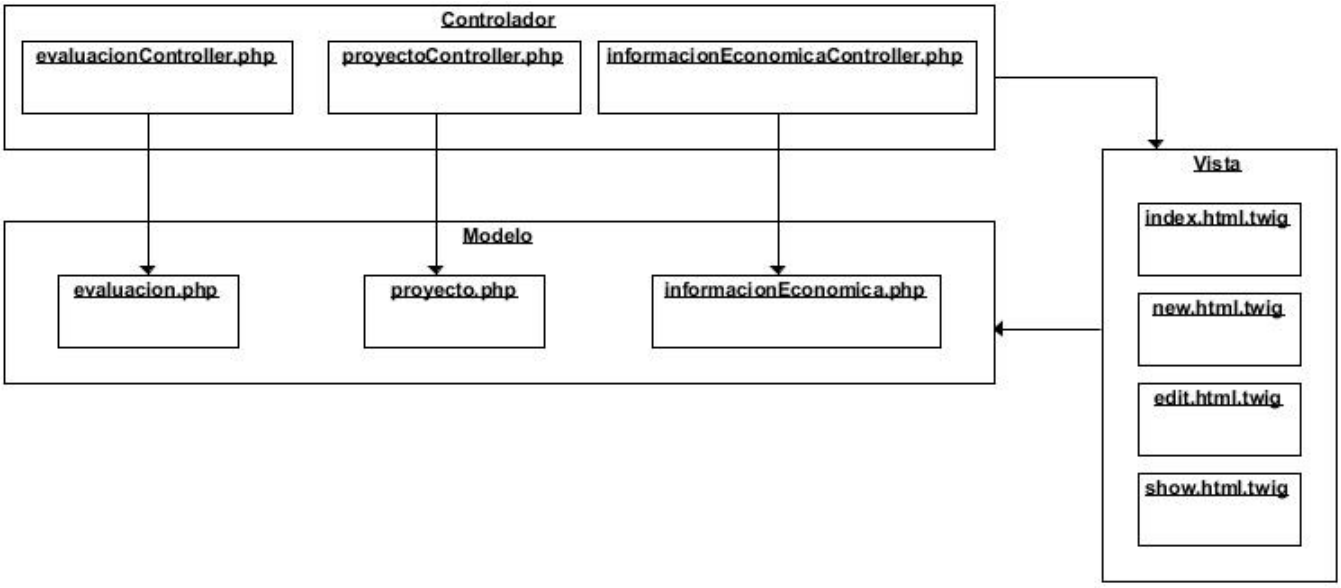


Ilustración 8: Patrón arquitectónico MVC.

2.4.4 Modelo de Datos

El equipo de desarrollo definió desarrollar el modelo de datos, este tiene como propósito "garantizar que los datos persistentes sean almacenados coherente y eficazmente y definir el comportamiento que debe ser

implementado en la base de datos" (Jacobson, 1999). El modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento (Date, 2001).

El modelo de datos correspondiente a la herramienta que se desarrolla está compuesto por 10 tablas, siendo las más significativas para realizar el análisis de factibilidad económica: proyecto, informacionEconomica y evaluacion.

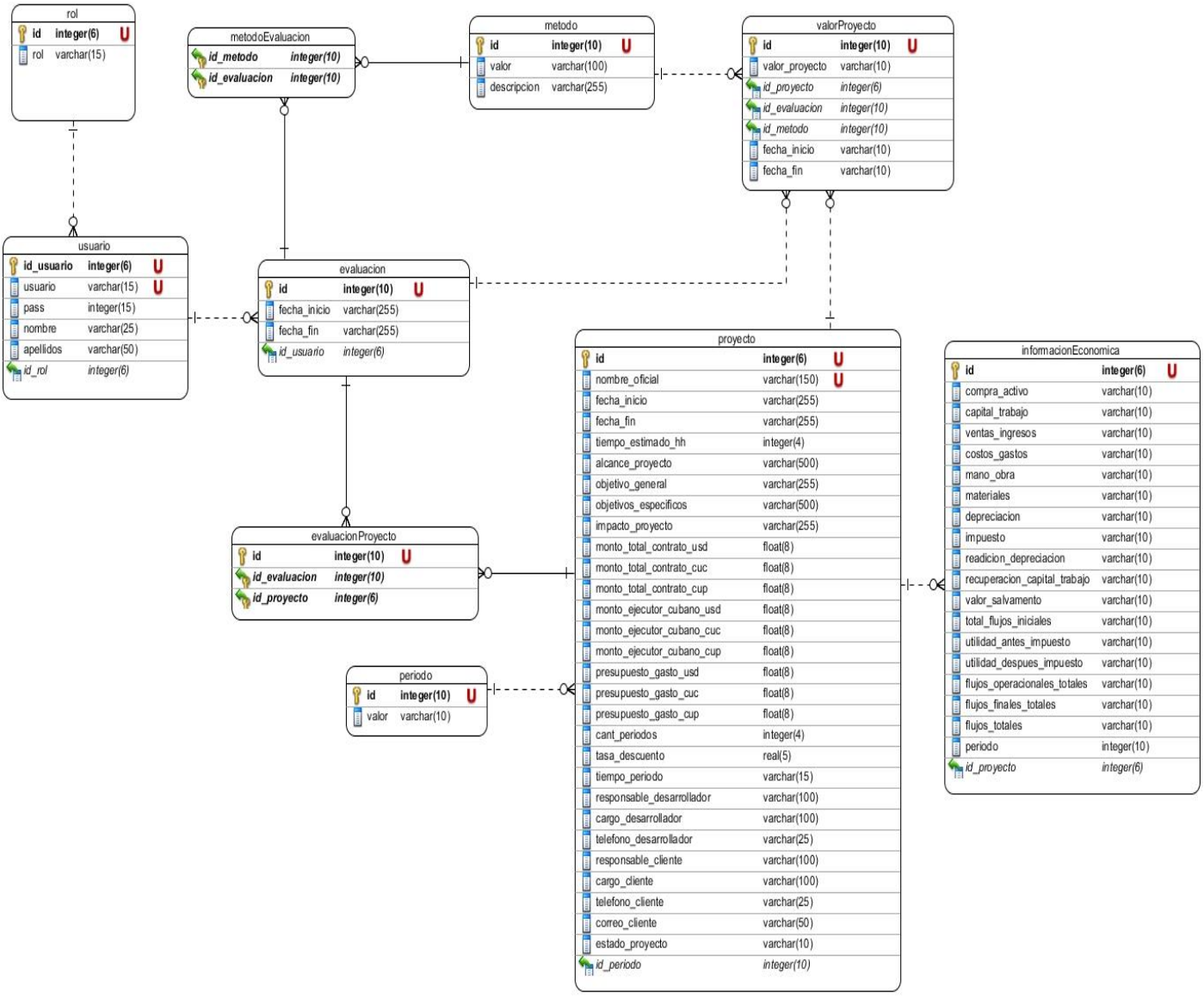


Ilustración 9: Modelo de datos.

2.5 Métricas para validar el diseño

Esta métrica se determina por el número total de operaciones que están encapsuladas dentro de la clase. Grandes valores de esta medida muestran que la clase puede tener demasiada responsabilidad, lo cual reducirá la reusabilidad de la misma y complicará su implementación. Por otro lado, en cuanto menor sea el

valor medio para el tamaño más probable es que las clases tengan menos responsabilidad y complejidad y más nivel de reutilización. Los resultados para la métrica TOC pueden ser consultados en el **Anexo 3**. Para determinar el valor de los atributos, se calcula el promedio de procedimientos por clases, en este caso se obtuvo como resultado 12,86.

2.5.1 Tamaño Operacional de Clases

Esta métrica se determina por el número total de operaciones que están encapsuladas dentro de la clase. Grandes valores de esta medida muestran que la clase puede tener demasiada responsabilidad, lo cual reducirá la reusabilidad de la misma y complicará su implementación. Por otro lado, en cuanto menor sea el valor medio para el tamaño más probable es que las clases tengan menos responsabilidad y complejidad y más nivel de reutilización. Los resultados para la métrica TOC pueden ser consultados en el **Anexo 3**. Para determinar el valor de los atributos, se calcula el promedio de procedimientos por clases, en este caso se obtuvo como resultado 12,86.

En el siguiente gráfico se muestra la representación de los resultados obtenidos agrupados en los intervalos definidos. El mismo refleja que la mayoría de las clases tienen de 0 a 8 procedimientos. Esto demuestra que el funcionamiento general del sistema está distribuido equitativamente entre las diferentes clases.

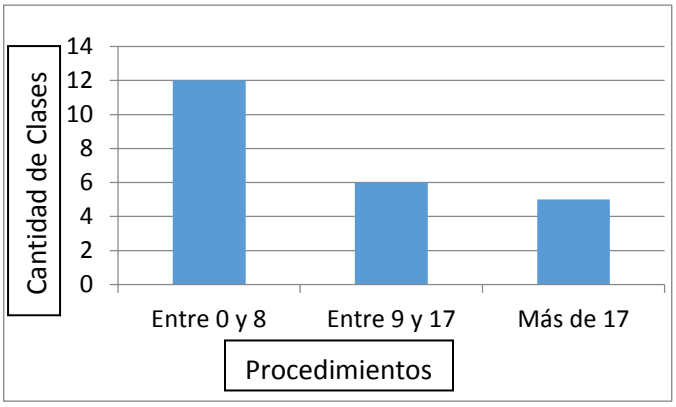


Ilustración 10: Representación de la métrica TOC

En la siguiente ilustración se muestran los resultados obtenidos en porciento agrupados en los intervalos definidos.

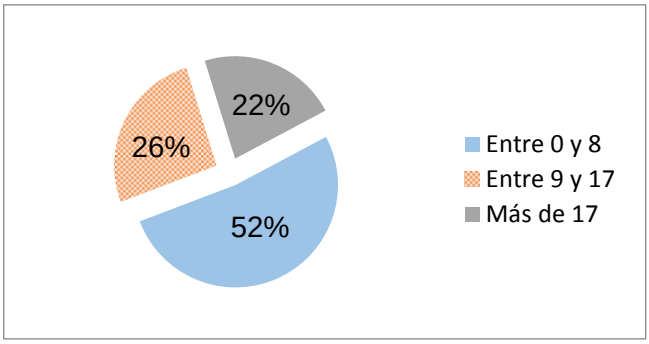


Ilustración 11: Representación en porciento obtenido en la evaluación de la métrica TOC

En la siguiente ilustración se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. Este gráfico muestra un resultado satisfactorio pues el 52% de las clases poseen una baja responsabilidad. Esta característica permite que en caso de fallos, como la responsabilidad está distribuida de forma equilibrada ninguna clase es demasiado crítica para dejar al sistema fuera de servicio.

