Universidad de las Ciencias Informáticas

Facultad 3

Herramienta para evaluar la factibilidad técnica y comercial de proyectos de software mediante la computación con palabras

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

Autores: Yadira García García

Alejandro David Álvarez Sosa

Tutores: M. Sc. Marieta Peña Abreu

Ing. Carlos Rafael Rodríguez Rodríguez

La Habana, junio de 2015

"Año 57 de la Revolución"

Pensar es el trabajo más difícil que existe. Quizá sea esta la razón por la que tan pocas personas lo practican.

Henry Ford

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 _____ del año _____.

Yadira García García

_____

Firma del Autor

Alejandro David Álvarez Sosa

_____

Firma del Autor

M. Sc. Marieta Peña Abreu

_____

Firma del Tutor

Ing. Carlos Rafael Rodríguez Rodríguez

_____

Firma del Tutor

DATOS DE CONTACTO

Autor: Yadira García

Correo electrónico: yggarcia@estudiantes.uci.cu

Autor: Alejandro David Álvarez Sosa

Correo electrónico: adsosa@estudiantes.uci.cu

Tutor: Marieta Peña Abreu

Correo electrónico: mpabreu@uci.cu

Tutor: Carlos Rafael Rodríguez Rodríguez

Correo electrónico: crrodriguez@uci.cu

Agradecimientos

A *mi abuelo, que esté donde esté sé que me cuida, por no haber podido ver mi sueño hecho realidad.*

A *mi papá, motor impulsor de este sueño y sin el cual no me hubiese graduado hoy en esta universidad, por no permitir que abandonara la oportunidad de estudiar en la UCI, gracias por todo.*

A *mi mejor amiga, mi madre, por sus consejos, sus regaños, por estar siempre ahí para mí incondicionalmente, gracias por todo lo que me has enseñado en esta vida.*

A mi hermanita, por quererme tanto y estar orgullosa de mí.

A *mis hermanos, por su amor y cariño, por cuidarme siempre, por estar presente cuando los he necesitado.*

A *mi abuela querida, a mi tía, a mi primo, a toda mi familia, por su apoyo y amor.*

A *mi amiga de toda una vida, Gretell, por ser como una hermana para mí, por todo su cariño, por sus locuras y por hacerme reír tanto, por ser incondicional.*

A *mi compañero de tesis, por siempre estar incondicionalmente para mí, por compartir conmigo esta difícil tarea de sacrificio y esfuerzo, por soportarme en los momentos de estrés.*

A *las amistades que a lo largo de mi vida he hecho, las de la secundaria, las del politécnico.*

A *las amistades con las que he compartido estos últimos 5 años de mi carrera, gracias por aguantar mis pesadeces, en especial a Lian y Lisi.*

A *Roniel por toda su ayuda a lo largo de mi carrera, por su apoyo, por contribuir con mi formación profesional.*

A *Roly, Ubesnel, Roberto, el tropa, por aguantarme cuatro años seguidos prácticamente las 24 horas del día y a Orlando y al hacker por quererme tanto y poder contar siempre con ellos.*

A *mis compañeros del actual 3501, y a los que hoy no se encuentran en esta universidad.*

A *todos los profesores que he tenido en la carrera y han contribuido a mi formación profesional.*

A *mi tutora y profesora en estos últimos 4 años de la carrera, por todos sus consejos, su apoyo y ayuda en la realización de esta tesis.*

A *mi tutor por sus consejos, ayuda y apoyo en mis momentos de estrés.*

A *la profe Mailen, por toda su ayuda en estos últimos meses.*

Al *tribunal por las recomendaciones y señalamientos.*

A *mí misma por siempre tener confianza en mí y sacrificarme para lograr mis metas.*

Yadira García García

Agradecimientos

A *mis padres, por todo el apoyo que siempre me han dado. Lo mejor de mi vida. Mis ídolos, como personas y como profesionales.*

A *mi hermano Frank que a pesar de sus locuras, siempre tiene tiempo para atender las mías.*

A *mis tías, que siempre me ayudaron y apoyaron en todo momento, en especial a Teresa. Mi madre en la Habana que siempre me recibe con los brazos abiertos en su casa. A Cristofer, mi hermanito menor.*

A *mis compañeros de aula, que hicieron que estos 5 años en la Universidad fueran como estar en mi casa con mi familia. En especial a mis hermanos, pues los considero como tal, Alejandro "Ing Brayan" Rodríguez Fernández mi hermano en la UCI desde primer año, Manuel González Flores alias "Mala Mia Navi Pro" con quien compartí momentos que siempre recordaré. Miguel Angel, Rigo, Pablo, todo el piquete del Dota y del doble.*

A *los que por un motivo u otro ya no se encuentran pero siempre serán especiales Héctor, Guzmán, Río, Ángel, Pumarino.*

A *mi compañera de tesis, Yadira, por aguantar todas mis rabietas y nunca perder la calma conmigo.*

A *mis tutores Marieta y Carlos, por su dedicación y ayuda.*

A *todos mis profesores durante la carrera. Ellos hicieron posible mi formación como persona e ingeniero.*

Alejandro David Álvarez Sosa

Dedicatoria

A mi abuelo, que me cuida desde el cielo.

A mi papá, motor impulsor de este sueño.

A mi mamá, por ser mi gran amiga y ejemplo a seguir.

A mis hermanos, por su apoyo y cariño.

Yadira García García

A mis padres por todo su apoyo.

A mi hermano por ser incondicional.

Alejandro David Álvarez Sosa

RESUMEN

La exigencia de desarrollar un producto con eficiencia y calidad conlleva a la necesidad de realizar estudios de factibilidad que garanticen una correcta selección de proyectos. Tradicionalmente, estos estudios, han sido realizados por expertos en el tema, mediante métodos tradicionales de evaluación. Por lo general en estos estudios solo se evalúan criterios económicos, sin tener en cuenta criterios técnicos y comerciales, siendo estos importantes para el éxito de los proyectos. La mayoría son realizados en entornos de incertidumbre, donde la información disponible de las diferentes alternativas a evaluar puede ser incompleta, vaga o imprecisa, por lo que debe ser valorada cualitativamente. Actualmente en la UCI, la Dirección para la Comercialización y los Negocios (DCN) se encuentra en un perfeccionamiento de los procesos, métodos y medios que utiliza para el estudio y selección de nuevos proyectos. El presente trabajo tiene como objetivo desarrollar una herramienta informática que emplee la computación con palabras para tratar la incertidumbre presente en las valoraciones de los expertos durante el análisis de la factibilidad técnica y comercial de proyectos de software.

El sistema fue desarrollado utilizando herramientas y tecnología libres, como metodología de desarrollo se empleó Programación Extrema (XP), el marco de trabajo PHP Symfony2 y el lenguaje PL/PgSQL para la implementación de los operadores de agregación del modelo lingüístico 2-tuplas. La solución desarrollada contribuye al análisis de factibilidad técnica y comercial de proyectos de software con tratamiento de la incertidumbre.

Palabras clave: estudios de factibilidad, factibilidad técnica, factibilidad comercial, incertidumbre.

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

INTRODUCCIÓN

El avance de la informática hoy en día ha conllevado al desarrollo e informatización de gran parte de los procesos de la sociedad actual. Esto trae consigo un aumento en la cantidad de proyectos en el mundo del software, donde resulta necesario destinar esfuerzos y recursos, estos pueden clasificarse en humanos, físicos, técnicos y financieros. La planificación, supervisión y control de los recursos es llevada a cabo mediante la gestión de proyectos, área interdisciplinaria del conocimiento. Resulta necesario establecer un conjunto de indicadores que aseguren una correcta toma de decisiones y hagan posible disminuir el riesgo de equivocarse al decidir la ejecución de un proyecto.

La gestión de proyectos ha ido evolucionando con el avance de las tecnologías, las comunicaciones y los productos. El incremento de proyectos en el sector informático se vuelve cada día más competitivo y exigente, por lo que resulta necesario desarrollar soluciones con mayor eficiencia y una elevada calidad, evitando su fracaso. Para lograr el éxito de un proyecto eficiente y con calidad es necesario realizar estudios de factibilidad antes de comenzar su ciclo de vida. El estudio de factibilidad es el paso más crítico antes de convertir la idea del negocio en realidad e invertir una cantidad de dinero significativa (Vega, 2015). El estudio de factibilidad formaliza, documenta y revalida la idea del negocio propuesto, reduciendo el riesgo asociado a tomar una decisión de inversión, sin embargo es importante aclarar que no es una garantía de éxito.

En la realización de los estudios de factibilidad intervienen numerosos factores que deben ser analizados desde diferentes aristas. Tradicionalmente han sido realizados por expertos en el tema, que aplicando métodos tradicionales de evaluación, toman la decisión final a partir del análisis de los resultados. Actualmente, en estos estudios por lo general solo se evalúan criterios económicos, en tanto se olvidan los criterios técnicos y comerciales, siendo estos también de vital importancia en el éxito de los proyectos.

El estudio de factibilidad técnica comprende áreas que estudian la viabilidad de los insumos, la tecnología y los recursos humanos que son necesarios para garantizar la producción en los plazos acordados y según los costos establecidos. Estos estudios dependen de las posibilidades de la tecnología instalada, las herramientas informáticas, los materiales e insumos necesarios y una planificación adecuada del tiempo y de los recursos humanos requeridos (Hernández, 2010).

Por su parte, el estudio comercial es fundamental para determinar la posible aceptación que tendrá un proyecto. Se debe caracterizar el producto, definir el mercado potencial, conocer las características de los productos competidores, estudiar la oferta, la demanda y el posible precio de la venta. La ejecución

de un proyecto puede ser por demanda de un cliente o a riesgo. Cuando la producción es a riesgo hay que prestarle principal atención al estudio comercial, pues este puede decidir si se ejecuta o no. (Hernández, 2010)

Adicionalmente los estudios de factibilidad por lo general se realizan en entornos de incertidumbre. Esto se manifiesta cuando la información disponible sobre las distintas alternativas a evaluar puede ser incompleta, vaga o imprecisa, lo que implica que la evaluación asignada a cada alternativa tenga que ser valorada de forma cualitativa. Esta incertidumbre surge a raíz del intento de modelar la imprecisión propia del comportamiento humano o la inherente a ciertos fenómenos que por su naturaleza son inciertos (Veliz, 2014).

En el contexto de la industria cubana del software, la Universidad de las Ciencias informáticas (UCI) aparece como una institución de características muy peculiares dado su novedoso modelo de universidad-empresa (Romillo Tarke, et al., 2013). La red de centros de desarrollo de software (RCDS) ejecuta anualmente un número considerable de proyectos para clientes nacionales y foráneos. Para las tareas de contratación y otras afines, la RCDS se auxilia de la Dirección para la Comercialización y los Negocios (DCN). La DCN es la encargada de guiar las primeras actividades del ciclo de vida del proyecto; esto comprende la etapa de estudio preliminar y dentro de ella los estudios de factibilidad. Actualmente esta oficina se encuentra en un perfeccionamiento constante de los procesos, métodos y medios que utiliza para el estudio y selección de nuevos proyectos.

Para conocer con mayor claridad cómo se lleva a cabo el proceso de análisis de factibilidad técnica y comercial en la RCDS de la UCI, se realizaron entrevistas a especialistas (Cleger, 2015) (Rodriguez, 2015) (Vega, 2015) que laboran en esta área. Las principales ideas recopiladas se resumen a continuación:

- En un alto por ciento los proyectos que comienzan no son antecedidos de un estudio de factibilidad profundo.
- Los principales especialistas y gerentes de proyectos que realizan estos estudios de factibilidad carecen de conocimiento de la existencia de modelos y herramientas para realizaros.
- En caso de realizarse un estudio de factibilidad se hace fundamentalmente en el área económica.
- Generalmente la información disponible sobre las alternativas a evaluar es incompleta, vaga e imprecisa, lo que provoca cierto grado de incertidumbre en las decisiones que se toman.

Por otra parte, las herramientas existentes en el mercado (DECIDE, 2013) (EasyPlanEx, 2014) (Choice, 2015) para realizar estos estudios de factibilidad son privativas y no evalúan en su totalidad criterios técnicos y comerciales. En investigaciones anteriores realizadas en la UCI (Castro, 2010) se propone la aplicación de un grupo de criterios técnicos y comerciales, pero estos son evaluados por métodos deterministas, haciendo poco tratamiento de la inconsistencia de las opiniones de los expertos. Posteriormente en Xedro-GESPRO (Piñero Pérez, et al., 2013) principal herramienta de gestión de proyectos de la universidad, se implementan un grupo de funcionalidades para realizar estudios de factibilidad pero solamente en el área económica.