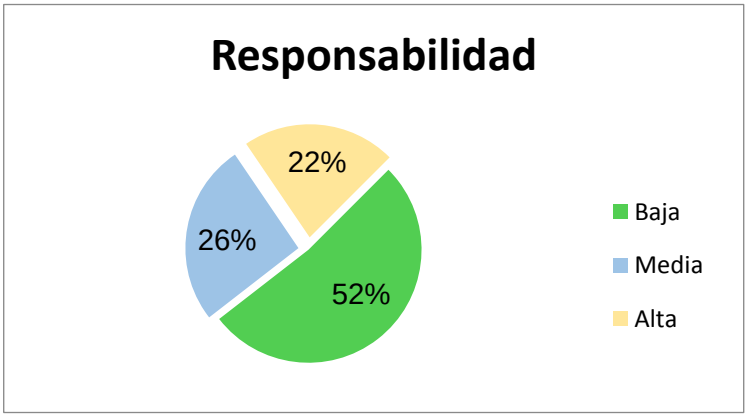


Ilustración 12: Resultado de la métrica TOC para el atributo responsabilidad

En la siguiente ilustración se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo complejidad de implementación. Este gráfico muestra un resultado satisfactorio pues el 52% de las clases poseen una baja complejidad de implementación. Esta característica permite mejorar el mantenimiento y soporte de estas clases. Además va a existir un bajo grado de esfuerzo para la implementación de la herramienta.

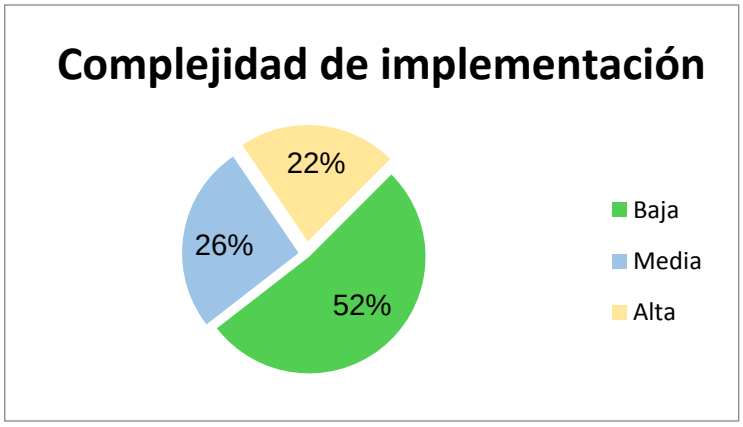


Ilustración 13: Resultado de la métrica TOC para el atributo complejidad de implementación

En la siguiente ilustración se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. El diseño del sistema tiene un grado de eficiencia aceptable pues solamente el 22% del total de las clases poseen una baja reutilización. Permitiendo que la mayor cantidad de clases sean reutilizables.

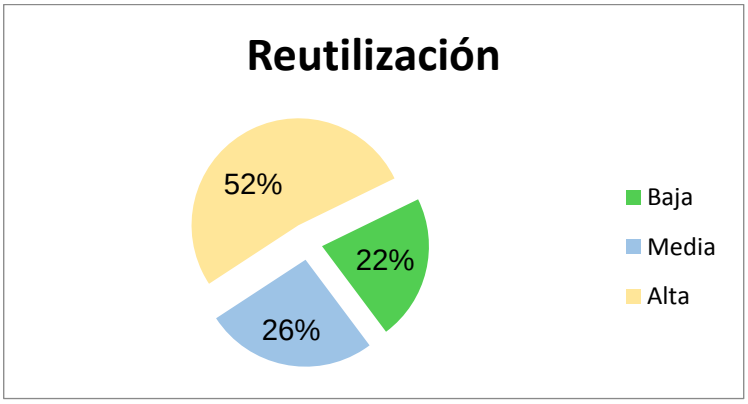


Ilustración 14: Resultado de la métrica TOC para el atributo reutilización

Tras un análisis de los resultados arrojados por la evaluación bajo los instrumentos de medición de la métrica Tamaño Operacional de Clase, se demuestra que se alcanzaron buenos valores para cada uno de los atributos de calidad evaluados, puesto que, como se puede observar, el 52% de las clases del software contienen un número menor que el promedio de procedimientos por clases, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases y hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables. Solamente un 22% de las clases tienen pocas posibilidades de reutilización y una gran complejidad de implementación. Estos valores demuestran que los indicadores de reutilización, complejidad y responsabilidad son aceptables.

2.5.2 Relaciones entre Clases

Esta métrica se determina por la cantidad de relaciones existentes entre las clases contenidas en el diseño. El número de dependencias es directamente proporcional al nivel de acoplamiento, a la complejidad del mantenimiento y a la cantidad de pruebas a realizar sobre las clases, y es inversamente proporcional al grado de reutilización de las mismas. La aplicación de esta métrica evidencia que el diseño del sistema es aceptable con respecto a todos los atributos de calidad afectados, arrojando resultados positivos. Los resultados para la métrica RC pueden ser consultados en el **Anexo 4**. El promedio utilizado para evaluar el criterio es el resultado del cálculo del promedio de la cantidad de relaciones entre clases, en este caso el promedio es 1,82.

En la siguiente ilustración se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento. Se evidencia un bajo acoplamiento entre las clases pues el 56% de las clases tienen una dependencia con otra. Este resultado es muy favorable para el diseño del sistema pues al existir poca dependencia entre las clases aumenta el grado de reutilización del sistema.

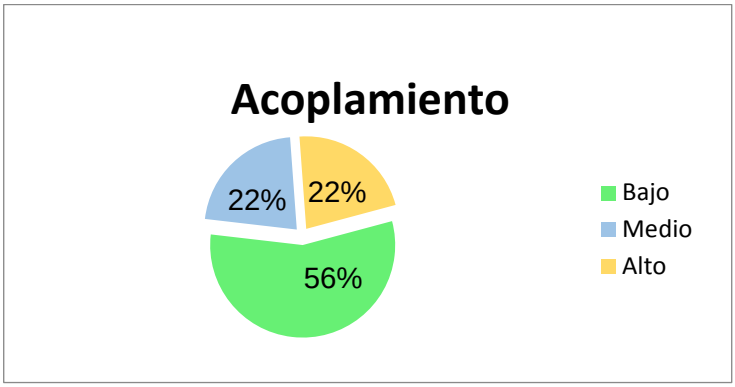


Ilustración 15: Resultado de la métrica RC para el atributo acoplamiento

En la siguiente ilustración se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento. Este resultado es muy favorable para el diseño del sistema pues existirá un bajo nivel de esfuerzo para la realización del mantenimiento.

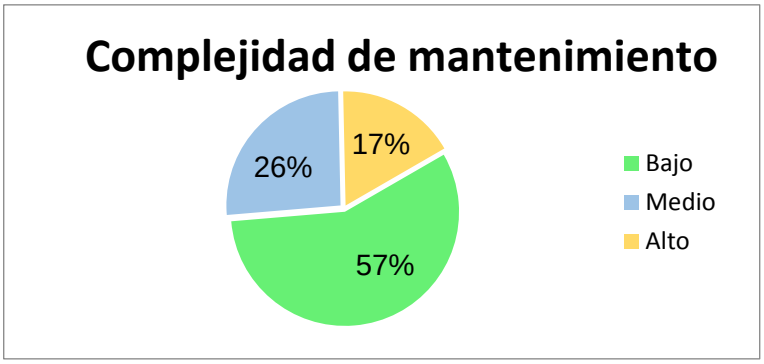


Ilustración 16: Resultado de la métrica RC para el atributo complejidad de mantenimiento

En la siguiente ilustración se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo reutilización. El 57% de las clases poseen una alta reutilización lo que es un factor fundamental que debe ser tenido en cuenta en el desarrollo de la herramienta.

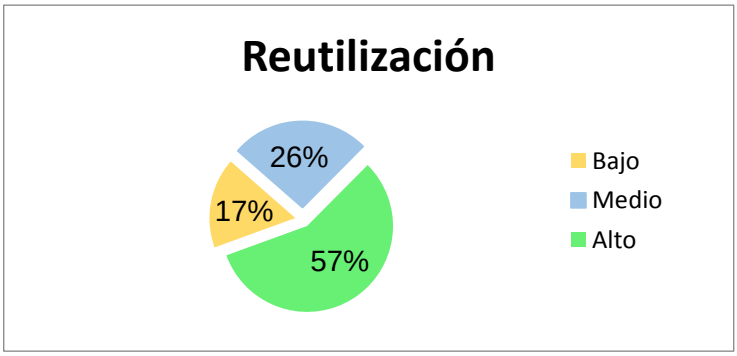


Ilustración 17: Resultado de la métrica RC para el atributo reutilización

En la siguiente ilustración se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas. Se evidencia que el 57% de las clases poseen baja cantidad

de pruebas, siendo esto un resultado favorable debido al nivel de esfuerzo que tendrá que realizar el equipo de desarrollo.

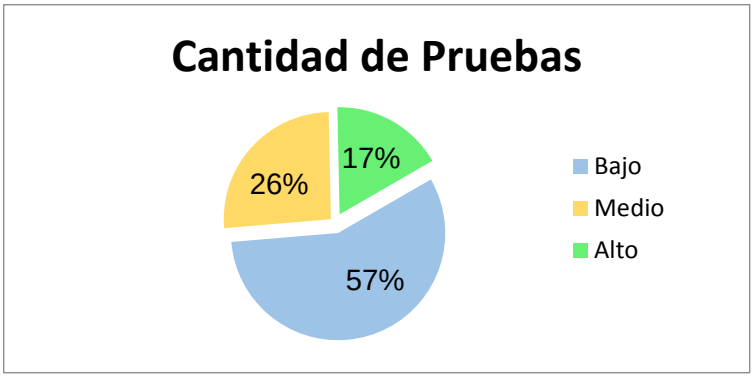


Ilustración 18: Resultado de la métrica RC para el atributo cantidad de pruebas

Realizando un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del sistema para evaluar factibilidad económica de proyectos de software tiene una calidad aceptable. La complejidad de mantenimiento y la cantidad de pruebas son bajas a un 57%, lo que representa valores favorables para el diseño realizado. Así mismo, existe un alto grado de reutilización al 57%, comportamiento también favorable para este atributo de calidad. El 56% de las clases que conforman el sistema poseen menos de dos dependencias con otras clases. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 56% de las clases el nivel de acoplamiento es mínimo.