Teniendo en cuenta lo antes mencionado, queda definido el siguiente **problema a resolver:** ¿Cómo dar tratamiento a la incertidumbre presente en las valoraciones de los expertos durante el análisis de la factibilidad técnica y comercial de proyectos de software?

Por lo que se determina el siguiente **objeto de estudio:** El Proceso de desarrollo de software en los estudios de factibilidad de proyectos.

Para la solución del problema descrito anteriormente se define como **objetivo general:** Desarrollar una herramienta informática que emplee la computación con palabras para tratar la incertidumbre presente en las valoraciones de los expertos durante el análisis de la factibilidad técnica y comercial de proyectos de software.

Por lo que se delimita como **campo de acción:** Herramientas informáticas para tratar la incertidumbre en los estudios de factibilidad técnica y económica de proyectos de software

Como **idea a defender** se plantea: Con el desarrollo de una herramienta informática que emplee la computación con palabras se contribuirá al tratamiento de la incertidumbre presente en las valoraciones de los expertos durante el análisis de factibilidad técnica y comercial de proyectos de software.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos:**

- Elaborar el marco teórico referencial de la investigación para sustentar el proceso de desarrollo de software en los estudios de factibilidad de proyectos.
- Realizar el análisis y diseño de la solución propuesta, mediante la metodología de desarrollo de software seleccionada para una mejor organización y aplicación de buenas prácticas en el desarrollo de software.

- Implementar todas las funcionalidades definidas para obtener la herramienta, mediante las tecnologías y herramientas seleccionadas.
- Analizar los resultados de la investigación mediante técnicas de validación y verificación, así como la aplicación de un caso de estudio para satisfacer las necesidades del cliente.

Con el objetivo de resolver el problema y lograr el objetivo planteado, se hizo necesaria la utilización de los siguientes **métodos de investigación**:

Métodos teóricos

- ✓ Histórico-lógico: se empleó para analizar la trayectoria y evolución de los modelos y herramientas existentes para la evaluación de proyectos.
- ✓ Analítico-sintético: permitió, fundamentalmente realizar el estudio teórico de la investigación, haciendo posible la selección de los elementos más importantes en el proceso de desarrollo de la herramienta.
- ✓ Inductivo-deductivo: se utilizó para el razonamiento de la información consultada y llegar así a la obtención de un grupo de conocimientos particulares y generales.
- ✓ Modelación: permitió, junto con la metodología de desarrollo seleccionada, modelar los componentes esenciales para el desarrollo de la herramienta.

Métodos empíricos

- ✓ Entrevista: se utilizó en el intercambio con el cliente para adquirir información sobre el negocio y los requisitos que se deben cumplir en el desarrollo del software. Además en las entrevistas a especialistas que laboran en el área de la RCDS de la UCI.

El presente trabajo de diploma está compuesto por 3 capítulos, estructurados de la siguiente manera:

Capítulo I: Fundamentación teórica

En este capítulo se realiza una revisión y análisis de los modelos, métodos, metodologías y herramientas existentes para evaluar proyectos de software. Se plantea la técnica de soft computing a utilizar, computación con palabras y dentro de esta el modelo lingüístico 2-tuplas. Se describen las principales técnicas de soft computing. Se enuncian los principales conceptos relacionados con el tema, así como un estudio de las metodologías, lenguajes, herramientas y patrones a usar como apoyo para dar solución al problema planteado. Por último se describen las pruebas a realizar para la validación y verificación del sistema.

Capítulo II: Análisis y diseño

En este capítulo se propone la solución de la investigación. Se exponen los requisitos funcionales y no funcionales. Se generan los artefactos del análisis y el diseño. Se describen los patrones de diseño y arquitectónico empleados. Se valida el diseño aplicando las métricas Tamaño operacional de clases y Relaciones entre clases.

Capítulo III: Implementación y prueba

En este capítulo se evalúa el cumplimiento de los objetivos planteados a partir de la realización de pruebas de caja blanca y caja negra al sistema, pruebas de aceptación por parte del cliente y la validación de la solución propuesta a través de la introducción de una base de casos reales a la herramienta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo son abordados los principales conceptos utilizados en la investigación. Se hace un estudio de diferentes modelos, métodos, metodologías y herramientas existentes, para la evaluación de proyectos de software con tratamiento de la incertidumbre en las valoraciones de los expertos durante el análisis de la factibilidad técnica y comercial. Se caracteriza el modelo lingüístico para la realización de procesos de computación con palabras 2-tuplas. Se describen las principales tecnologías, lenguajes de programación y herramientas definidas para el desarrollo de la solución, así como la metodología de desarrollo, los patrones a emplear, las métricas para la validación del diseño y las pruebas para garantizar la calidad del producto final.

1.2 Principales conceptos

Para comprender los conceptos de *factibilidad*, *factibilidad técnica, factibilidad comercial y soft computing* se presentan las siguientes definiciones:

Factibilidad: se realiza al mismo tiempo que se ejecuta el proyecto de acuerdo con los resultados que se obtienen y ayuda a decidir si se realiza la inversión para introducir los resultados. (Hernández, 2002)

Factibilidad técnica: contendrá toda aquella información que permita establecer la infraestructura necesaria para atender su mercado objetivo, así como cuantificar el monto de las inversiones y de los costos de operación de la entidad en formación. (Almaguer, 2009)

Factibilidad comercial: vincula a los consumidores con el encargado de estudiar el mercado a través de la información, la cual se utiliza para identificar y definir tanto las oportunidades como las amenazas del entorno. Por su carácter preliminar, constituye un sondeo de mercado, antes de incurrir en costos innecesarios. (Almaguer, 2009)

Soft computing: especie de consorcio o sociedad entre la lógica borrosa, la neurocomputación y el razonamiento probabilístico, en el que esta última incluye los algoritmos genéticos, el razonamiento evidencial y los sistemas caóticos. Trata problemas relativos a la imprecisión, la incertidumbre y el razonamiento aproximado. (Zadeh, 1996)

1.3 Modelos, métodos y metodologías para estudios de factibilidad

A continuación se estudian algunos modelos, métodos y metodologías propuestos para la realización de estudios de factibilidad en el ámbito nacional e internacional:

Evaluación de proyectos en América

Guía práctica para el diseño, administración y evaluación de proyectos sociales:

Guía básica para el diseño, administración y evaluación de proyectos sociales a fin de perfeccionar el trabajo en las empresas. Brinda fundamental importancia a la creación de indicadores para la medición; son los fundamentales la eficiencia y la eficacia. Esta metodología da importancia al criterio de expertos para poder transmitir sus experiencias. Esta guía permite una flexibilización para seleccionar los criterios a evaluar. No trabaja la incertidumbre de la información. (Rosa, 2005)

Metodología propuesta por el International Project Management Association:

Metodología propuesta por la Administración de Estándares de Portafolio. Permite dividir y evaluar los proyectos en componentes más pequeños, mediante la definición de diferentes fases. Para la realización de la evaluación se apoya en el uso de técnicas y herramientas como el criterio de expertos y representaciones gráficas. Los criterios de evaluación que se proponen se encuentran agrupados por categorías: negocio, beneficio financiero, riesgo, recurso humano, mercado y técnico. Esta metodología no trabaja la incertidumbre de la información. (IPMA, 2006)

Evaluación de proyectos en Cuba

Método de evaluación de proyectos para decidir su aceptación:

Método para evaluar la factibilidad técnica, económica y comercial de los proyectos de software basado en criterios de expertos y estudios de factibilidad económica, lo que apoyado en análisis estadísticos y herramientas informáticas, permite organizar una cartera de proyectos factibles con alta probabilidad de éxito. El método determina el índice de concordancia de la opinión de los expertos, siendo completamente determinista. No tiene en cuenta la incertidumbre de la información. (Castro, 2010)

Modelo para análisis de factibilidad en la evaluación de proyectos de software:

Modelo para realizar análisis de factibilidad en los órdenes técnico, económico y comercial, a proyectos de software, basado en técnicas de soft computing, para evaluarlos desde la plataforma de Gestión de Proyectos GESPRO. El modelo contiene dos fases: iniciación y evaluación, en las cuales a partir de criterios técnicos, económicos y comerciales, se aplican métodos de evaluación para finalmente, a través de un sistema de inferencia borroso, obtener como salida un listado de proyectos evaluados, según el orden de factibilidad arrojado por la herramienta. Este modelo no es determinista y tiene en cuenta la incertidumbre de la información. (Abreu, 2012)

Modelo para la toma de decisiones en los proyectos de software basado en los criterios de expertos:

Modelo para la toma de decisiones en los proyectos de software basado en los criterios de expertos, para ayudar en la selección de alternativas y criterios en los análisis de factibilidad, selección de recursos humanos y priorización de proyectos en la plataforma de gestión de proyectos GESPRO

v13.05. El modelo contiene seis etapas, donde a partir de las evaluaciones de los expertos, se aplican métodos matemáticos para la comparación de las diferentes alternativas y la selección de las mejores. Este modelo es determinista y no tiene en cuenta la incertidumbre de la información. (Palenzuela, 2013)

Valoración

Los modelos, métodos y metodologías analizados para la realización de estudios de factibilidad evalúan principalmente criterios económicos, solo algunos de estos tienen en cuenta criterios técnicos y comerciales. Por lo general participan expertos, emitiendo sus valoraciones en las evaluaciones de diferentes alternativas, existiendo incertidumbre en estas. En su mayoría son deterministas. Solamente el modelo propuesto por (Abreu, 2012) tiene en cuenta la incertidumbre y evalúa además criterios técnicos y comerciales, conjuntamente con el método propuesto por (Castro, 2010) y la metodología propuesta por el International Project Management Association. Estos modelos, por separado, no cumplen con el objetivo de la investigación de tratar la incertidumbre presente en las valoraciones de los expertos durante el análisis de la factibilidad técnica y comercial de proyectos de software.

1.4 Herramientas informáticas para estudios de factibilidad

Se estudiaron además herramientas informáticas dedicadas a facilitar la toma de decisiones y los estudios de factibilidad. A continuación se realiza un análisis de algunas de ellas.

DECIDE

Software especializado que facilita la elaboración de planes de negocios, formulación y evaluación de proyectos de inversión, proporciona metodologías y herramientas para una toma de decisiones sustentada. DECIDE ha sido desarrollado por la empresa mexicana Soluciones Informáticas y Aplicaciones Crediticias S. A. de C.V. Permite realizar evaluación de un proyecto de inversión, elaborar estudios de mercado, análisis técnicos, económicos, financieros y de riesgos. Es una herramienta privativa, por lo que tiene costo de licencia para su uso. (DECIDE, 2013)

Expert Choice

Software para la toma de decisiones, basado en el Proceso Jerárquico Analítico (AHP, Analytic Hierarchy Process). Asiste a los decisores organizando la información relacionada con la complejidad del problema en un modelo jerárquico que consta de un objetivo, escenarios posibles, criterios y alternativas. Ha sido usado exitosamente en una variedad de aplicaciones incluyendo, priorización y evaluación de proyectos, planeamiento estratégico y análisis de costo/beneficio. Es una herramienta privativa, por lo que tiene costo de licencia para su uso. (Choice, 2015)

EasyPlanEx

Software que provee una solución integral para evaluar y optimizar la formulación de proyectos de inversión desarrollado por BoraSystems. Realiza análisis de riesgo, utiliza la técnica de Montecarlo[1], emplea diferentes distribuciones de probabilidades. Permite calcular la probabilidad de ocurrencia de un resultado. Realiza análisis de sensibilidad en forma automática, genera los resultados para todos los escenarios posibles. Además, permite optimizar un proyecto, encontrando la mejor alternativa. Es una herramienta privativa por lo que tiene costo de licencia para su uso (EasyPlanEx, 2014).