2.6 Conclusiones del capítulo

En este capítulo se arribaron a las siguientes conclusiones:

- Se confeccionó el modelo conceptual para una representación visual del dominio del problema.
- La elaboración de los artefactos generados en la fase de análisis y diseño permitieron un mayor entendimiento de la solución y tener una visión de la velocidad del proyecto.
- Se fundamentó la elección del patrón Modelo-Vista-Controlador para la arquitectura del sistema.
- La validación del diseño permitió mediante las métricas TOC y RC permitieron asegurar un diseño satisfactorio.

Capítulo 3: Implementación y pruebas

3.1 Introducción

En el presente capítulo se describe de forma general las fases de implementación y pruebas de la metodología seleccionada. En la fase de implementación se generan las tareas de ingeniería por iteraciones y las tareas de ingeniería detalladas para una mejor implementación de la solución. Se describen los estándares de codificación a utilizar, así como el tratamiento de la seguridad propuesta por el marco de trabajo utilizado. Se describen las pruebas realizadas al sistema, que tienen como objetivo garantizar la calidad del mismo y el total cumplimiento de los requisitos establecidos con el cliente.

3.2 Fase de implementación

En esta fase se realiza la implementación de las historias de usuario que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar alguna modificación. Como parte de este plan se crean las tareas de ingeniería por iteraciones y las tareas de ingeniería detalladas para ayudar a organizar la implementación de las historias de usuario. A continuación se detallan las historias de usuario por iteraciones.

3.2.1 Tareas de ingeniería por iteraciones

Cada historia de usuario está compuesta por una o varias tareas de ingeniería, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación (Beck, 2000). A continuación se detallan las historias de usuario:

Historias de Usuario	Tareas de Ingeniería por Historias de Usuario
Gestionar usuario	➢ Elaborar prototipo de interfaz. ➢ Crear formulario para adicionar usuario. ➢ Agregar información del usuario. ➢ Modificar y eliminar usuario. ➢ Listar usuarios.
Gestionar proyecto	➢ Elaborar prototipos de interfaz. ➢ Crear formulario para el proyecto. ➢ Agregar información del proyecto. ➢ Modificar y eliminar proyecto. ➢ Listar proyectos.
Gestionar información económica	➢ Elaborar prototipo de interfaz. ➢ Crear formulario para adicionar información económica. ➢ Modificar y eliminar información económica.

Gestionar proceso de evaluación	➢ Elaborar prototipo de interfaz. ➢ Crear formularios para crear un proceso de evaluación.
Gestionar método a evaluar durante una evaluación	➢ Mostrar interfaz con los métodos a evaluar.
Realizar análisis de factibilidad económica	➢ Aplicar los métodos económicos. ➢ Mostrar resultados de la evaluación
Gestionar reportes	➢ Elaborar prototipo de interfaz ➢ Realizar reportes.
Mostrar análisis de factibilidad económica	➢ Elaborar prototipo de interfaz. ➢ Crear tabla para mostrar análisis de factibilidad.

Tabla 14: Tareas de ingeniería por iteraciones por historias de usuario

3.2.2 Tareas de ingeniería detalladas

Luego del equipo de desarrollo desglosar en tareas las historias de usuario, se pasa a la confección de las tareas de ingenierías detalladas, siendo este elemento de gran ayuda a los programadores a la hora de comenzar la implementación.

Tarea:	
Número de tarea: # 1.	**Historia de Usuario:** Gestionar proyecto.
Nombre de la tarea: Elaborar prototipo de interfaz.	
Tipo de tarea: desarrollo.	**Puntos estimados:** 1.
Fecha inicio: 20-02-2015.	**Fecha fin:** 20-02-2015.
Programador responsable: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	
Descripción: Se elabora el prototipo de interfaz como apoyo para la posterior implementación de la funcionalidad gestionar proyecto.	

Tabla 15: Descripción de la tarea de ingeniería detallada elaborar prototipo de interfaz.

Tarea:	
Número de tarea: # 2.	**Historia de Usuario:** Gestionar proyecto.
Nombre de la tarea: Crear formulario para el proyecto.	
Tipo de tarea: desarrollo.	**Puntos estimados:** 2.
Fecha inicio: 23-02-2015.	**Fecha fin:** 24-02-2015.
Programador responsable: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	
Descripción: se debe crear un formulario con todos los campos para recoger la información necesaria de un proyecto.	

Tabla 16: Descripción de la tarea de ingeniería detallada crear formulario para el proyecto.

Tarea:	
Número de tarea: # 3.	**Historia de Usuario:** Gestionar proyecto.
Nombre de la tarea: Adicionar proyecto.	
Tipo de tarea: desarrollo.	**Puntos estimados:** 1.
Fecha inicio: 25-02-2015.	**Fecha fin:** 25-02-2015.
Programador responsable: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	
Descripción: se muestra un formulario para introducir los datos del proyecto, se adiciona los datos del proyecto.	

Tabla 17: Descripción de la tarea de ingeniería detallada adicionar proyecto.

Tarea:	
Número de tarea: # 4.	**Historia de Usuario:** Gestionar proyecto.
Nombre de la tarea: Modificar y eliminar proyecto.	
Tipo de tarea: desarrollo.	**Puntos estimados:** 1.
Fecha inicio: 26-02-2015.	**Fecha fin:** 26-02-2015.
Programador responsable: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	
Descripción: se muestra un formulario con los datos del proyecto a modificar, se modifican los datos del proyecto. Se elimina el proyecto seleccionado.	

Tabla 18: Descripción de la tarea de ingeniería detallada modificar y eliminar proyecto.

Tarea:	
Número de tarea: # 5.	**Historia de Usuario:** Gestionar proyecto
Nombre de la tarea: Listar proyecto.	
Tipo de tarea: desarrollo.	**Puntos estimados:** 1.
Fecha inicio: 26-02-2015.	**Fecha fin:** 26-02-2015
Programador responsable: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	
Descripción: se muestra en una tabla todos los proyectos que han sido adicionados con anterioridad.	

Tabla 19: Descripción de la tarea de ingeniería detallada listar proyectos.

3.3 Estándares de codificación

XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación (Joskowicz, 2008). En el caso de la herramienta que se desarrolla, los estándares utilizados son:

- La indentación debe ser a cuatro espacios sin caracteres de tabulación.
- Todas las clases terminarán con una etiqueta "**;**", no con la etiqueta usual de cierre "**>**".
- Las clases y métodos comenzarán con letra inicial minúscula, en el caso de ser un nombre compuesto por más de una palabra no serán separadas.

```
public function selectPeriodo($id) {

    $em=$this->getEntityManager();

    $sql = "select p.tiempoperiodo   from proyecto p where p.id='$id'";

    $query=$em->getConnection()->executeQuery($sql);


    return $query->fetchAll();
}
```

Ilustración 19: Utilización del camelCase en la funcionalidad selectPeríodo ($id).

- Todas las interfaces utilizarán el sufijo **interface**.

```
22  interface UserInterface {
24      public function getRoles();
26      public function getPassword();
28      public function getSalt();
30      public function getUsername();
32      public function eraseCredentials();
33  }
```

Ilustración 20: Uso del sufijo interface en las vistas.

- Todas las clases empezarán con el sufijo **namespace**.

```
namespace TesisUO\SeguridadBundle\Entity;

use Symfony\Component\Validator\Constraints as Assert;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\UserInterface;
```

Ilustración 21: Uso del sufijo namespace.

El uso de los estándares de codificación permitirá tener un código más organizado y entendible para el equipo de trabajo. Garantizan la mantenibilidad del código, permite un mejor trabajo a la hora de realizar algunas de las siguientes acciones: modificar o añadir nuevas funcionalidades, mejorar el rendimiento y depurar errores.

3.4 Seguridad del sistema

"La seguridad es un proceso de dos etapas, cuyo objetivo es evitar que un usuario acceda a un recurso al cual no debería tener acceso. Symfony controla su seguridad a través de la autenticación y la autorización. La autenticación es identificar quién es el usuario, obligándolo a presentar algún tipo de identificación, significando

que el sistema está tratando de verificar qué eres quién dices ser. Para la autenticación utiliza distintos métodos, ejemplos de estos son, formularios de inicio de sesión, autenticación http y método de autenticación personalizado. La autorización es comprobar si tienes suficientes privilegios para acceder a un recurso y realizar una determinada acción" (Pacheco, 2013).

Cortafuegos (autenticación)

"Cuando un usuario hace una petición a una URL que está protegida por un cortafuego, se activa el sistema de seguridad. El trabajo del cortafuego es determinar si el usuario necesita estar autenticado, y si lo hace, enviar una respuesta al usuario para iniciar el proceso de autenticación. Un cortafuego se activa cuando la URL de una petición entrante concuerda con el patrón de la expresión regular configurada en el valor **config** del cortafuego" (Pacheco, 2013).

Control de acceso (autorización)

"Los roles son la base para la mayor parte de la autorización, el usuario puede acceder solo al módulo que tiene permiso. Symfony define el trabajo con rutas para una mejor seguridad del sistema. El sistema de seguridad automáticamente comprobará las credenciales del usuario, o bien el usuario es autenticado o se enviará al formulario de acceso donde se puede mostrar el error. El proceso de autorización tiene dos lados diferentes, el usuario tiene un conjunto de roles específicos y un recurso requiere un rol específico a fin de tener acceso" (Pacheco, 2013).

La seguridad del sistema es un tema fundamental, debido a que no se permite el acceso no autorizado a la herramienta para evitar pérdidas de datos. Además se garantiza mediante la políticas de roles que cada usuario solo pueda tener acceso a las funcionalidades que le corresponden.

3.5 Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos *(Visconti, 2012)*. En la herramienta, el usuario accede, desde su estación de trabajo al sistema que se encuentra instalado en el servidor de aplicaciones web. Luego el servidor de aplicaciones se conecta al servidor de base de datos mediante el protocolo TCP[18] para realizar las consultas y obtener los resultados. A continuación se muestra el diagrama de despliegue realizado por el equipo de desarrollo:

[18] **TCP:** Acrónimo en inglés de Transmission Control Protocol.

Ilustración 22: Diagrama de despliegue.

3.6 Fase de pruebas

Una de las principales fortalezas de la metodología de desarrollo XP es el proceso de prueba, el cual se realiza de manera continua para asegurar durante todo el proceso de desarrollo, el éxito del producto que se está elaborando. Esto permite elaborar un software de más calidad ya que los errores son detectados en un plazo de tiempo corto y se corrigen de una manera más sencilla (Gutiérrez, 2006).