Gespro

Paquete de Gestión de Proyectos desarrollado por la UCI, registrado en el Centro Nacional de Derecho de Autor (CENDA) con el No. de Registro 1540-2010. Abarca varias áreas de la Gestión de Proyectos, las cuales están presentes en el sistema mediante funcionalidades y módulos. Entre estas se encuentran: la gestión de portafolios de proyectos, la gestión del alcance de productos, la gestión del tiempo, la gestión de riesgos de proyectos, la gestión de comunicaciones, la gestión de recursos humanos y materiales y la gestión documental. Permite realizar estudios de factibilidad económica, a una cartera de proyectos, y obtener finalmente un listado de proyectos ordenados según su factibilidad (Piñero Pérez, et al., 2013)

Valoración de las herramientas

Luego de realizado el estudio de las herramientas de evaluación de proyectos mencionadas, se muestra una tabla comparativa entre estas herramientas y los indicadores definidos.

Tabla 1. Comparación de las herramientas de evaluación de proyectos estudiadas.

Herramientas	Evalúan criterios técnicos y comerciales	Trabajan la incertidumbre	Licencia
EasyPlanEx	No	Sí	Privativa
Expert Choice	No	No	Privativa
Decide	Sí	No	Privativa
Gespro	Sí	No	Libre

Las herramientas analizadas facilitan la toma de decisiones en el momento de evaluar un proyecto, lo cual constituye una de sus ventajas. Como se puede evidenciar en la tabla comparativa, en su mayoría son propietarias por lo que es necesario pagar una licencia para su uso, solamente dos de estas tienen en cuenta la factibilidad técnica y comercial, elementos de gran importancia cuando se evalúa la

[1]El método de MonteCarlo es una técnica numérica para calcular probabilidades y otras cantidades relacionadas, utilizando secuencias de números aleatorios.

factibilidad de un proyecto, pero no tratan la incertidumbre de la información, siendo estas sus principales desventajas. Estas herramientas no satisfacen completamente las necesidades que posee actualmente la UCI de evaluar técnica y comercialmente un proyecto de software antes de darle comienzo. Por lo anteriormente expuesto, se propone el desarrollo de una herramienta informática para evaluar la factibilidad técnica y comercial de proyectos bajo condiciones de incertidumbre, haciendo uso de la computación con palabras.

1.5 Técnicas de Soft Computing

Las técnicas de soft computing proponen resolver problemas asociados a la toma de decisiones. Son recomendables en entornos donde exista incertidumbre en la información.

Computación con palabras

La computación con palabras, en lo adelante CWW[2], es una técnica sobre la base de la lógica difusa donde las palabras se utilizan en lugar de números (Zadeh, 1996). Se presenta actualmente como uno de los paradigmas para el tratamiento de la incertidumbre y la información lingüística (Martínez, 2010) (Martínez, 2012).

Modelo lingüístico 2-tuplas

Entre los modelos existentes para realizar procesos de CWW se encuentra el modelo lingüístico 2-tuplas, este es factible para el tratamiento de la incertidumbre. El uso de la información lingüística hace más fiable los modelos decisionales bajo ambientes de incertidumbre.

1.6 Modelo lingüístico 2-tuplas

Para realizar el análisis se deben identificar un conjunto de proyectos $P = \{p_j \mid j \in (1, \dots, n)\}$, los cuales constituyen la entrada al proceso de análisis donde se evalúan para determinar aquellos que son más factibles técnica y comercialmente. La evaluación es realizada por un conjunto de expertos $E = \{e_i \mid i \in (1, \dots, m)\}$ que analizan los proyectos. La evaluación de los proyectos se realiza a partir de un conjunto de criterios técnicos y comerciales $C = \{c_k \mid k \in (1, \dots, p)\}$. Estos criterios tendrán diferentes pesos asignados de acuerdo a su importancia en el análisis de los proyectos por lo cual se define un vector de pesos $W^c = (W^{c_1}, \dots, W^{c_p})$. Para expresar las preferencias de los expertos se utiliza el vector de utilidad $X = \left(x_j^{ki}, \dots, x_j^{ki}\right)$, donde x_j^{ki} representa la preferencia del experto e_i sobre el proyecto e_j de acuerdo al criterio c_k. Los expertos podrán emitir sus preferencias a través de valores lingüísticos (S): $x_j^{ki} = s_j^{ki} \in S = \{S_0, \dots, S_g\}$ siendo g+1 la cardinalidad del Conjunto de Términos Lingüísticos (CTL)

[2] **CWW:** Acrónimo en inglés de Computing With Words

S, es decir, la cantidad de términos de S. Cada término lingüístico s_i tiene asociada una función de pertenencia [0; 1].

Recopilación de las preferencias de los expertos

En esta etapa los expertos proveen sus valoraciones a través de vectores de preferencia: $X = (x_j^{ki}, \dots, x_j^{ki})$ que pueden recopilarse como se muestra en la tabla:

Tabla 2. Preferencias de los expertos.

Proyectos	Criterios	Expertos		
		e_1	…	e_m
p_1	c_1	x_1^{11}	…	x_1^{1m}
	…	…	…	…
	c_k	x_1^{k1}	…	x_1^{km}
p_2	c_1	x_2^{11}	…	x_2^{1m}
	…	…	…	…
	c_k	x_2^{k1}	…	x_2^{km}
p_n	c_1	x_n^{11}	…	x_n^{1m}
	…	…	…	…
	c_k	x_n^{k1}	…	x_n^{km}

Análisis de los proyectos

Transformación de la entrada en un conjunto difuso:

El conjunto difuso que representa un término lingüístico s_i será en todos 0 excepto en el valor correspondiente al ordinal, i, de la etiqueta lingüística que será uno. Por ejemplo, para la etiqueta Muy Alto, en el CTL S= {Muy Bajo, Bajo, Medio, Alto, Muy Alto}, el conjunto difuso que la representa es (0, 0, 0, 0, 1).

Transformación de los conjuntos difusos en 2-tuplas:

El modelo de representación basado en 2-tuplas parte del concepto de traslación simbólica. La traslación simbólica de un término lingüístico es un número valorado en el intervalo [-0.5, 0.5) que representa la "diferencia de información" entre una cantidad de información expresada por el valor $\beta \in [0, g]$ obtenido en una operación simbólica y el valor entero más próximo $i \in \{0,\dots, g\}$ que indica el índice de la etiqueta lingüística (S_i) más cercana en S. Partiendo de este concepto, (Martínez, 1999) desarrolla un modelo de representación para la información lingüística que utiliza como base la

representación 2-tuplas, (s_a,α_a), $s_a \in S$ y $\alpha_a \in [-0.5,0.5)$, - donde: S_a representa la etiqueta lingüística, y α_a es un número que expresa el valor de la distancia desde el resultado original al índice de la etiqueta lingüística más cercana en el conjunto de términos lingüísticos, es decir, su traslación simbólica.

Tomando en consideración estos conceptos se utiliza la siguiente función para transformar los conjuntos difusos ya obtenidos en 2-tuplas lingüísticas (Herrera, 2005)

$$x\big(F(S_t)\big) = x\big(\{(S_j,\gamma_j), j = 0,\dots,g\}\big) = \frac{\sum_{j=0}^{g} j\gamma_j}{\sum_{j=0}^{g}\gamma_j} = \Delta\beta = (s_i,\alpha)$$

Tabla 3. Preferencia de los expertos expresados en 2-tuplas.

Proyectos	Criterios	Expertos		
		e_1	…	e_m
p_1	c_1	$(s_a,\alpha_a)_1^{11}$	…	$(s_a,\alpha_a)_1^{1m}$
	…	…	…	…
	c_k	$(s_a,\alpha_a)_1^{k1}$	…	$(s_a,\alpha_a)_1^{km}$
p_2	c_1	$(s_a,\alpha_a)_2^{11}$	…	$(s_a,\alpha_a)_2^{1m}$
	…	…	…	…
	c_k	$(s_a,\alpha_a)_2^{k1}$	…	$(s_a,\alpha_a)_2^{km}$
p_n	c_1	$(s_a,\alpha_a)_n^{11}$	…	$(s_a,\alpha_a)_n^{1m}$
	…	…	…	…
	c_k	$(s_a,\alpha_a)_n^{k1}$	…	$(s_a,\alpha_a)_n^{km}$

Agregación:

El modelo de representación de la información lingüística es soportado por un modelo computacional basado en las funciones Δ y Δ^{-1} , que transforman valores numéricos en 2-tuplas y viceversa sin pérdida de información por lo que los operadores de agregación numéricos tradicionales pueden extenderse a 2-tuplas de forma sencilla con el fin de obtener resultados precisos y proporcionar una representación que facilite su interpretación. Estos operadores serán utilizados para obtener el valor colectivo de cada criterio para cada proyecto a partir de la agregación de las preferencias de todos los expertos en 2-tuplas y para obtener la factibilidad que tiene cada proyecto a partir de la agregación de las preferencias de todos sus criterios.

Operador de agregación:

Media Aritmética Extendida:

Este operador permite determinar el punto de equilibrio o centro del conjunto de valores. Para un conjunto de 2-tuplas $x = \{(s_1,\alpha_1),\dots,(s_n,\alpha_n)\}$*, la extensión de este operador se obtiene de la siguiente manera:*

$$x^{-e}(x) = \Delta\left(\frac{1}{n}\sum_{i=1}^{n}\Delta^{-1}((s_i,\alpha_i))\right) = \Delta\left(\frac{1}{n}\sum_{i=1}^{n}\beta_i\right)$$

Ecuación 1. Operador Media Aritmética Extendida

OWA (Ordered Weighted Averaging):

Este operador fue introducido por (Yager, 1988), es un operador de agregación ponderado, en el cuál, los pesos no están asociados a un valor predeterminado sino a una posición determinada. De manera que si se tiene un conjunto de 2-tuplas $x = \{(s_1,\alpha_1),\dots,(s_n,\alpha_n)\}$ y $W = (w_1,\dots,w_n)$ es su vector de pesos asociado tal que $w_i \in [0,1]$ y $\sum w_i = 1$, la extensión del operador se obtiene como se muestra:

$$OWA(x) = \Delta\left(\sum_{i=1}^{n} w_i * \beta_i\right)$$

Ecuación 2. Extensión del operador OWA

Donde β_i es el i-ésimo mayor valor de los $\Delta^{-1}(s_i,\alpha_i)$.

Calcular el valor colectivo de cada criterio para cada proyecto:

Como se pretende manejar las preferencias de múltiples expertos, es necesario determinar el valor colectivo de los criterios para cada proyecto. Se asume que todos los expertos tienen el mismo peso en la evaluación. Este valor colectivo se podrá obtener de la siguiente manera:

$$(s_b,\alpha_b)_j^k = \bar{x}^e\left((s_a,\alpha_a)_j^{k1},\dots,(s_a,\alpha_a)_j^{km}\right)$$

Tabla 4. Valores colectivos de los criterios para cada proyecto.

Proyectos	Criterios	Expertos			Valores colectivos x criterios
		e_1	…	e_m	
p_1	c_1	$(s_a,\alpha_a)_1^{11}$	…	$(s_a,\alpha_a)_1^{1m}$	$(s_b,\alpha_b)_1^1$
	…	…	…	…	…
	c_k	$(s_a,\alpha_a)_1^{k1}$	…	$(s_a,\alpha_a)_1^{km}$	$(s_b,\alpha_b)_1^k$
p_2	c_1	$(s_a,\alpha_a)_2^{11}$	…	$(s_a,\alpha_a)_2^{1m}$	$(s_b,\alpha_b)_2^1$
	…	…	…	…	…
	c_k	$(s_a,\alpha_a)_2^{k1}$	…	$(s_a,\alpha_a)_2^{km}$	$(s_b,\alpha_b)_2^k$
p_n	c_1	$(s_a,\alpha_a)_n^{11}$	…	$(s_a,\alpha_a)_n^{1m}$	$(s_b,\alpha_b)_n^1$
	…	…	…	…	…
	c_k	$(s_a,\alpha_a)_n^{k1}$	…	$(s_a,\alpha_a)_n^{km}$	$(s_b,\alpha_b)_n^k$

Calcular factibilidad de cada proyecto:

Obtenido el valor colectivo de los criterios para cada proyecto, se procede a determinar la factibilidad de cada proyecto. Teniendo en cuenta el peso de cada criterio (w^{c_k}), esta factibilidad se puede obtener como se muestra:

$$(s_c, \alpha_c)_j = \bar{x}^e\left((s_b, \alpha_b)_j^1, \dots, (s_b, \alpha_b)_j^n\right)$$

Tabla 5. Valores colectivos de los proyectos.

Proyectos	Valores colectivos de los proyectos
$\boldsymbol{p_1}$	$(s_c, \alpha_c)_1$
$\boldsymbol{p_2}$	$(s_c, \alpha_c)_2$
…	…
$\boldsymbol{p_n}$	$(s_c, \alpha_c)_n$

Interpretación de los resultados:

Una vez que se tienen los valores de preferencias colectivas de cada uno de los criterios y de cada proyecto, es necesario ordenarlos para poder obtener una adecuada interpretación de los resultados, de modo que los especialistas puedan determinar con facilidad qué proyecto tiene mayor factibilidad. Para realizar este ordenamiento es necesario utilizar operadores de comparación para 2-tuplas como las presentadas en (Herrera, 2000). Estos operadores permiten obtener conjuntos ordenados como parte de la solución del problema.

Criterio de comparación de 2-tuplas:

Para las 2-tuplas (s_k, α_1) y (s_l, α_2) que representan dos valoraciones:

- Si $k > l$ entonces $(s_k, \alpha_1) > (s_l, \alpha_2)$
- Si $k < l$ entonces $(s_k, \alpha_1) < (s_l, \alpha_2)$
- Si $k = l$ entonces:
  - Si $\alpha_1 = \alpha_2$ entonces $(s_k, \alpha_1) = (s_l, \alpha_2)$
  - Si $\alpha_1 < \alpha_2$ entonces $(s_k, \alpha_1) < (s_l, \alpha_2)$
  - Si $\alpha_1 > \alpha_2$ entonces $(s_k, \alpha_1) > (s_l, \alpha_2)$

Como resultado de la comparación se obtiene el listado de los proyectos ordenados según su prioridad.

1.7 Metodología de desarrollo

Dentro del desarrollo de software y con la necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor para los clientes, se hace necesario el empleo de una metodología. La

metodología seleccionada debe ser aquella que mejor se ajuste a las características del equipo de desarrollo y las exigencias de los usuarios finales.

Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las metodologías tradicionales y las metodologías ágiles. Las tradicionales enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto. Son recomendadas para proyectos de grandes dimensiones y con grandes equipos de desarrollo. Las metodologías ágiles dan importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueven el trabajo en equipo (Figueroa, y otros, 2010).

Para poder llevar a cabo el desarrollo de la propuesta de solución en un corto período de tiempo, donde el cliente esté en constante participación y colaboración y poder realizar cambios en los requisitos si fuese necesario, se opta por una metodología ágil de desarrollo de software en lugar de una metodología tradicional.

Se realizó un estudio de las metodologías ágiles XP, SCRUM y AUP, teniendo en cuenta la respuesta a los cambios, el trabajo con el cliente, los planes de entrega de las iteraciones, el trabajo en equipo y el enfoque.

La **Programación Extrema** (*e**X**treme **P**rogramming*, XP) es posiblemente el método ágil más conocido y ampliamente utilizado. En esta las iteraciones de entrega son entre una y tres semanas. Cuando se van finalizando las entregas son susceptibles a modificaciones durante el transcurso de todo el proyecto. El cliente forma parte del equipo de desarrollo, está en constante participación. Los miembros programan en parejas. Se centra más en la propia programación o creación del producto.

La metodología **SCRUM** está enfocada a la gestión de procesos. En esta las iteraciones de entregas son de dos a cuatro semanas. Al finalizar una entrega las tareas que se han realizado no pueden ser modificadas. Se mantiene una estrecha colaboración con el cliente. Cada miembro del equipo trabaja de forma individual. Se centra más en la administración del proyecto.

En la metodología **AUP** la primera producción de liberación puede tomar doce meses, para entregar la segunda versión nueve meses, y luego otras liberaciones se entregan cada seis meses. Permite una gestión de cambios ágil. Se mantiene una estrecha colaboración con el cliente. Se centra especialmente en la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo.

Se selecciona como metodología de desarrollo la **Programación Extrema (XP)** por ser una metodología flexible a los cambios en las entregas, diseñada para equipos de trabajo pequeños donde la programación es por parejas, lo que cumple con las características del equipo de desarrollo. El cliente será partícipe del proceso de desarrollo estando en constante participación. Por parte del cliente se solicita que las entregas por iteraciones sean de un corto período de tiempo, con lo cual cumple esta metodología. Lo más importante tanto para el equipo de desarrollo como para el cliente es el producto final. Se generan los artefactos mínimos para la comunicación con el cliente. Esta consta de 4 fases: planificación, diseño, desarrollo y pruebas.

1.8 Herramientas, tecnologías y lenguajes

Se seleccionó como marco de trabajo Symfony2 pues el equipo de trabajo tenía experiencia en el desarrollo de aplicaciones usando este framework PHP. Debido a que se pretende integrar la solución a la herramienta GESPRO, se seleccionó como Gestor de Base de Datos, PostgreSQL v9.2 y el lenguaje PL/PgSQL para la implementación de los operadores de agregación del modelo lingüístico. Como Entorno de Desarrollo Integrado se seleccionó Netbeans v8.0 ya que es el IDE que mejor integración tiene con el marco de trabajo Symfony2, permitiendo la ejecución de comandos y auto-completamiento de código. A continuación se describen estas herramientas, tecnologías y lenguajes seleccionados:

1.8.1 Lenguaje de modelado

El **Lenguaje Unificado de Modelado** (*Unified Modeling Language*, UML) es un lenguaje de modelado visual usado para especificar, visualizar, construir y documentar artefactos de un software. Capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Brinda apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (Rumbaugh, y otros, 2000)

Se emplea el lenguaje de modelado UML en su versión 2.0 para la confección del diagrama de componentes y del modelo de datos de la solución.

1.8.2 Herramienta de modelado

Visual Paradigm for UML es una herramienta de ingeniería de software asistida por computadora (*Computer Aided Software Engineering*, CASE) que soporta el modelado mediante UML. Proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todo el ciclo de vida de desarrollo de un software. Puede integrarse con otras aplicaciones, como herramientas ofimáticas. Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores

en la codificación del software. Además de generar diversos informes a partir de la información introducida en la herramienta. (Paradigm, 2015).

Se selecciona Visual Paradigm en su versión 8.0 para la creación de los diagramas necesarios para el desarrollo de la solución.

1.8.3 Lenguaje de programación

PHP Hypertext Preprocessor (PHP) es un lenguaje de código abierto del lado del servidor, especialmente adecuado para el desarrollo web de contenido dinámico. Puede ser utilizado en cualquier sistema operativo y servidor web. Posibilita la utilización de programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas (PHP, 2015).

Se selecciona el lenguaje PHP en su versión 5.4, el cual permite:

- Soporte para bases de datos: MySQL, PostgreSQL, Oracle, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader).
- Admite servidores web como Apache.

1.8.4 Entorno de Desarrollo Integrado

Un IDE (*Integrated Development Environment*) es un programa compuesto por un conjunto de herramientas de programación**.** Puede dedicarse a un solo lenguaje de programación o a varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. (Rouse, 2007)

NetBeans es un IDE hecho principalmente para Java, pero puede ser utilizado para cualquier lenguaje de programación. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Es un producto libre y gratuito sin restricciones de uso. Facilita el trabajo mediante auto completamiento de código, visor de clases, métodos y componentes (NetBeans, 2015).

Se selecciona el IDE NetBeans en su versión 8.0. por ser un producto libre, además de brindar facilidades para la implementación de la solución.

1.8.5 Marco de trabajo

Un marco de trabajo o *framework* es un conjunto de componentes físicos y lógicos estructurados de manera que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información.

Permiten la utilización de modelos arquitectónicos como Cuatro Capas y Modelo Vista Controlador (*Model View Controller*, MVC) (Recaman, 2012).

Symfony2 es un marco de trabajo diseñado para optimizar el desarrollo de aplicaciones web. Utiliza el patrón arquitectónico MVC que separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Automatiza tareas comunes, permite al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Está desarrollado completamente con PHP 5. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL y Oracle (Eguiluz, 2012).

Características:

- Extensible: desarrollado en su totalidad alrededor de Bundles, los cuales son directorios que contienen todo tipo de archivos (clases php y archivos web como JavaScript, css e imágenes), dentro de una estructura jerarquizada de directorios.
- Flexible: cuenta con un micro-kernel basado en contenedores de inyección de dependencia y en disparadores de eventos.
- Pensado para desarrolladores: provee herramientas que mejora la productividad de los desarrolladores, como por ejemplo la barra de debug, el soporte nativo para diferentes entornos, los errores y excepciones en páginas detalladas.
- Usabilidad avanzada: posee una simple API (*Interfaz de Programación de Aplicaciones*) que brinda la posibilidad de usar el marco de trabajo con gran facilidad.

Se selecciona como marco de trabajo Symfony2 en su versión 2.5 por las características y facilidades que brinda, además de contar con abundante documentación. Es sencillo y fácil de entender por parte de los programadores.

1.8.6 Sistema Gestor de Base de Datos (SGBD)

Un sistema gestor de base de datos es un conjunto de programas que administran y gestionan la información contenida en una base de datos. Entre las acciones que realiza se encuentran: definición de los datos, mantenimiento de la integridad de los datos dentro de la base de datos control de la seguridad y privacidad de los datos y manipulación de los datos. (SGBD, 2007)

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (*Berkeley Software Distribution*[3]) y con código fuente disponible libremente. Utiliza un modelo

[3] **BSD:** Licencia de software otorgada principalmente para los sistemas BSD. Es una licencia de software libre permisiva.

cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afecta al resto y el sistema continua funcionando. Funciona bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (PostgreSQL, 2009).

Características:

- Completa documentación
- Soporta diferentes tipos de datos
- Disponible para Linux y UNIX en todas sus variantes y Windows 32/64bit.

Se selecciona como sistema gestor de base de datos PostgreSQL en su versión 9.2 por las características antes mencionadas, además de permitir realizar consultas a la base de datos, crear funciones y vistas para el trabajo con el lenguaje PL/PgSQL.

1.8.7 Lenguaje PL/PgSQL (Procedural Language/Postgres SQL Structured Query Language)

Este es un lenguaje procedural cargable provisto por el gestor de base de datos PostgreSQL, el cual puede ser usado para crear funciones y triggers, dispone de estructuras de control repetitivas, puede realizar cálculos complejos y es fácil de usar. Entre sus ventajas se encuentran que tiene soporte para SQL, o sea, con PL/PgSQL se pueden usar todos los tipos de datos, columnas, operadores y funciones de SQL, y su portabilidad. Es utilizado para la implementación de los operadores de agregación del modelo lingüístico (PostgreSQL, 2009).

1.8.7 Servidor web

Un servidor web es un "programa que atiende y responde a las diversas peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o HTTPS[4]" (Mateu, 2004)

Apache es un servidor web de código abierto para la creación de páginas y servicios web. Es un servidor multiplataforma, gratuito, robusto y seguro. (Apache, 2014)

Se selecciona como servidor web Apache en su versión 2.4.4 por ser multiplataforma, cuenta con abundante documentación y soporta varios lenguajes de programación, dentro de los cuales se encuentra el seleccionado para la solución.

[4] HTTPS: versión segura, cifrada y autenticada de HTTP

1.9 Patrones para el desarrollo de software

Según (Larman, 1999) un patrón es una descripción de un problema y su solución, que recibe un nombre y que puede emplearse en otros contextos. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas. Para el diseño y el desarrollo de la solución se hizo uso de un patrón arquitectónico y de un grupo de patrones de diseño.

1.9.1 Patrón arquitectónico

Los patrones arquitectónicos representan el nivel más alto dentro del sistema de patrones y expresan el esquema de la estructura fundamental de la organización para sistemas de software (Almeira, y otros, 2007).

Para el desarrollo de la solución se selecciona **Modelo Vista Controlador** (*Model View Controller*, MVC) como patrón arquitectónico. Este separa los datos y la lógica del negocio de la interfaz de usuario en tres componentes: el modelo, las vistas y los controladores. (Almeira, y otros, 2007)

1.9.2 Patrones de diseño

Los patrones de diseño son una estructura de clases que se presenta en forma repetida en distintos diseños orientados a objetos, la cual es empleada para resolver un problema determinado de manera flexible y adaptable en forma dinámica (Almeira, y otros, 2007). Para el desarrollo del sistema se utilizaron varios patrones de diseño. A continuación se dan a conocer los patrones empleados.

Patrones generales de software para asignación de responsabilidades (GRASP[5])

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos. Constituyen el fundamento de cómo se diseña el sistema (Larman, 1999). A continuación se mencionan los patrones utilizados en el diseño de la solución.