XP divide las pruebas de software o de sistema en dos grupos, las pruebas unitarias y las pruebas de aceptación. Las pruebas unitarias son las encargadas de verificar el código y estas son diseñadas por los programadores. Mientras que las pruebas de aceptación o pruebas funcionales están destinadas a evaluar si al final de una iteración se consiguió la funcionalidad que se esperaba y que esta esté en función de los requisitos establecidos inicialmente, estas pruebas usualmente son diseñadas por el usuario o cliente final (Gutiérrez, 2006).

3.6.1 Pruebas de caja blanca

Las pruebas unitarias o pruebas de caja blanca se basa en realizar pruebas al código del sistema. Estas pruebas no se le pueden realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial. Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos (Pressman, 2005). A continuación se enumeran las sentencias de código del método updateAction (Request $request, $id), método utilizado en la solución para la modificación de un proyecto.

```
161     public function updateAction(Request $request, $id) {
162         $em = $this->getDoctrine()->getManager(); //1
163         $entity = $em->getRepository('EvaluacionBundle:Proyecto')->find($id); //1
164 
165         if (!$entity) { //2
166             $var = $this->createNotFoundException('No se ha podido encontrar la entidad Proyecto.'); //3
167         } else { //4
168             $deleteForm = $this->createDeleteForm($id); //5
169             $editForm = $this->createForm(new ProyectoType(), $entity); //5
170             $editForm->bind($request); //5
171 
172             if ($editForm->isValid()) { //6
173                 $em->persist($entity); //7
174                 $em->flush(); //7
175                 $var = $this->redirect($this->generateUrl('proyecto', array('id' => $id))); //7
176             } else { //8
177                 $var = $this->render('EvaluacionBundle:Proyecto:edit.html.twig', array('entity' => $entity,
178                     'edit_form' => $editForm->createView(),
179                     'delete_form' => $deleteForm->createView(),)); //9
180             }
181         }
182 
183         return $var; //10
184     }
```

Ilustración 23: Código fuente de la funcionalidad updateAction (Request $request, $id)

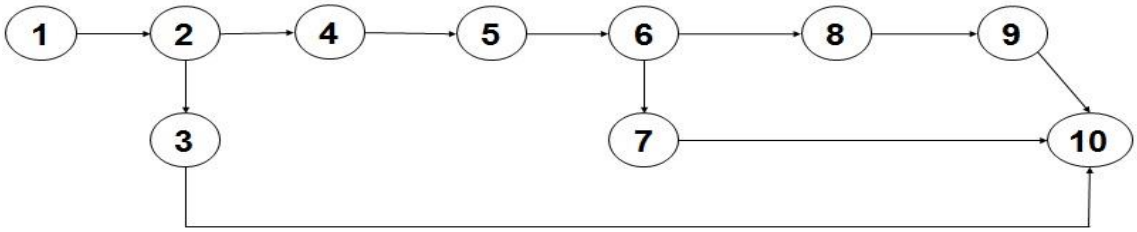


Ilustración 24: Grafo de flujo asociado a la funcionalidad updateAction (Request $request, $id)

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea el correcto (Pressman, 2005).

1. **V (G) = R** donde R representa la cantidad total de regiones.

   V (G) = 3

2. **V (G) = A – N + 2** donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

   V (G) = 11– 10 + 2

   V (G) = 3

3. **V (G) = P + 1** donde P es el número de nodos predicado contenidos en el grafo de flujo (se denomina nodo predicado a los nodos de los cuales parten dos o más aristas).

   V (G) = 2 + 1

   V (G) = 3

Dado a que el cálculo de las tres fórmulas anteriormente mencionadas arrojó el mismo resultado se puede plantear que la complejidad ciclomática del método updateAction (Request $request, $id) es 3. Esto significa que existen 3 posibles caminos por donde el flujo puede circular. Este valor representa el número mínimo de casos de pruebas para el procedimiento tratado (Pressman, 2005).

- Camino básico #1: 1-2-3-10.
- Camino básico #2: 1-2-4-5-6-7-10.
- Camino básico #3: 1-2-4-5-6-8-9-10.

Para cada camino básico determinado se realiza un diseño de caso de prueba.

Caso de prueba para el camino básico #1 (1-2-3-10)	
Descripción	Debe determinar una entidad proyecto nula mostrándole al usuario el error ocurrido.
Condición de ejecución	La entidad proyecto debe estar vacía.
Entrada	id = 8
Resultado esperado	Se obtiene una entidad proyecto nula.
Resultado de la prueba: satisfactorio	

Tabla 20: Caso de prueba para el camino básico #1

Caso de prueba para el camino básico #2 (1-2-4-5-6-7-10)	
Descripción	Debe mostrar un formulario con los datos de la entidad proyecto a modificar y al adicionar datos correctos debe permitir la modificación de la entidad, y mostrar un mensaje de afirmación.
Condición de ejecución	La entidad proyecto no debe estar vacía y el formulario debe ser válido.
Entrada	id = 2
Resultado esperado	Se obtiene una entidad proyecto modificada satisfactoriamente.
Resultado de la prueba: satisfactorio	

Tabla 21: Caso de prueba para el camino básico #2

Caso de prueba para el camino básico #3 (1-2-4-5-6-8-9-10)	
Descripción	Debe mostrar un formulario con los datos de la entidad proyecto a modificar y al adicionar datos incorrectos mostrar un mensaje de error.
Condición de ejecución	La entidad proyecto no debe estar vacía y el formulario no debe ser válido.
Entrada	id = 2
Resultado esperado	Se obtiene un formulario con los errores encontrados.
Resultado de la prueba: satisfactorio	

Tabla 22: Caso de prueba para el camino básico #3.

3.6.2 Pruebas de caja negra

Luego de aplicadas las pruebas de caja blanca, se pasó a la aplicación de las pruebas de caja negra. Estas pruebas fueron aplicadas con el objetivo de buscar errores de interfaz. Para la aplicación de las pruebas de caja negra se confeccionaron los diseños de casos de pruebas para cada una de las funcionalidades de la herramienta. Para comprobar la calidad de la herramienta fueron realizadas dos iteraciones de pruebas por el grupo de calidad del centro CEIGE[19]. El equipo de desarrollo clasificó las no conformidades en:

- No conformidades detectadas en la documentación.
- No conformidades detectadas en la aplicación.

En la siguiente tabla se recoge la cantidad de no conformidades detectadas en la documentación y en la aplicación por cada iteración:

Tipo de no conformidades	Documentación	Aplicación
Primera iteración		
Significativa	0	0
No significativa	1	1
Total	1	1
Segunda iteración		
Significativa	0	0
No significativa	0	0
Total	0	0

Tabla 23: Tabla de no conformidades detectadas en la documentación y en la aplicación

3.6.3 Pruebas de aceptación

Luego de aplicadas las pruebas de caja negra, se aplicaron las pruebas de aceptación. Estas pruebas son aplicadas por el cliente en conjunto con el equipo de desarrollo. Las pruebas de aceptación tienen como objetivos validar que la herramienta desarrollada cumple con los requisitos identificados en la fase de planificación.

A continuación se muestran varios Casos de Pruebas de Aceptación (CPA) que se realizaron para validar el sistema a partir de las historias de usuarios y los requisitos del sistema, de un total de 33, el resto de los CPA se encuentran en el **Anexo 5**. Estos CPA se aplican a la herramienta utilizando un caso de estudio de factibilidad, lo cual posibilita probar el funcionamiento completo de la aplicación por parte del usuario o cliente final ya que mediante este se puede ejecutar todo el proceso de análisis de factibilidad económica.

Casos de pruebas de aceptación	
Código: 05	**Historia de Usuario (Nro. y nombre):** HU #02 Gestionar proyecto.
Funcionalidad que se prueba: adicionar proyecto.	
Condiciones de ejecución: el usuario que está autenticado debe tener privilegios de especialista.	

[19] **CEIGE:** Centro de Gestión de Entidades

Acción	Datos de entrada	Resultado esperado
Se selecciona la opción evaluación		Se despliega el menú mostrando la opción Proyecto
Se selecciona la opción proyecto		El sistema muestra el listado con los proyectos que han sido adicionados con anterioridad. Se muestran las acciones a realizar sobre cada proyecto.
Se escoge la opción de Adicionar		El sistema muestra un formulario para introducir los datos necesarios para adicionar un nuevo proyecto.
Se introducen los datos necesarios para adicionar un proyecto	datos obligatorios	
Se selecciona el botón Adicionar	nuevo proyecto	El sistema muestra un mensaje indicando que la acción se ha realizado satisfactoriamente. Se muestra el listado de los proyectos incluyendo el adicionado
Resultado de la prueba: satisfactorio		

Tabla 24: Descripción del caso de prueba de aceptación HU#2.

Casos de pruebas de aceptación		
Código: 06	**Historia de Usuario (Nro. y nombre):** HU #02 Gestionar proyecto.	
Funcionalidad que se prueba: modificar proyecto.		
Condiciones de ejecución: el usuario que está autenticado debe tener privilegios de especialista. Debe existir al menos un proyecto adicionado con anterioridad.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción evaluación		Se despliega el menú mostrando la opción Proyecto
Se selecciona la opción proyecto		El sistema muestra el listado con los proyectos que han sido adicionados con anterioridad. Se muestran las acciones a realizar sobre cada proyecto.
Se selecciona el proyecto a modificar, se selecciona la opción modificar	proyecto a modificar	El sistema muestra un formulario con los datos actuales
Se introducen los nuevos valores para actualizar los campos que se desean modificar	datos obligatorios	

Se selecciona el botón Modificar	proyecto modificado	El sistema muestra un mensaje indicando que se ha realizado la acción se ha realizado satisfactoria. Se muestra el listado de proyectos.
Resultado de la prueba: satisfactorio		

Tabla 25: Descripción del caso de prueba de aceptación HU#2.

Casos de pruebas de aceptación		
Código: 07	**Historia de Usuario (Nro. y nombre):** HU #02 Gestionar proyecto.	
Funcionalidad que se prueba: eliminar proyecto.		
Condiciones de ejecución: el usuario que está autenticado debe tener privilegios de especialista. Debe existir al menos un proyecto adicionado con anterioridad.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción evaluación		Se despliega el menú mostrando la opción Proyecto
Se selecciona la opción proyecto		El sistema muestra el listado con los proyectos que han sido adicionados con anterioridad. Se muestran las acciones a realizar sobre cada proyecto.
Se selecciona el proyecto que se desea eliminar, se escoge la opción eliminar	proyecto a eliminar	Se muestra un mensaje preguntando la confirmación de la acción a realizar.
Se selecciona la opción Aceptar	proyecto a eliminar	Se elimina el proyecto. Se muestra un mensaje indicando que la acción se ha realizado satisfactoriamente. Se muestra el listado de proyectos
Resultado de la prueba: satisfactorio		

Tabla 26: Descripción del caso de prueba de aceptación HU#2.

Casos de pruebas de aceptación		
Código: 08	**Historia de Usuario (Nro. y nombre):** HU #02 Gestionar proyecto.	
Funcionalidad que se prueba: buscar proyecto.		
Condiciones de ejecución: el usuario que está autenticado debe tener privilegios de especialista. Debe existir al menos un proyecto adicionado con anterioridad.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción evaluación		Se despliega el menú mostrando la opción Proyecto

Se selecciona la opción proyecto		El sistema muestra el listado con los proyectos que han sido adicionados con anterioridad. Se muestran las acciones a realizar sobre cada proyecto.
Se introduce en el buscador el proyecto a buscar	proyecto a buscar	El sistema muestra un mensaje en caso de no encontrarse el proyecto buscado, sino muestra el proyecto con la respectiva información.
Resultado de la prueba: satisfactorio		

Tabla 27: Descripción del caso de prueba de aceptación HU#2.

3.7 Aprobación de la solución

Luego de la culminación de las pruebas unitarias y de sistema se procedió a realizar las pruebas de aceptación; estas últimas bajo un ambiente controlado son capaces de determinar si el software realiza lo que desea el cliente (Pressman, 2005). Terminadas estas pruebas y corregidas las no conformidades el cliente avaló la solución con la entrega del Certificado de Aceptación del Producto por parte de la MSc. Marieta Peña Abreu, debido a cumplía con los requisitos funcionales identificados por el cliente y el equipo de desarrollo. La carta de aceptación puede ser consultada en el **Anexo 6**.

3.8 Validación de la investigación

La siguiente tabla muestra una comparación de los resultados arrojados por los métodos económicos con números exactos y con números difusos triangulares a una muestra de cuatro proyectos seleccionados en la universidad. Los números difusos triangulares poseen un nivel de confianza, siendo un valor central cuyo nivel de confianza es igual a uno y dos valores extremos cuyos niveles de confianza es igual a cero.

	Resultado arrojado manualmente con números exactos			**Resultado arrojado con la herramienta**		
Proyectos	**VAN**	**TIR**	**PR (meses)**	**VAN**	**TIR**	**PR**
SIGC	$ 25521.95	0.24	9.81	24920.19, **25521.95**, 26123.72	0.21, **0.24**, 0.34	9.75, **9.81**, 9.84
GINA	$ 257653.93	0.65	3.21	257052.18, **257653.93**, 258255.71	0.56, **0.65**, 0.80	3.21, **3.21**, 3.21
DAEDOICIM	$ 627.39	0.12	11.10	354.83, **627.39**, 899.95	0.03, **0.12**, 0.14	11.04, **11.10**, 11.13
Prisiones de	$ 633718.49	0.56	3.63	633141.46,	0.53,	3.63,

Cuba				**633718.49**, 634508.60	**0.56**, 0.60	**3.63**, 3.63

Tabla 28: Resultados comparativos antes y después

El desarrollo de la Herramienta para evaluar factibilidad económica de proyectos de software basada en técnicas de Soft Computing contribuirá a un mejor desarrollo de software en la universidad, facilitando la realización de estudios de factibilidad económica en entornos de incertidumbre.

3.9 Conclusiones del capítulo

- La elaboración de los artefactos en la fase de implementación, permitió el correcto desarrollo de la herramienta.
- El uso de los estándares de codificación, permitió tener más organizado y entendible el código para el equipo de desarrollo
- Se definió el manejo de la seguridad del sistema, permitiendo controlar a qué parte del software puede acceder cada usuario.
- Las pruebas unitarias realizadas a varios algoritmos del sistema posibilitaron la detección de errores, arrojando finalmente resultados satisfactorios luego de corregidos estos.
- A partir de las dos iteraciones de pruebas funcionales realizadas por el grupo de calidad del CEIGE, se concluye que las funcionalidades implementadas mostraron un correcto funcionamiento y dan respuesta a los requisitos funcionales.
- Las pruebas de aceptación permitieron validar la solución, y verificar que se cumplieron los requisitos identificados por el cliente y el equipo de desarrollo.

Conclusiones Generales

Mediante el desarrollo de este trabajo se llegó a las siguientes conclusiones:

- Luego de realizar un análisis de las herramientas informáticas y modelos utilizados actualmente para realizar estudios de factibilidad de proyectos, se llegó a la conclusión de que no tratan suficientemente la incertidumbre presente en la información.
- Los artefactos generados en cada una de las fases permitió un mejor entendimiento de la solución, así como
- Los artefactos generados en la fase de planificación y análisis permitieron realizar la herramienta con la mayor calidad posible.
- Para el desarrollo de la herramienta se analizaron un conjunto de posibles herramientas y tecnologías a utilizar, seleccionando finalmente las que satisfacían las necesidades del sistema a desarrollar y todas son tecnologías libre.
- Al sistema se le realizaron pruebas funcionales, de caja blanca y aceptación, permitiendo encontrar errores en los datos, algoritmos y en la interfaz.

Recomendaciones

1. Hacer uso de la herramienta en los proyectos de la universidad.
2. Integrar con estudios de factibilidad técnico y comercial.
3. Introducir los datos del proyecto mediante la importación de archivos Excel.

Referencias

Abreu, Marieta Peña. 2012. Modelo para análisis de factibilidad en la evaluación de proyectos de software. La Habana : Universidad de las Ciencias Informáticas, 2012.

Acevedo, Karen. 2010. *Estudio de Factibilidad de un proyecto.* Atlántico : Slideshare, 2010.

Allen, Brealey Myers. 2008. *Principios de Finanzas Corporativas.* 9. México : Facultad de Contaduría y Administración, 2008.

Álvarez, Alberto Díaz. 2014. *Inteligencia Artificial y Soft Computing.* [ed.] Universidad Politécnica de Madrid. Madrid : s.n., 2014.

Analyzer, Managerial. 2014. Managerial Analyzer. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] http://www.managerialanalyzer.com/web/index.php.

Apache. 2014. Apache. HTTP server project. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] http://httpd.apache.org/docs/2.4/.

Beck, Kent. 2000. *Una explicación de la programación extrema. Aceptar el cambio.* s.l. : Pearson Education, 2000.

Bishop, Christopher M. 2006. *Pattern Re cognition and Machine Learning. Springer Science+Bussiness Media.* Gran Bretaña : s.n., 2006.

Castro, Maylé Díaz. 2010. Método de evaluación de proyectos para decidir su aceptación. La Habana : Universidad de las Ciencias Informáticas, 2010.

Chaves, Michael Arias. 2007. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. Costa Rica : s.n., 2007. Vol. VI, 10.

Choice, Expert. 2015. Collaboration software to meet the needs of your organization. [En línea] 02 07, 2015. [Citado el: 02 07, 2015.] http://expertchoice.com/products-services/.

Computing, Soft. 2015. European Centre for Soft Computing. [En línea] 02 07, 2015. [Citado el: 02 07, 2015.] http://www.softcomputing.es/metaspace/portal/3/73.

Cubas, Soledad Valero. 2010. *Arquitectura de Búsqueda Basada en Técnicas Soft Computing Para La Resolución De Problemas CoMbinatorios En Diferentes Dominios De Aplicación.* Valencia : Universidad Politécnica de Valencia, 2010. pág. 24.

Date, C. J. 2001. *Introducción a los sistemas de bases de datos.* Mexico : s.n., 2001.

Difusa, Lógica. 2015. Logica Difusa - Enciclopedia EMVI . [En línea] 02 04, 2015. [Citado el: 02 04, 2015.] http://www.eumed.net/diccionario/definicion.php?dic=1&def=136.

Diseño, Métricas de. 2014. Métricas de Software. [En línea] 08 22, 2014. [Citado el: 03 05, 2015.] http://www.desarrolloweb.com.

Easyplanex. 2014. Descripción de EasyPlanEx. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] https://www.borasystems.com/es/productos-software/easyplanex/.

Económicos, Modelos. 2015. *Economía.* [En línea] 01 17, 2015. [Citado el: 01 14, 2015.] http://economiaes.com/economicos-modelos.html.

Eguiluz, Javier. 2011. *Desarrollo Web Ágil con Symfony2.* Primera edición. 2011. págs. 13-14.

Escobar, Karel Riverón. 2013. Sistema basado en casos para apoyar el proceso de toma de decisiones en las organizaciones frente a una mejora de procesos de software. *Sistema basado en casos para apoyar el proceso de toma de decisiones en las organizaciones frente a una mejora de procesos de software.* La Habana, Cuba : s.n., 06 03, 2013.

EvalAs. 2014. EvalAs. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] http://www.elsitioagricola.com/soft/evalas/evalAs1_0.asp.

Fernández, Adelaida Ramírez. 2012. Clasificación de tipos de requisitos para la mejora del proceso de desarrollo de software. 2012.

Firefox, Mozilla. 2014. caracteristicas-principales-de-firefox. [En línea] 11 23, 2014. [Citado el: 02 05, 2015.] (http://www.proyectoautodidacta.com/comics/caracteristicas-principales-de-firefox/).

Gálvez, Nils. 2013. Metodologías ágiles para el desarrollo de software: programación extrema xp. Perú : s.n., 12 06, 2013. pág. 23.

Ganesh, Sai Gunda. 2008. *Requirements Engineering: Elicitation Techniques.* Suecia : Universidad de Suecia, 2008.

Gespro. 2015. Suite Gespro: Ecosistema Gespro. [En línea] 02 08, 2015. [Citado el: 02 08, 2015.] http://suitegespro.blogspot.com/2013/02/normal-0-false-false-false-en-us-x-none.html.

González, Alejandro Nieto. 2011. Economía y finanzas. *Economía y finanzas.* [En línea] 02 11, 2011. [Citado el: 06 11, 2015.] http://www.elblogsalmon.com/conceptos-de-economia/que-son-el-van-y-el-tir.

Gónzalez, S. 2002. Preparación y evaluación de proyectos informáticos. s.l. : Departamento de industria y negocio, 2002. págs. 1 - 135.

Gutiérrez, Escalona, Mejías, Torres. 2006. Pruebas del Sistema en Programación Extrema. Departamento de Lenguajes y Sistemas Informáticos., Universidad de Sevilla. : s.n., 2006.