- Experto: asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador**:** guía la asignación de responsabilidades relacionadas con la creación de objetos.
- Bajo acoplamiento: medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras.

[5] GRASP: del inglés General Responsibility Asignment Software Patterns

- Alta cohesión: medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.
- Controlador: objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define el método de su operación.

Banda de los cuatro (GoF[6])

Los patrones GoF pertenecen al campo del Diseño Orientado a Objetos. Están conformados por 23 patrones que se clasifican según su propósito en creacionales, estructurales y comportamiento. (Christiansson, 2008)

- Creacionales: definen la forma en la que un objeto puede ser creado teniendo en cuenta la reutilización y mutabilidad. Estos describen la mejor manera de manejar instancias.
- Estructurales: describen cómo los objetos y las clases se pueden combinar para forman estructuras.
- Comportamiento: describen la forma de cómo organizar, administrar y combinar conductas y responsabilidades de objetos, centrándose en la comunicación entre ellos.

Para el desarrollo de la solución se empleó el patrón creacional Método de Fábrica, los patrones estructurales Adaptador y Decorador, y como patrón de comportamiento Mediador.

Mediante el uso de estos patrones se garantizó la asignación de las responsabilidades correspondientes a cada una de las clases, y el menor número de dependencias y relaciones entre estas, lo que contribuye a un correcto diseño del sistema, y facilita su implementación.

1.10 Métricas

Una métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado. (Pressman, 2010)

1.10.1 Métricas para diseño

Estas métricas cuantifican los atributos del diseño de manera que permiten evaluar su calidad. Dentro de las métricas para el diseño se encuentran las especializadas en diseño orientado a objetos, las cuales miden características de clases. Permiten averiguar cuan bien están definidas las clases y el sistema (Pressman, 2010). Miden de forma cuantitativa la calidad de los atributos internos del producto

[6] GoF: del inglés Gand of Four

como son la responsabilidad de las clases, la reutilización, la complejidad de implementación y mantenimiento.

Se realizó un estudio de varias métricas de diseño que evaluaran en su totalidad los parámetros antes mencionados. Se escogieron las métricas orientadas a clases Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC) (Kidd, 1994), que permiten evaluar estos parámetros y adicionalmente el bajo acoplamiento y la cantidad de pruebas de unidad que deben realizarse. Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Algunos especialistas plantean umbrales basándose en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño de la herramienta, se reflejan en la Tabla 6 Tabla 7.

Tamaño Operacional de Clase (TOC)

El tamaño operacional de una clase está dado por el número de métodos asignados a una clase. Para ello mide los siguientes atributos de calidad:

- Responsabilidad: responsabilidad asignada a una clase. Un aumento del TOC implica un aumento de esta responsabilidad, teniendo así demasiada responsabilidad, lo cual reduce la posibilidad de reutilización de la clase, lo que hace complicada la implementación y la prueba.
- Complejidad de implementación: grado de dificultad que tiene implementar un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de una determinada clase.
- Reutilización: grado de reutilización presente en una clase o estructura de clase. Un aumento del TOC implica una disminución del grado de reutilización de una determinada clase.

Tabla 6 Criterios de evaluación para la métrica TOC.

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

Relaciones entre Clases (RC)

Esta métrica está dada por el número de relaciones de uso de una clase. Para ello mide los siguientes atributos de calidad:

- Acoplamiento: grado de dependencia de una clase o estructura de clase, con otras. Un aumento del RC provoca aumento del acoplamiento de la clase.
- Complejidad de mantenimiento: grado de esfuerzo necesario para desarrollar un arreglo, rectificación de algún error o mejora de un diseño. Un aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
- Reutilización: grado de reutilización presente en una clase o estructura de clase. Un aumento del RC provoca disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: grado de esfuerzo necesario para realizar pruebas de unidad al sistema diseñado. Un aumento del RC provoca aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 7 Criterios de evaluación para la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>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

1.11 Evaluación de software

Para la evaluación del software se realizan pruebas de caja blanca, pruebas de caja negra y pruebas de aceptación.

Pruebas de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. (Pressman, 2010)

Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. Estas permiten al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (Pressman, 2010)

Pruebas de aceptación

Las pruebas de aceptación son definidas por el cliente para cada historia de usuario. Se utilizan para validar que cada requerimiento implementado funciona como se había especificado. (Malfará, 2006)

1.12 Conclusiones parciales

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- Se enunciaron los principales términos asociados al dominio de la investigación, lo que permitió adquirir un mayor conocimiento sobre el objeto de estudio de la investigación.
- El estudio de la computación con palabras, específicamente del modelo lingüístico 2-tuplas permitió un mejor entendimiento para su posterior implementación en el lenguaje plpgsql.
- La metodología de software que guia el proceso de desarrollo es XP, dada la interacción con el cliente, el poco tiempo de duración y que el equipo de desarrollo es pequeño.
- Realizada una revisión de los principales elementos teóricos para el desarrollo del trabajo, se decide desarrollar una aplicación web mediante el marco de trabajo Symfony, con el uso del lenguaje de programación PHP y como gestor de base de datos PostgreSQL.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

2.1 Introducción

En el presente capítulo se describe la propuesta de solución y los roles que interactúan con el sistema. Se presenta el modelo de dominio. Se presentan los requisitos funcionales identificados mediante técnicas de tormentas de ideas y entrevista, así como los prototipos empleados para su validación y los requisitos no funcionales del sistema. Resultado de la planificación, se presentan las historias de usuario y el plan de iteraciones para el desarrollo del sistema. Resultado del diseño se describe la arquitectura y el diseño del software mediante las tarjetas clase-responsabilidad-colaborador (CRC), el modelo de datos y los patrones de diseño aplicados, y se valida el diseño mediante el uso de métricas.

2.2 Descripción general de la propuesta de solución

Para dar cumplimiento a los objetivos previamente planteados, el sistema propuesto permitirá la gestión de fichas de proyectos y con estas poder realizar la evaluación técnica y comercial de estos proyectos. El proceso de evaluación recibirá como entrada un grupo de proyectos adicionados por el especialista, de estos proyectos se seleccionan los criterios técnicos y comerciales que se desean evaluar, los criterios están definidos por (Abreu, 2012). A cada uno de los criterios seleccionados se le da un valor de importancia. Luego cada uno de los expertos que participan en el proceso da una evaluación para cada uno de los criterios. Con estas evaluaciones se ejecuta el modelo lingüístico 2-tuplas, para obtener como resultado final un listado de los proyectos ordenados según la prioridad dada por el sistema.

2.3 Roles relacionados con el sistema

Con el objetivo de restringir el acceso a las opciones que presenta el sistema, se definen los siguientes roles:

Tabla 8. Roles relacionados con el sistema.

Actores	Descripción
Administrador	Posee acceso a todo el sistema. Puede efectuar cualquier funcionalidad en el sistema.
Especialista	Se encarga de gestionar en el sistema las fichas de proyectos y de realizar la evaluación.
Experto	Se encarga de evaluar los proyectos atendiendo a los criterios que sean seleccionados por el especialista.

2.4 Modelo conceptual

Un modelo conceptual es una representación de conceptos en un dominio del problema. En UML se ilustra como un grupo de diagramas de estructura estática donde no se define ninguna operación.

Ofrece como ventaja subrayar fuertemente una concentración en los conceptos del dominio, no en las entidades del software. (Larman, 1999)

La Figura 1 muestra los principales conceptos del dominio y las relaciones entre ellos.

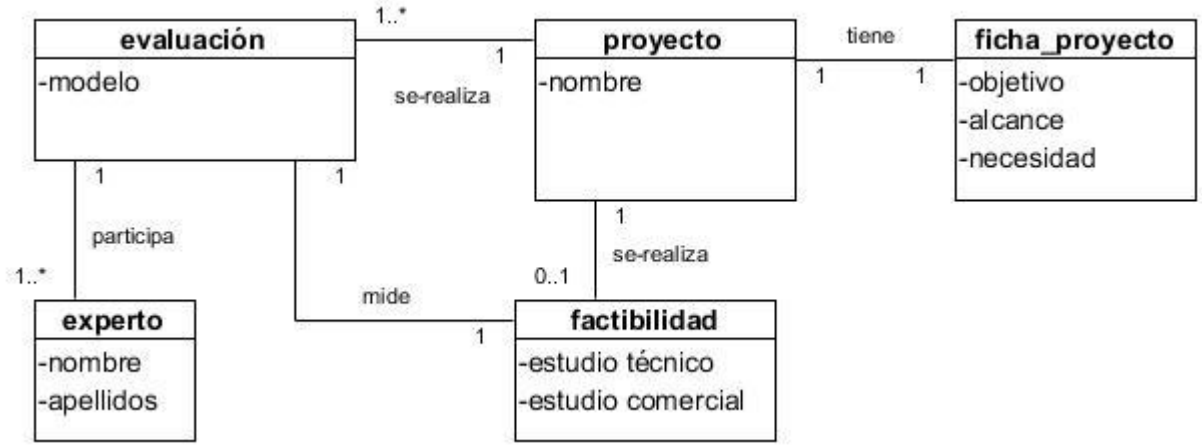


Figura 1. Modelo conceptual.

Descripción de los conceptos

Proyecto: Representa los proyectos que serán evaluados para realizar análisis de factibilidad técnica y comercial.

Ficha proyecto: Representa los datos de los proyectos que serían evaluados.

Evaluación: Representa la evaluación de los proyectos para la realización de los análisis de factibilidad, haciendo uso del modelo lingüístico 2-tuplas.

Factibilidad: Representa los estudios de factibilidad técnica y comercial.

Experto: Personal capacitado en conocimientos sobre los estudios de factibilidad técnica y comercial.

2.5 Fase de planificación

Para el desarrollo de la herramienta se dividirá el trabajo en las fases propuestas por la metodología XP.

En la fase de planificación se efectúa un diálogo entre las partes involucradas en el proyecto, incluyendo al cliente y a los programadores. Se comienza recopilando las historias de usuario, las cuales sustituyen a los casos de uso y se evalúa el tiempo de desarrollo de cada una por parte de los programadores. Mediante un plan de iteraciones se planifica el tiempo que dura el desarrollo del sistema.

2.5.1 Requisitos del software

Como parte del análisis previo al diseño de la solución se emplearon las siguientes técnicas de obtención o captura de requisitos:

Según (Escalona, y otros, 2002) la **entrevista** es una técnica muy aceptada dentro de la ingeniería de requisitos. Permiten tomar conocimiento del problema y comprender los objetivos de la solución buscada, para tener una amplia visión de las necesidades del usuario. Se realizaron entrevistas con el cliente para obtener información acerca de las necesidades y perspectivas de la solución. Se determinó el desarrollo de una aplicación que sirva de apoyo a la toma de decisiones de los principales especialistas de los centros productivos, y facilitar la realización de estudios de factibilidad técnica y comercial (Consultar Anexo 1).

La **tormenta de ideas** una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la acumulación de ideas y/o información sin ser evaluadas (Escalona, y otros, 2002). En varias reuniones se desarrollaron tormentas de ideas con la participación del equipo de desarrollo y el cliente. Como resultado se logró generar opiniones sobre los requisitos a satisfacer por el software.

2.5.1.1 Requisitos funcionales

Los requisitos funcionales describen el funcionamiento del sistema, la forma en que debe reaccionar ante ciertas entradas y cómo se debe comportar en situaciones específicas. Detallan la función del producto de software, entrada, salidas, excepciones y usuarios (Sommerville, 2007).

Luego de aplicadas las técnicas de entrevista y tormenta de ideas se identificaron un total de 36 requisitos funcionales (RF). La prioridad de cada requisito fue definida por el cliente en función de la importancia para el negocio.

Tabla 9. Requisitos funcionales.

Código	Descripción	Prioridad
RF-1	Adicionar usuario	Media
RF-2	Modificar usuario	Media
RF-3	Eliminar usuario	Media
RF-4	Listar usuarios	Baja
RF-5	Buscar usuario	Baja
RF-6	Adicionar rol	Media
RF-7	Modificar rol	Media

RF-8	Eliminar rol	Media
RF-9	Listar roles	Baja
RF-10	Buscar rol	Baja
RF-11	Adicionar ficha de proyecto	Alta
RF-12	Modificar ficha de proyecto	Media
RF-13	Eliminar ficha de proyecto	Media
RF-14	Mostrar ficha de proyecto	Media
RF-15	Mostrar criterios técnicos de una ficha de proyecto	Media
RF-16	Mostrar criterios comerciales de una ficha de proyecto	Media
RF-17	Listar fichas de proyectos	Baja
RF-18	Buscar ficha de proyecto	Baja
RF-19	Adicionar evaluación de proyectos	Alta
RF-20	Eliminar evaluación de proyectos	Media
RF-21	Mostrar resultado de la evaluación de proyectos	Alta
RF-22	Listar evaluaciones de proyectos	Baja
RF-23	Buscar evaluación de proyectos	Baja
RF-24	Agregar proyecto a una evaluación	Alta
RF-25	Eliminar proyecto de una evaluación	Media
RF-26	Agregar experto a una evaluación	Alta
RF-27	Eliminar experto de una evaluación	Media
RF-28	Agregar criterio comercial a evaluar	Alta
RF-29	Agregar criterio técnico a evaluar	Alta
RF-30	Eliminar criterio comercial de una evaluación	Media
RF-31	Eliminar criterio técnico de una evaluación	Media
RF-32	Añadir valor de importancia de los criterios	Alta
RF-33	Evaluar criterio por parte del experto	Alta
RF-34	Aplicar modelo 2-tuplas para la evaluación de criterios técnicos y comerciales	Alta

RF-35	Generar listado ordenado de proyectos evaluados	Alta
RF-36	Exportar resultado de la evaluación	Media

2.5.1.2 Requisitos no funcionales

Los requisitos no funcionales definen propiedades del sistema, tales como: tiempo de respuesta, necesidades de almacenamiento, fiabilidad, usabilidad y seguridad (Sommerville, 2007). A continuación se listan los requisitos no funcionales de la aplicación a desarrollar.

Tabla 10. Requisitos no funcionales.

Código	Descripción
Usabilidad	
RNF 1	La herramienta informática a desarrollar será una aplicación web.
RNF 2	La aplicación debe presentar una vista sencilla, descriptiva y fácil de usar, con el objetivo de proporcionar a los usuarios un mejor entendimiento con la aplicación.
Portabilidad	
RNF 3	La aplicación debe poder instalarse en los sistemas operativos Windows y Linux/Unix
Fiabilidad	
RNF 4	La aplicación restringirá el acceso por roles a los usuarios autorizados.
Disponibilidad	
RNF 5	La aplicación deberá estar disponible siempre, asegurando el acceso desde cualquier lugar de red a los usuarios autorizados.
Eficiencia	
RNF 6	El tiempo de respuesta en el procesamiento de los datos y solicitud de estos no debe sobrepasar los 5 segundos.
Interfaz	
RNF 7	La aplicación debe ofrecer una interfaz amigable y un diseño sencillo que le permita al usuario interactuar fácilmente.
RNF 8	La aplicación, al ocurrir un error con los datos de entrada muestra mensajes al usuario informando este.
Software	

RNF 9	El servidor de aplicación debe tener instalado el servidor web Apache y el marco de trabajo Symfony.
RNF 10	El servidor de base de datos debe tener como Gestor de Base de Datos PostgreSQL en su versión 9.2 o superior.
RNF 11	Las PC clientes deben tener instalado un navegador web (Mozilla Firefox, Google Chrome); se recomienda para estas Mozilla Firefox en su versión 24.0 o superior.
Hardware	
RNF 12	La PC servidor debe tener 2 GB de memoria RAM o superior, un disco duro de 320 GB de almacenamiento o superior, un procesador Pentium IV 2.4 GHz o superior.
RNF 13	Las PC clientes deben tener 512 MB, como mínimo de memoria RAM, un procesador Pentium IV 1.7 GHz o superior.
Seguridad	
RNF 14	Existirán diferentes tipos de usuarios según las acciones que puedan realizar. Los permisos serán limitados, basados en los roles definidos y el usuario no podrá gestionarlos.
RNF 15	La aplicación debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
RNF 16	La aplicación debe permitir la comprobación de credenciales en la autenticación del usuario en el servidor de aplicaciones y no en el servidor de base de datos, para evitar inyecciones SQL que falseen la autenticación y provoquen la suplantación de identidad.
Implementación	
RNF 17	Los operadores de agregación del modelo 2-tuplas deberán ser implementados en el lenguaje plpgsql de PostgreSQL.
RNF 18	La herramienta informática será desarrollada con HTML5 y haciendo uso de CSS y Javascript.

2.5.2 Historias de usuario

Las historias de usuario sustituyen a los documentos de especificación funcional y a los casos de uso. Estas son escritas por el cliente en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar (Joskowicz, 2008). A continuación se muestran las historias de usuario correspondientes

a los requisitos funcionales *Adicionar evaluación de proyectos* y *Mostrar resultado de la evaluación de proyectos*, de un total de 36, el resto pueden ser consultadas en el Anexo 3.

Tabla 11. Historia de usuario: *Adicionar evaluación de proyectos*

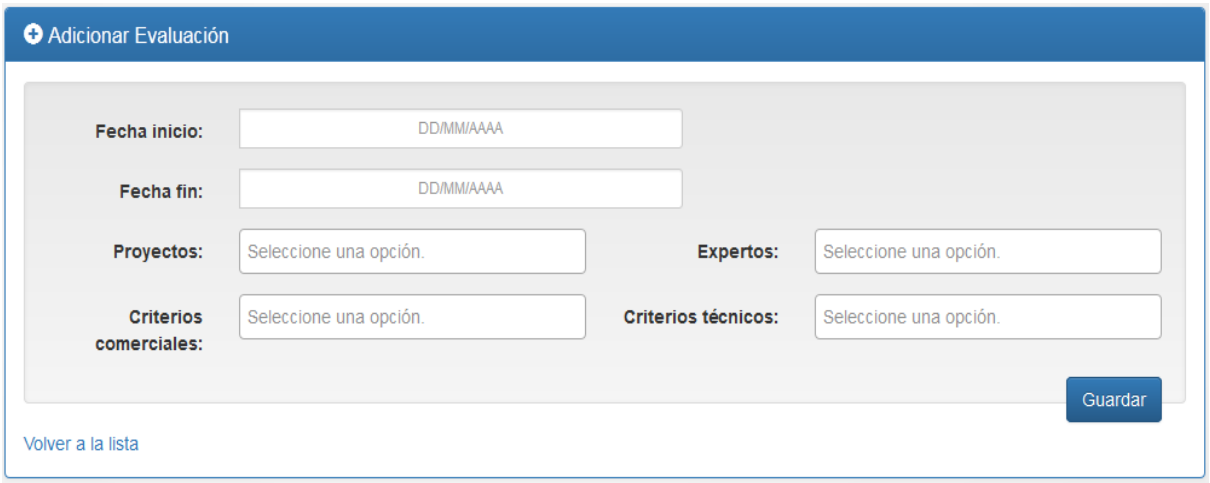
HISTORIA DE USUARIO	
Número: HU-19	**Nombre Historia de Usuario:** Adicionar evaluación de proyectos
Usuario: Alejandro Álvarez	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos Estimados:** 3 días
Riesgo en desarrollo:	**Puntos Reales:** 3 días
Descripción: Permite al especialista adicionar una evaluación de proyectos adicionados en el sistema. Se debe mostrar una interfaz para insertar los datos de la evaluación.	
Observaciones: Esta funcionalidad solo puede ser realizada por un usuario con rol de especialista.	
Interfaz de usuario: 	

Tabla 12. Mostrar resultado de la evaluación de proyectos

HISTORIA DE USUARIO	
Número: HU-21	**Nombre Historia de Usuario:** Mostrar resultado de la evaluación de proyectos
Usuario: Alejandro Álvarez	**Iteración Asignada:** 1

Prioridad en negocio: Alta	**Puntos Estimados:** 2 días
Riesgo en desarrollo:	**Puntos Reales:** 2 días
Descripción: Permite al especialista ver el resultado de la evaluación de los proyectos.	
Observaciones: Esta funcionalidad solo puede ser realizada por un usuario con rol de especialista.	
Interfaz de usuario:	

2.5.3 Plan de iteraciones

Las historias de usuarios seleccionadas para cada plan de entrega son desarrolladas y probadas en un ciclo de iteración de acuerdo al orden preestablecido, se estima el tiempo de duración de cada una de ellas. Cada historia de usuario se traduce en tareas específicas de programación (Joskowicz, 2008). El sistema será desarrollado en tres iteraciones, en una primera iteración serán implementadas las historias de usuarios de prioridad alta para el negocio, en una segunda iteración las de prioridad media y en una tercera iteración las de prioridad baja.

Tabla 13. Plan de iteraciones

Iteración	Historia de usuario	Prioridad	Duración de HU (días)	Duración total (semanas)
1	Adicionar ficha de proyecto	Alta	3	4
	Adicionar evaluación de proyectos	Alta	3	
	Mostrar resultado de la evaluación de proyectos	Alta	2	
	Agregar proyecto a una evaluación	Alta	1	
	Agregar experto a una evaluación	Alta	1	
	Agregar criterio comercial a evaluar	Alta	1	
	Agregar criterio técnico a evaluar	Alta	1	
	Añadir valor de importancia de los criterios	Alta	1	
	Evaluar criterio por parte del experto	Alta	2	
	Aplicar modelo 2-tuplas para la evaluación de criterios técnicos y comerciales	Alta	3	
	Generar listado ordenado de proyectos evaluados	Alta	1	
2	Adicionar usuario	Media	2	5
	Modificar usuario	Media	2	
	Eliminar usuario	Media	1	
	Adicionar rol	Media	1	
	Modificar rol	Media	1	
	Eliminar rol	Media	1	
	Modificar ficha de proyecto	Media	2	
	Eliminar ficha de proyecto	Media	1	
	Mostrar ficha de proyecto	Media	1	
	Mostrar criterios técnicos de una ficha de proyecto	Media	2	
	Mostrar criterios comerciales de una ficha de proyecto	Media	2	
	Eliminar evaluación de proyectos	Media	1	
	Eliminar proyecto de una evaluación	Media	1	
	Eliminar experto de una evaluación	Media	1	

	Eliminar criterio comercial de una evaluación	Media	1	
	Eliminar criterio técnico de una evaluación	Media	1	
	Exportar resultado de la evaluación	Media	3	
3	Listar usuarios	Baja	1	2
	Buscar usuario	Baja	1	
	Listar roles	Baja	1	
	Buscar rol	Baja	1	
	Listar fichas de proyectos	Baja	1	
	Buscar ficha de proyecto	Baja	1	
	Listar evaluaciones de proyectos	Baja	1	
	Buscar evaluación de proyectos	Baja	1	
Total				11

2.6 Fase de diseño

En la fase de diseño se confeccionan las tarjetas Clase Responsabilidad Colaborador (CRC) para crear diseños de clases orientados a responsabilidades. Se selecciona la arquitectura adecuada para el desarrollo de la herramienta.

2.6.1 Tarjetas clase-responsabilidad-colaborador (CRC)

Las tarjetas CRC se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. Tienen la finalidad de obtener un diseño simple y fácil de comprender por parte de los programadores. A continuación se muestra la tarjeta CRC correspondientes a la clase *ProcesoEvaluacion,* el resto de las tarjetas CRC pueden ser consultadas en el Anexo 3

Tabla 14. Tarjeta CRC: *ProcesoEvaluacion*

TARJETA CRC	
Clase: ProcesoEvaluacion	
Descripción: Almacena las evaluaciones de los expertos	
Atributos	
Nombre	**Descripción**
id	Campo identificador
idEvaluacion	Identificador de la evaluación asociada
idExperto	Identificador del experto asociado

criterio	
evaluacion	
importancia	
Responsabilidades:	**Colaboraciones:**
Contener la información de las evaluaciones de los expertos	

2.6.2 Modelo de datos

El modelado y diseño de la base de datos tiene como objetivo generar un conjunto de entidades relacionadas entre sí. Este modelo es un artefacto significativo en el diseño de la propuesta de solución, pues se definen los conceptos que se manejan en el sistema y se describe la estructura de la base de datos diseñada. En la Figura 2 se muestra el modelo de datos correspondiente a la solución, donde se representan los datos, sus atributos y tipos, así como las relaciones, este está compuesto por 11 tablas, de ellas 2 nomencladoras.