Hernandez, Cire. 2013. Neurocomputación. *Neurocomputación.* [En línea] 10 10, 2013. [Citado el: 02 10, 2015.] https://prezi.com/ryxrembh_i6j/neurocomputacion/.

Herrera. 2015. Introducción a la Ingeniería Industrial. *Introducción a la Ingeniería Industrial.* [En línea] 01 18, 2015. [Citado el: 02 10, 2015.] http://www1.herrera.unt.edu.ar/faceyt/fyep/files/2013/03/APENDICE_PRACTICO_4.pdf.

HTML5. 2015. Algunas Características de HTML5 que usted debe saber. [En línea] 02 08, 2015. [Citado el: 02 08, 2015.] http://latindesigner.net/algunas-caracteristicas-de-html5-que-usted-debe-saber/.

IA. 2015. La Inteligencia Artificial. [En línea] 02 07, 2015. [Citado el: 02 07, 2015.] http://www.fgcsic.es/lychnos/es_es/articulos/inteligencia_artificial.

Introducción al razonamiento aproximado: lógica difusa. **Vito, Eduardo Luis De. 2006.** Buenos Aires : Revista Argentina de Medicina Respiratoria, 2006, pág. 129.

Jacobson, Ivar. 1999. *El Proceso Unificado de Desarrollo del Software.* 1999.

Joskowicz, José. 2008. *Reglas y prácticas en eXtreme Programming.* España : s.n., 2008. pág. 10.

Lamadrid, Giorgy Gilberto Cabrera. 2012. Herramienta para evaluar factibilidad en proyectos. *Herramienta para evaluar factibilidad en proyectos.* La Habana : s.n., 2012. pág. 15.

Larman, Craig. 1999. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2da. México : s.n., 1999. pág. 162.

Letelier, Patricio. 2012. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Valencia : Universidad de Valencia, 2012.

Llano, Giorgy Gilberto Cabrera Lamadrid y Pedro Frank Cadenas del. 2012. Herramienta para evaluar factibilidad en proyectos de software. La Habana : Universidad de las Ciencias Informáticas, 2012.

Matich, Damián Jorge. 2001. *Redes Neuronales: Conceptos Básicos y Aplicaciones.* Rosario : Grupo de Investigación Aplicada a la Ingeniería Química, 2001.

Mesa, Gonzalo M. Rodríguez. 2006. *La evaluación financiera y social de proyectos de inversión.* 3ra. La Habana, Cuba : Universidad de La Habana, 2006. pág. 22.

Morales, Dunia Gónzalez. 2014. Tratamiento Borroso para valorar el nivel de importancia. Villa Clara : s.n., 08 25, 2014.

Moreno, Ana. 2005. Técnicas de evaluación de software. 2005. págs. 40-43.

—. 2005. Técnicas de evaluación de software. *Técnicas de evaluación de software.* 10 17, 2005. págs. 40-43.

NetBeans. 2014. NetBeans IDE Features. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] https://netbeans.org/features/index.html.

Pacheco, Nacho. 2013. *Seguridad_Manual de Symfony en Español.* 2013.

Paradigm, Visual. 2014. Visual Paradigm For Um. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] http://es.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

PHP. 2014. ventajas de PHP. [En línea] 12 02, 2014. [Citado el: 12 02, 2014.] http://www.4rsoluciones.com/las-ventajas-de-php-para-el-desarrollo-de-aplicaciones-y-sitios-web/.

PMI. 2008. *Guía de los fundamentos de la dirección de proyectos.* s.l. : Project Management Institute,Inc, 2008.

PostgreSQL. 2014. PostgreSQL. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] http://www.postgresql.org.es/sobre_postgresql.

Pressman, Roger. 2005. *Ingeniería de Software. Un enfoque práctico.* Quinta edición. 2005.

Prieto, Félix. 2009. *Patrones de diseño.* España : Departamento de Informática. Universidad de Valladolid., 2009.

Proyectos, Dirección de. 2014. Causas del fracaso en los proyectos. *Causas del fracaso en los proyectos.* [En línea] 09 10, 2014. [Citado el: 03 07, 10.] http://www.bpmsat.com/causas-fracaso-proyectos/.

Proyectos, Gestión de. 2013. Gestion de Proyectos Software. [En línea] 05 14, 2013. [Citado el: enero 20, 2015.] https://sites.google.com/site/gestiondeproyectossoftware/.

Serrano, Gloria Pérez. 1999. *Elaboración de proyectos sociales. Casos prácticos.* Madrid, España. : s.n., 1999.

Software, Proyectos de. 2014. Gestión de Software. [En línea] 01 25, 2014. [Citado el: 01 25, 2015.] https://arlethparedes.wordpress.com/2012/08/26/porque-fallan-los-proyectos-de-software/.

Sommerville, Ian. 2005. *Ingeniería de Software.* 7ma. 2005.

Symfony. 2012. El patrón MVC(Symfony). *El patrón MVC(Symfony).* [En línea] 04 10, 2012. [Citado el: 06 12, 2015.] http://librosweb.es/libro/symfony_1_4/capitulo_2/el_patron_mvc.html.

—. 2014. Symfony. [En línea] 11 09, 2014. [Citado el: 11 09, 2014.] symfony.es.

Tamayo, Karina Sánchez. 2010. Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones. La Habana : Universidad de las Ciencias Informáticas, 2010.

Tobón, luis Miguel Echeverry. 2007. Caso práctico de la metodología ágil XP al desarrollo de software. *Caso práctico de la metodología ágil XP al desarrollo de software.* 2007.

Trigas, Manuel. 2011. Gestión de proyectos informáticos. Metodología Scrum. 2011.

Vázquez, Burguette. 2010. Una propuesta de síntesis para la evaluación y selección de proyectos de nuevos productos. s.l., Europa, Investigaciones Europeas de Dirección y Economía de la empresa : Universidad de León, 2010. Vol. 2.

Visconti, Marcelo. 2012. Daigrama de Despliegue. [En línea] 2012. [Citado el: 03 08, 2015.] http://www.diadspg.blogspot.com/.

Zadeh, Lotfi A. 1996. *Nacimiento y evolución de la lógica borrosa, el soft computing y la computación con palabras: un punto de vista personal.* California : s.n., 1996. pág. 421.

Anexos

Anexo 1: Historia de usuario.

Historia de usuario	
Número: HU #01	**Historia de usuario:** Gestionar usuario.
Modificación de la historia de usuario: ninguna	
Programador: Ubesnel Rojas Orozco y Orlando Guilarte Cutiño	**Iteración asignada:** primera
Prioridad: alta	**Puntos estimados:** 4 días
Riesgo en desarrollo: bajo	**Puntos reales:** 4 días
Descripción Inicialmente se deben listar todos los usuarios que han sido adicionados con anterioridad. De los usuarios se debe mostrar la siguiente información: nombre, apellidos, rol. Además debe permitir realizar las siguientes acciones: adicionar, modificar, eliminar, buscar y listar.	
Observaciones • Las acciones sobre los proyectos solo pueden ser realizadas por usuarios con rol especialista. • Para modificar, eliminar, buscar, mostrar y listar un proyecto, debe haber sido adicionado al menos un proyecto con anterioridad.	

Historia de Usuario	
Código: HU # 03	**Nombre Historia de Usuario:** Gestionar información económica.
Modificación de la Historia de Usuario: ninguna	
Programador: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco	**Iteración asignada:** primera
Prioridad: alta	**Puntos estimados:** 5 días
Riesgo de desarrollo: alta	**Puntos reales:** 5 días
Descripción Inicialmente debe mostrarse un listado con todos los proyectos que han sido adicionados con anterioridad. Además de las acciones: adicionar, modificar, eliminar y mostrar la información económica asociada a un proyecto.	
Observaciones: • Las acciones sobre la información económica solo pueden ser realizadas por usuarios con rol especialista.	

Historia de Usuario	
Código: HU # 04	**Nombre Historia de Usuario:** Gestionar proceso de evaluación.
Modificación de la Historia de Usuario: ninguna	
Programador: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco	**Iteración asignada:** primera
Prioridad: alta	**Puntos estimados:** 6 días
Riesgo de desarrollo: bajo	**Puntos reales:** 6 días
Descripción La funcionalidad gestionar proceso de evaluación debe permitir adicionar, modificar, mostrar, eliminar y buscar un proceso de evaluación. Inicialmente se debe mostrar una página donde aparezcan todos los procesos de evaluación que han sido adicionados con anterioridad con la fecha de inicio y fecha fin del proceso de evaluación y el estado en que se encuentra.	
Observaciones: • Las acciones de gestionar proceso de evaluación solo pueden ser realizadas por los usuarios con rol de especialista.	

Historia de Usuario	
Código: HU # 05	**Nombre Historia de Usuario:** Gestionar proyecto a evaluar durante una evaluación.
Modificación de la Historia de Usuario: ninguna	
Programador: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco	**Iteración asignada:** primera
Prioridad: alta	**Puntos estimados:** 3 días
Riesgo de desarrollo: bajo	**Puntos reales:** 3 días
Descripción La funcionalidad gestionar proyecto a evaluar durante una evaluación, debe permitir adicionar y eliminar un proyecto a evaluar durante una evaluación. Inicialmente debe mostrarse un formulario donde aparezcan los proyectos que se pueden seleccionar para evaluar.	
Observaciones: • Las acciones solo pueden ser realizadas por un usuario con rol de especialista. • Para eliminar un proyecto debe existir un proceso de evaluación agregado con anterioridad.	

Historia de Usuario	
Código: HU # 06	**Nombre Historia de Usuario:** Gestionar método a evaluar durante una evaluación.
Modificación de la Historia de Usuario: ninguna	
Programador: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	**Iteración asignada:** primera.
Prioridad: alta	**Puntos estimados:** 3 días
Riesgo de desarrollo: alta	**Puntos reales:** 3 días
Descripción La funcionalidad gestionar método a evaluar durante una evaluación, debe permitir adicionar y eliminar un método al proceso de evaluación. Inicialmente debe mostrarse un formulario donde aparezcan los cuatro métodos que se pueden seleccionar para evaluar un proyecto.	
Observaciones: • Las acciones solo pueden ser realizadas por un usuario con rol de especialista. • Se debe agregar un proceso de evaluación para seleccionar los métodos económicos a evaluar.	