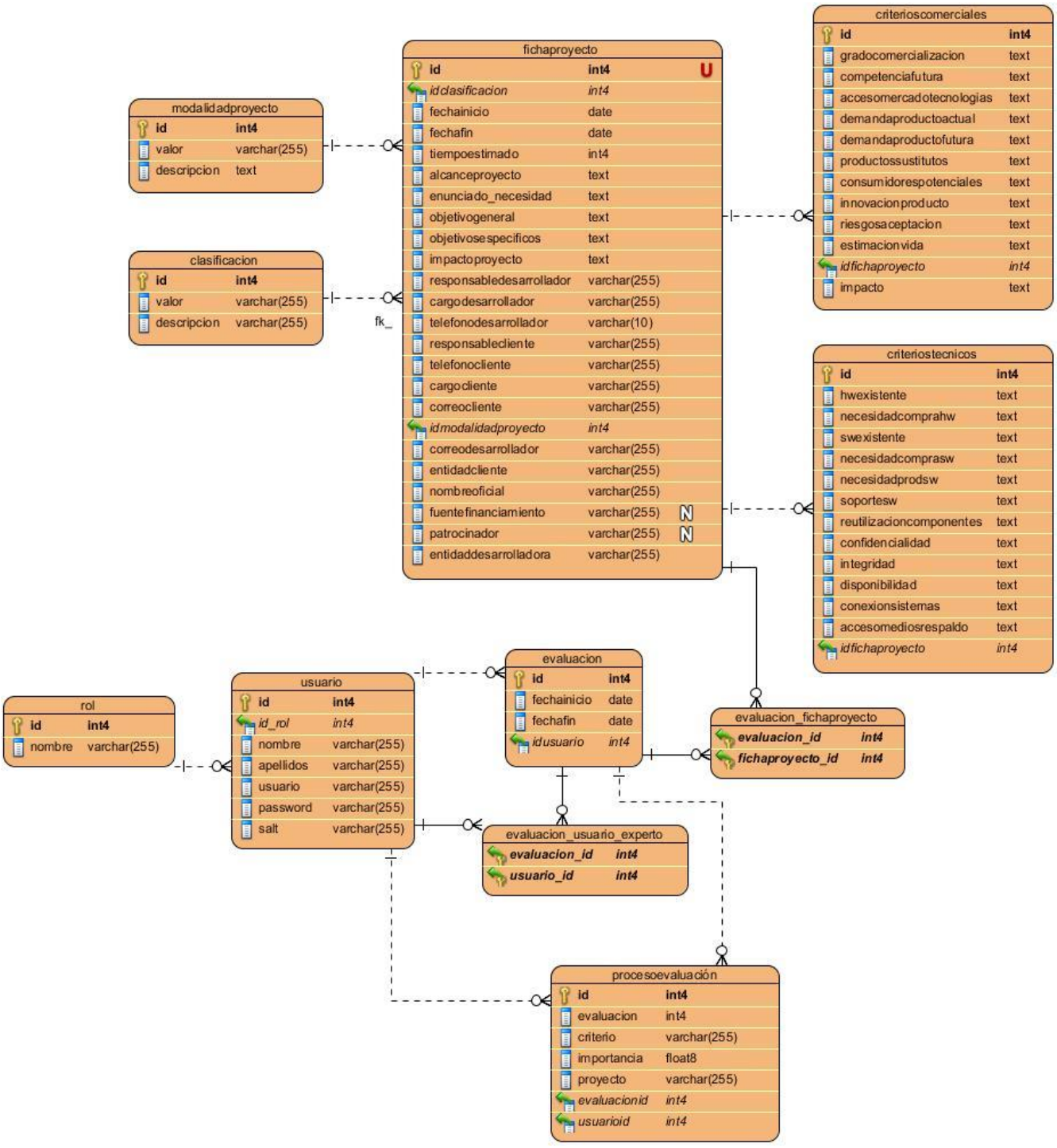


Figura 2. Modelo de datos

2.6.3 Arquitectura del sistema

La arquitectura de la herramienta está basada en el patrón arquitectónico Modelo Vista Controlador, en el cual está basado el marco de trabajo seleccionado Symfony2. Este garantiza una organización en el

código de la aplicación mediante la separación de los datos y la lógica del negocio de la interfaz de usuario en tres componentes: el modelo, la vista y el controlador.

- Modelo**:** representa y gestiona los datos manejados por la aplicación. Integra las clases que contienen las consultas a la base de datos, así como la definición de las tablas, sus relaciones y atributos.
- Vista**:** presenta la información del modelo al usuario. Determina la interfaz que se muestra finalmente al cliente para su intercambio con la aplicación.
- Controlador**:** recibe las órdenes del usuario para servir a las demandas del modelo o las vistas. Gestiona las peticiones del usuario y se encarga de darle respuestas.

Para mostrar la estructura y composición de la herramienta haciendo uso de este patrón arquitectónico se realiza un diagrama de componentes, donde se muestra la relación entre estos.

Un diagrama de componentes muestra las dependencias entre tipos de componentes. Cada componente se conecta a otros componentes cuyos servicios utiliza (Rumbaugh, y otros, 2000). Permite describir los elementos físicos que integran el sistema y las relaciones que existen entre ellos.

La Figura 3 muestra la arquitectura de la herramienta a través del diagrama de componentes. Este está compuesto por un componente principal llamado HerramientaBundle, el cual está asociado a otros componentes. Se puede apreciar el componente Modelo, donde se encuentran las clases Repositorios, que hacen uso del componente ORM Doctrine, y las clases Entidades. La Vista hace uso de los componentes Twig y TwitterBootstrap. Esta a su vez es usada por el Controlador, el cual utiliza los componentes Enrutamiento, Entidades y SpraedPdfGenerator.

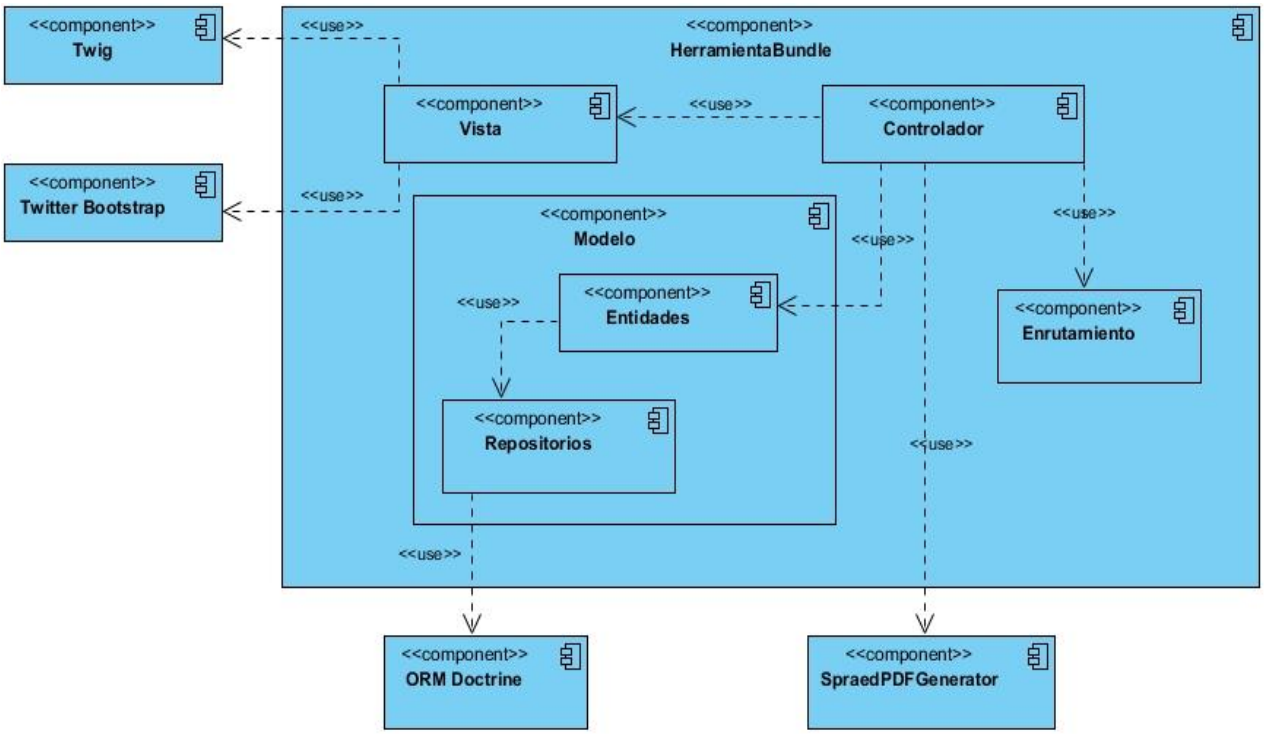


Figura 3. Arquitectura de la herramienta mediante diagrama de componentes

2.6.4 Patrones de diseño

Patrones generales de software para asignación de responsabilidades (GRASP)

A continuación se describen los patrones GRASP empleados en el desarrollo del sistema.

Experto: el uso de este patrón resulta de utilidad en las clases del modelo, las cuales contienen toda la información relacionada con los objetos persistentes que representan. La Figura 4 muestra el uso de este patrón en la entidad Usuario. ***Figura 4***

```
private $nombre;

public function setNombre($nombre) {
    $this->nombre = $nombre;
}

public function getNombre() {
    return $this->nombre;
}
```

Figura 4. Patrón experto en la entidad Usuario.

Creador: este patrón se pone de manifiesto en las clases controladoras del sistema. Brinda soporte a un bajo acoplamiento y mejora la reutilización. La Figura 5 muestra el uso de este patrón en la clase controladora EvaluacionController.php

```
public function createAction(Request $request) {
    $entity = new Evaluacion(); //1
    $form = $this->createCreateForm($entity); //1
    $form->handleRequest($request); //1
```

Figura 5. Patrón creador en la clase EvaluacionController.php.

Bajo acoplamiento: en el diseño del sistema cada clase depende lo menos posible de otra, de tal manera que una de ellas solo recurre a otra en caso de que exista referencia dentro de sus atributos, lo que permite que el sistema sea mucho más robusto y de fácil mantenimiento. Este se evidencia en el marco de trabajo, ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.

Alta cohesión: en el diseño del sistema las clases tienen responsabilidades estrechamente relacionadas y no realizan un trabajo excesivo, lo que permite simplificar el mantenimiento y aumentar la capacidad de reutilización de estas. Symfony permite asignar responsabilidades con una alta cohesión. Este se evidencia en las clases repositorio, que tienen como única responsabilidad realizar operaciones de acceso a datos solo con la entidad que representa.

Controlador: este patrón se evidencia en las clases controladoras, responsables de implementar las funcionalidades pertenecientes a una interfaz determinada. La Figura 6 muestra el uso de este patrón en la clase controladora FichaProyectoController.php.

```
class FichaProyectoController extends Controller {

    public function indexAction() {...}

    public function createAction(Request $request) {...}

    private function createCreateForm(FichaProyecto $entity) {...}

    public function newAction() {...}

    public function showAction($id) {...}

    public function editAction($id) {...}

    private function eliminarCriterios($id) {...}

    public function verCriteriosComercialesAction($id) {...}

    public function verCriteriosTecnicosAction($id) {...}

}
```

Figura 6. Patrón controlador en la clase FichaProyectoController.php

Patrones GoF aplicados

A continuación se describen los patrones GoF utilizados en el desarrollo del sistema.

Creacionales:

Método de Fábrica (del inglés Factory Method): define una interfaz para la creación de un objeto, lo que permite a las subclases decidir de qué clase instanciarlo. Permite que una clase difiera la instanciación en favor de sus subclases. En Symfony 2 a la hora de crear un Formulario se llama al método createForm(), el cual decide el tipo de formulario a crear, así como los tipos de datos correspondientes a los campos del formularios (text,textarea,select, etc).

```
53      /**
54       * Creates a form to create a FichaProyecto entity.
55       *
56       * @param FichaProyecto $entity The entity
57       *
58       * @return \Symfony\Component\Form\Form The form
59       */
60      private function createCreateForm(FichaProyecto $entity) {
61          $form = $this->createForm(new FichaProyectoType(), $entity, array(
62              'action' => $this->generateUrl('fichaproyecto_create'),
63              'method' => 'POST',
64          ));
65
66
67          return $form;
68      }
```

Figura 7. Uso del patrón Método de Fábrica.