Historia de Usuario	
Código: HU # 07	**Nombre Historia de Usuario:** Realizar análisis de factibilidad económica.
Modificación de la Historia de Usuario: ninguna.	
Programador: Ubesnel Rojas Orozco y Orlando Guilarte Cutiño	**Iteración Asignada:** primera
Prioridad: alta	**Puntos Estimados:** 20 días
Riesgo en Desarrollo: alta	**Puntos Reales:** 20 días
Descripción La funcionalidad realizar análisis de factibilidad económica debe permitir calcular cada uno de los métodos económicos (FC, VAN, PR, TIR) para todos los proyectos que han sido adicionados al sistema. Se calculan por independiente cada uno de los métodos a los proyectos que se le haya adicionado el método al proceso de evaluación.	
Observaciones • Las acciones solo pueden ser realizadas por un usuario con rol de especialista. • Debe existir un proyecto adicionado con anterioridad y tener asociado una información económica. • El proyecto debe estar asociado a un proceso de evaluación.	

Historia de Usuario	
Código: HU # 08	**Nombre Historia de Usuario:** Generar Reporte.
Modificación de la Historia de Usuario: ninguna	
Programador: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco	**Iteración asignada:** segunda
Prioridad: media	**Puntos estimados:** 10 días
Riesgo de desarrollo: medio	**Puntos reales:** 10 días

Descripción

La funcionalidad generar reporte debe permitir al usuario, generar reporte de un proyecto en formato PDF. Además debe permitir generar un reporte de todos los proyectos. Inicialmente se debe escoger cuál de los dos reportes desea realizar.

Reporte de un proyecto: se seleccionará el proyecto sobre el que se desea obtener un reporte.

Reporte de todos los proyectos: se seleccionará la opción generar reporte de todos los proyectos. Se mostrará la opción de guardar el reporte.

Observaciones:

- Las acciones solo pueden ser realizadas por los decisores
- Debe existir algún proyecto evaluado con anterioridad

Historia de Usuario	
Código: HU # 09	**Nombre Historia de Usuario:** Mostrar análisis de factibilidad económica.
Modificación de la Historia de Usuario: ninguna	
Programador: Orlando Guilarte Cutiño y Ubesnel Rojas Orozco.	**Iteración asignada:** tercera
Prioridad: baja.	**Puntos estimados:** 10 días
Riesgo de desarrollo: bajo	**Puntos reales:** 10 días

Descripción

La funcionalidad mostrar análisis de factibilidad económica debe permitir mostrar el resultado de la evaluación de los proyectos por cada uno de los métodos económicos que ha sido evaluado. Además debe permitir mostrar el análisis de factibilidad económica de todos los proyectos.

Observaciones:

- Debe existir algún proyecto evaluado con anterioridad.
- Las acciones solo pueden ser realizadas por usuario con rol de especialista o decisor.

Anexo 2: Tarjetas CRC.

Clase: informacionEconomica	
Tipo de clase: Entidad	
Responsabilidades	**Colaborador**
Obtener compra de activo.	
Modificar compra de activo.	
Obtener capital de trabajo.	
Modificar capital de trabajo.	
Obtener ventas e ingresos.	
Modificar ventas e ingresos.	
Obtener mano de obra.	
Modificar mano de obra.	
Obtener materiales.	
Modificar materiales.	
Obtener depreciación.	
Modificar depreciación.	

Clase: Evaluacion	
Tipo de clase: Entidad	
Responsabilidades	**Colaborador**
Obtener la fecha inicio de la evaluación.	
Modificar la fecha inicio de la evaluación.	
Obtener la fecha fin de la evaluación.	
Modificar la fecha fin de la evaluación.	
Adicionar métodos económicos a la evaluación.	Metodo
Eliminar métodos económicos de la evaluación.	Metodo
Obtener métodos económicos de la evaluación.	Metodo
Modificar métodos económicos de la	Metodo
Adicionar proyectos a la evaluación.	Proyecto
Eliminar proyectos de la evaluación.	Proyecto
Obtener proyectos de la evaluación.	Proyecto
Modificar proyectos de la evaluación.	Proyecto
Guardar resultado de la evaluación.	valorProyecto
Mostrar resultado de la evaluación.	valorProyecto

Clase: EvaluacionController	
Tipo de clase: Controladora	
Responsabilidades	**Colaborador**
Listar las evaluaciones.	Evaluacion
Adicionar evaluación.	Evaluacion
Modificar evaluación.	Evaluacion
Mostrar evaluación.	Evaluacion
Eliminar evaluación.	Evaluacion

Clase: informacionEconomicaController	
Tipo de clase: Controladora	
Responsabilidades	**Colaborador**
Listar proyectos a evaluar.	Proyecto
Calcular flujo de caja.	informacionEconomica
Calcular el valor actual neto.	informacionEconomica
Adicionar información económica.	informacionEconomica
Modificar información económica.	informacionEconomica
Mostrar información económica.	informacionEconomica
Eliminar información económica.	informacionEconomica
Calcular tasa interna de retorno.	informacionEconomica
Calcular período de recuperación.	informacionEconomica

Clase: Metodo	
Tipo de clase: Entidad	
Responsabilidades	**Colaborador**
Obtener el método económico.	
Modificar el método económico.	

Clase: Usuario	
Tipo de clase: Entidad	
Responsabilidades	**Colaborador**
Obtener usuario.	
Modificar usuario.	

Obtener nombre del usuario.	
Modificar nombre del usuario.	
Asignar un rol a un usuario.	Rol
Mostrar el rol de un usuario.	Rol
Modificar rol de un usuario	Rol

Clase: UsuarioController	
Tipo de clase: Controladora	
Responsabilidades	**Colaborador**
Listar usuarios.	Usuario
Adicionar usuario.	Usuario
Modificar usuario.	Usuario
Mostrar usuario.	Usuario
Eliminar usuario.	Usuario

Clase: valorProyecto	
Tipo de clase: Entidad	
Responsabilidades	**Colaborador**
Obtener el resultado de la evaluación.	
Guardar el resultado de la evaluación.	

Clase: valorProyectoController	
Tipo de clase: Controladora	
Responsabilidades	**Colaborador**
Mostrar los flujos de caja.	valorProyecto
Mostar valor actual neto.	valorProyecto
Mostrar tasa interna de retorno.	valorProyecto
Mostrar período de recuperación.	valorProyecto
Mostrar listado de proyectos evaluados.	valorProyecto, Proyecto

Anexo 3: Resultados de la métrica TOC.

No	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
1	UsuarioController	7	Baja	Baja	Alta
2	DefaultController	1	Baja	Baja	Alta
3	UsuarioRepository	2	Baja	Baja	Alta
4	WelcomeController	1	Baja	Baja	Alta
5	RolRepository	0	Baja	Baja	Alta
6	Rol	10	Media	Media	Media
7	Usuario	16	Media	Media	Media
8	EvaluacionController	8	Baja	Baja	Alta
9	ProyectoController	9	Media	Media	Media
10	ReporteController	4	Baja	Baja	Alta
11	ValorProyectoController	6	Baja	Baja	Alta
12	InformacionEconomicaController	18	Alta	Alta	Baja
13	Periodo	9	Media	Media	Media
14	Evaluacion	20	Alta	Alta	Baja
15	EvaluacionRepository	5	Baja	Baja	Alta
16	Metodo	15	Media	Media	Media
17	MetodoRepository	1	Baja	Baja	Alta
18	Proyecto	70	Alta	Alta	Baja
19	ProyectoRepository	6	Baja	Baja	Alta
20	InformacionEconomica	40	Alta	Alta	Baja
21	InformacionEconomicaRepository	26	Alta	Alta	Baja
22	ValorProyecto	14	Baja	Baja	Alta
23	ValorProyectoRepository	8	Baja	Baja	Alta

Anexo 4: Resultados de la métrica RC.

No	Clase	Cantidad de relaciones	Acoplamiento	Complejidad de mantenimiento	Cantidad de Pruebas	Reutilización
1	UsuarioController	4	Alto	Alta	Alta	Baja
2	DefaultController	2	Medio	Media	Media	Media
3	UsuarioRepository	1	Bajo	Baja	Baja	Alta
4	WelcomeController	1	Bajo	Baja	Baja	Alta
5	RolRepository	1	Bajo	Baja	Baja	Alta
6	Rol	2	Medio	Media	Media	Media
7	Usuario	3	Alto	Media	Media	Media
8	EvaluacionController	4	Alto	Alta	Alta	Baja
9	ProyectoController	4	Alto	Alta	Alta	Baja
10	ReporteController	2	Medio	Media	Media	Media
11	ValorProyectoController	1	Bajo	Baja	Baja	Alta
12	InformacionEconomicaController	4	Alto	Alta	Alta	Baja
13	Periodo	1	Bajo	Baja	Baja	Alta
14	Evaluacion	2	Medio	Media	Media	Media
15	EvaluacionRepository	1	Bajo	Baja	Baja	Alta
16	Metodo	1	Bajo	Baja	Baja	Alta
17	MetodoRepository	1	Bajo	Baja	Baja	Alta
18	Proyecto	2	Medio	Media	Media	Media
19	ProyectoRepository	1	Bajo	Baja	Baja	Alta
20	InformacionEconomica	1	Bajo	Baja	Baja	Alta
21	InformacionEconomicaRepository	1	Bajo	Baja	Baja	Alta
22	ValorProyecto	1	Bajo	Baja	Baja	Alta
23	ValorProyectoRepository	1	Bajo	Baja	Baja	Alta

Anexo 5: Casos de pruebas de aceptación.

Casos de prueba de aceptación		
Código: 22	**Historia de Usuario (Nro. y nombre):** HU #07 Realizar análisis de factibilidad	
Funcionalidad que se prueba: evaluar método flujo de caja con números triangulares		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista. Debe existir al menos un proyecto adicionado con anterioridad. El proyecto debe estar asociado a un proceso de evaluación		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción Evaluación		Se despliega el menú, mostrando la opción Evaluar proyectos
Se selecciona la opción Evaluar proyectos		Se muestra una lista con todos los proyectos que han sido adicionados con anterioridad
Se selecciona el proyecto a evaluar, se selecciona la opción Calcular FC		Se muestra un mensaje, confirmando que se ha evaluado el proyecto satisfactoriamente
Resultado de la prueba: Satisfactorio		

Casos de prueba de aceptación		
Código: 29	**Historia de Usuario (Nro. y nombre):** HU #01 Gestionar usuario	
Funcionalidad que se prueba: Listar usuarios		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de administrador. Debe existir al menos un usuario adicionado con anterioridad		
Acción	**Datos de entrada**	**Resultado esperado**
Se accede a la funcionalidad Administración		Se despliega el menú, visualizando el menú usuario
Se accede a la funcionalidad usuario		El sistema muestra el listado con los usuarios que han sido adicionados con anterioridad
Resultado de la prueba: Satisfactorio		

Casos de prueba de aceptación		
Código: 09	**Historia de Usuario (Nro. y nombre):** HU #03 Gestionar información económica.	
Funcionalidad que se prueba: Adicionar información económica.		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción Evaluación		Se despliega el menú, mostrando la opción Información económica
Se selecciona la opción Información económica		Se muestra el listado de proyectos que han sido adicionados con anterioridad. Además se muestran las acciones a realizar sobre cada proyecto
Se selecciona el proyecto que se desea adicionar la información económica. Se selecciona la opción Adicionar	proyecto	El sistema muestra un formulario para adicionar la información económica del proyecto seleccionado
Se introducen los datos necesarios para adicionar la información económica	datos obligatorios	
Se guardan los datos introducidos en cada uno de los campos del formulario	información económica	El sistema muestra un mensaje indicando que la acción se ha realizado satisfactoriamente. Se muestra el listado de proyectos.
Resultado de la prueba: Satisfactorio		

Casos de prueba de aceptación		
Código: 10	**Historia de Usuario (Nro. y nombre):** HU #03 Gestionar información económica.	
Funcionalidad que se prueba: Modificar información económica.		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista. Debe existir al menos un proyecto adicionado con anterioridad.		
Acción	**Datos de entrada**	**Resultado esperado**

Se selecciona la opción Evaluación		Se despliega el menú, mostrando la opción Información económica
Se selecciona la opción Información económica		Se muestra el listado de proyectos que han sido adicionados con anterioridad. Además se muestran las acciones a realizar sobre cada proyecto
Se selecciona el proyecto a modificar la información económica. Se selecciona la opción modificar	proyecto a modificar información económica	El sistema muestra un formulario con toda la información económica del proyecto
Se introducen los nuevos valores para actualizar los campos que se desean modificar	valores de los campos que se modifican	
Se selecciona la opción modificar	Información económica modificada	El sistema muestra un mensaje indicando que se ha realizado la operación de forma satisfactoria. Se muestra el listado con todos los proyectos.
Resultado de la prueba: Satisfactorio		

Casos de prueba de aceptación		
Código: 13	**Historia de Usuario (Nro. y nombre):** HU #04 Gestionar proceso de evaluación.	
Funcionalidad que se prueba: Adicionar proceso de evaluación.		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción Evaluación		El sistema despliega el menú evaluación, mostrando la opción Proceso de evaluación

Se selecciona la opción Proceso de evaluación		El sistema muestra los procesos de evaluación que han sido adicionados con anterioridad. Además se muestran las acciones a realizar
Se selecciona la opción Adicionar	proceso de evaluación a adicionar	Se muestra un formulario para seleccionar los proyectos, métodos, fechas y estado del proceso de evaluación
Se selecciona la opción Adicionar	proceso de evaluación	Se adiciona el proceso de evaluación, se muestra un mensaje indicando que el proceso de evaluación se ha adicionado satisfactoriamente
Resultado de la prueba: Satisfactorio		

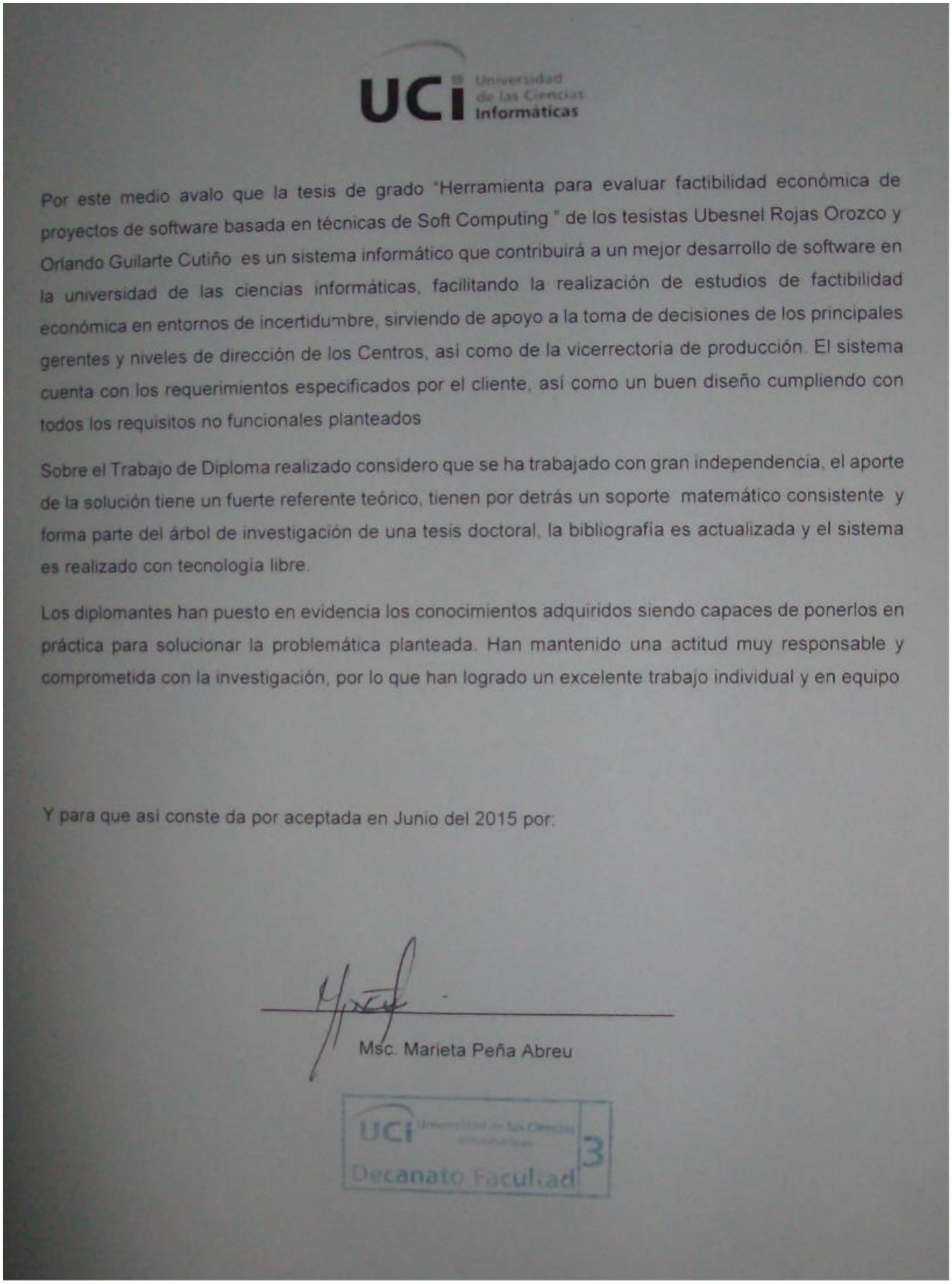
Casos de prueba de aceptación		
Código: 18	**Historia de Usuario (Nro. y nombre):** HU #05 Gestionar proyecto a evaluar durante una evaluación	
Funcionalidad que se prueba: Adicionar proyecto a evaluar durante una evaluación		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción Evaluación		El sistema despliega el menú evaluación, mostrando la opción Proceso de evaluación
Se selecciona la opción Proceso de evaluación		El sistema muestra los procesos de evaluación que han sido adicionados con anterioridad. Además se muestran las acciones a realizar
Se selecciona la opción Adicionar	Proceso de evaluación a adicionar	Se muestra un formulario para seleccionar los proyectos, métodos, fechas y estado del proceso de evaluación

Se seleccionan los proyectos a adicionar al proceso de evaluación	proceso de evaluación	
Se selecciona la opción Adicionar		Se muestra un mensaje confirmando que la acción se ha realizado satisfactoriamente
Resultado de la prueba: Satisfactorio		

Casos de prueba de aceptación		
Código: 26	**Historia de Usuario (Nro. y nombre):** HU #08 Generar reportes	
Funcionalidad que se prueba: Generar reporte de un proyecto		
Condiciones de ejecución: El usuario autenticando debe tener rol de decisor. Debe existir al menos un proyecto evaluado con anterioridad.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción Generar reportes de: Análisis de factibilidad de un proyecto		Se muestra una ventana para seleccionar el proyecto a realizar el reporte
Se selecciona el proyecto a realizar reporte	proyecto	Se muestra una ventana para seleccionar la opción a realizar
Se selecciona la opción guardar archivo	proyecto	Se muestra una ventana para guardar el documento PDF
Se guarda el archivo	proyecto	
Resultado de la prueba: Satisfactorio		

Casos de prueba de aceptación		
Código: 30	**Historia de Usuario (Nro. y nombre):** HU #09 Mostrar análisis de factibilidad	
Funcionalidad que se prueba: Mostrar análisis de factibilidad económica		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista. Debe existir al menos un proyecto evaluado.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción Análisis		Se despliega el menú, se muestra la opción Listado de proyectos evaluados
Se selecciona la opción Listado de proyectos evaluados		Se muestran todos los proyectos evaluados, así como el resultados de los métodos VAN, TIR, PR.
Resultado de la prueba: Satisfactorio		

Anexo 6: Certificado de Aceptación del Producto.



UCI Universidad de las Ciencias Informáticas

Por este medio avalo que la tesis de grado "Herramienta para evaluar factibilidad económica de proyectos de software basada en técnicas de Soft Computing " de los tesistas Ubesnel Rojas Orozco y Orlando Guilarte Cutiño es un sistema informático que contribuirá a un mejor desarrollo de software en la universidad de las ciencias informáticas, facilitando la realización de estudios de factibilidad económica en entornos de incertidumbre, sirviendo de apoyo a la toma de decisiones de los principales gerentes y niveles de dirección de los Centros, así como de la vicerrectoría de producción. El sistema cuenta con los requerimientos especificados por el cliente, así como un buen diseño cumpliendo con todos los requisitos no funcionales planteados

Sobre el Trabajo de Diploma realizado considero que se ha trabajado con gran independencia, el aporte de la solución tiene un fuerte referente teórico, tienen por detrás un soporte matemático consistente y forma parte del árbol de investigación de una tesis doctoral, la bibliografía es actualizada y el sistema es realizado con tecnología libre.

Los diplomantes han puesto en evidencia los conocimientos adquiridos siendo capaces de ponerlos en práctica para solucionar la problemática planteada. Han mantenido una actitud muy responsable y comprometida con la investigación, por lo que han logrado un excelente trabajo individual y en equipo

Y para que así conste da por aceptada en Junio del 2015 por:

Msc. Marieta Peña Abreu

UCI Universidad de las Ciencias Informáticas
Decanato Facultad 3