Estructurales

Adaptador (del inglés Adapter): permite a una interfaz de una clase existente ser usada por otra interfaz; y a las clases, trabajar con otras sin cambiar su código. La clase Usuario implementa la interfaz UserInterface, el cual es un componente del núcleo de Seguridad de Symfony2. Esta le permite a la clase Usuario implementar los métodos eraseCredentials(), getUsername() y getRoles(), necesarias para el correcto funcionamiento de la autenticación en Symfony2.

```
179
180         /**
181          * Set permiso
182          *
183          * @param string $permiso
184          * @return Usuario
185          */
186         public function eraseCredentials() {
187             return false;
188         }
189
190         public function getUsername() {
191             return $this->usuario;
192         }
193
194         public function getRoles() {
195             $a=$this->rol->getRole();
196             return array($a);
197         }
198
199     }
200
```

Figura 8. Uso del patrón Adaptador en la clase Usuario.php.

Decorador (del inglés Decorator): aplicado a la generación de vistas, la solución que ofrece este patrón es la de añadir funcionalidad adicional a las plantillas. Ejemplo, añadir el menú y el pie de página a las plantillas que lo requieran; se trata de decorar las plantillas con elementos adicionales reutilizables. El sistema de plantillas Twig está provisto de un mecanismo de herencia gracias al cual la decoración de plantillas resulta de una flexibilidad y versatilidad total.

```
5     <div class="col-xs-10">
6         <div class="panel panel-primary">
7             <div class="panel-heading">
8                 <h3 class="panel-title">Criterios Comerciales del proyecto</h3>
9             </div>
10            <div class="panel-body">
11                <form id="criteriosComerciales" class="form-horizontal" role="form" action="{{ path('criterioscomerciales_create') }}" {{
                  form_enctype(form) }} method="POST">
12                    {% if form_errors(form)!=null %}
13                        <div class="alert alert-danger">{{ form_errors(form) }}</div>
14                    {% endif %}
15                    <div class="tab-content">
16                        <div class="tab-pane fade in active" id="primero" >
17                            {%include 'TesisBundle:CriteriosComerciales:panel_1.html.twig'%}
18                        </div>
19                        <div class="tab-pane fade" id="segundo">
20                            {%include 'TesisBundle:CriteriosComerciales:panel_2.html.twig'%}
21                        </div>
22                    </div>
23                </form>
24
25            </div>
26        </div>
27    </div>
28 {% endblock %}
29
```

Figura 9. Uso del patrón Decorador.

Comportamiento

Mediador (del inglés Mediator): Las clases controladoras son mediadoras entre las capas de modelo y vista.

2.6.5 Validación del diseño

Para la validación del diseño se aplicaron las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC).

Tamaño Operacional de Clases (TOC)

Los resultados obtenidos luego de aplicar la métrica se muestran en la Figura 10.

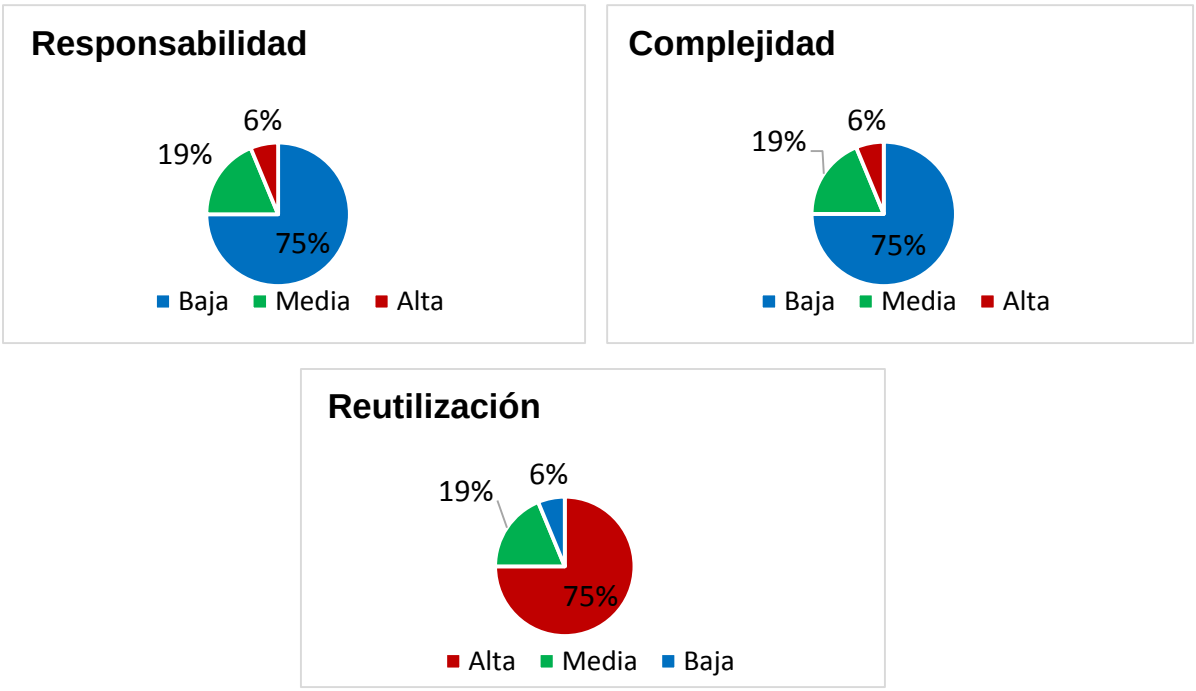


Figura 10. Representación de los atributos de calidad de la métrica TOC

Relaciones entre Clases (RC)

Los resultados obtenidos luego de aplicar la métrica se muestran en la Figura 11.

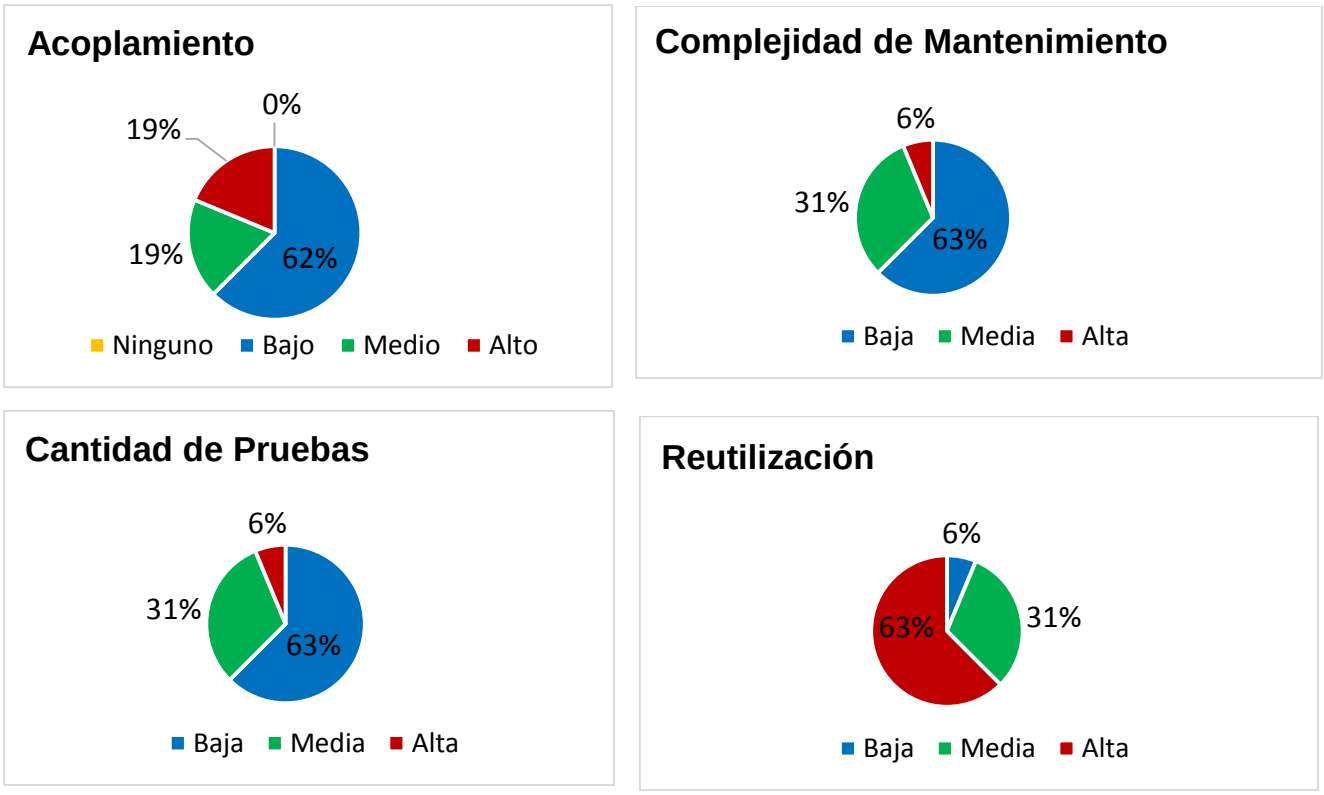


Figura 11. Representación de los atributos de calidad de la métrica RC.

Luego de aplicadas las métricas se verificó que el diseño propuesto provee un nivel bajo de responsabilidad. Esto trae consigo un porciento bajo de complejidad de implementación, lo cual significa un menor grado de dificultad en el desarrollo de la herramienta. El 63% de las clases presentan un nivel bajo de mantenimiento, lo que reduce el grado de esfuerzo necesario para desarrollar un arreglo o rectificar algún error. Teniendo en cuenta que el 63% de las clases tienen una baja necesidad de realizar pruebas unitarias al sistema diseñado, se reduce el esfuerzo necesario para la realización de estas. En cuanto a la reutilización se alcanzaron los resultados deseados, pues se obtuvieron valores altos de reutilización en ambas métricas, 75% en TOC y 63% en RC, ya que se busca el poder reutilizar las clases en caso de ser necesario. Se tiene en cuenta además el acoplamiento, que atendiendo a los resultados obtenidos el 62% de las clases tienen bajo acoplamiento, lo que evidencia el uso del patrón bajo acoplamiento. Con los resultados anteriores queda validado el diseño de la herramienta a desarrollar, atendiendo a los parámetros definidos.

2.7 Conclusiones parciales

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- Se confeccionó el modelo conceptual para una representación visual del dominio del problema.

- La confección de los artefactos historias de usuario, plan de iteraciones, tarjetas CRC, y el modelo de datos permitieron organizar el proceso de desarrollo del sistema.
- Se fundamentó la elección del patrón Modelo Vista Controlador para la arquitectura del sistema mediante un diagrama de componentes.
- El uso de los patrones de diseño y de las métricas TOC y RC para la validación de este contribuyeron a la obtención de un diseño con calidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el presente capítulo se muestran las tareas de ingeniería, artefactos generados en la fase de desarrollo de la metodología de software utilizada. Se abordan los estándares de codificación empleados durante la implementación del sistema para garantizar un buen entendimiento y legibilidad del código. Se describen las pruebas efectuadas al software que tienen como objetivo: detectar y corregir el máximo de errores en el sistema, antes de su entrega al cliente.

3.2 Fase de desarrollo

En la fase de desarrollo se planifican y ejecutan las tareas de ingeniería. Durante esta se codifica todo el sistema diseñado y se obtiene como resultado la herramienta propuesta.

3.2.1 Tareas de ingeniería por iteraciones

Las tareas de ingeniería son actividades que se derivan de las HU para simplificar su implementación. A continuación se muestran las tareas a desarrollar para la implementación, por cada HU.

Tabla 15. Tareas de ingeniería por historia de usuario.

Iteración	Historia de usuario	Tareas de ingeniería
1	Adicionar ficha de proyecto	Implementar una funcionalidad que permita a un especialista adicionar una ficha de proyecto en el sistema.
	Adicionar evaluación de proyectos	Implementar una funcionalidad que permita a un especialista adicionar una evaluación de proyectos en el sistema.
	Mostrar resultado de la evaluación de proyectos	Implementar una funcionalidad que permita mostrar el resultado de la evaluación realizada por un especialista, una vez evaluados los criterios por parte del experto.
	Agregar proyecto a una evaluación	Implementar una funcionalidad que permita a un especialista agregar un proyecto a una evaluación.
	Agregar experto a una evaluación	Implementar una funcionalidad que permita a un especialista agregar un experto a una evaluación.
	Agregar criterio comercial a evaluar	Implementar una funcionalidad que permita a un especialista agregar un criterio comercial a una evaluación.

	Agregar criterio técnico a evaluar	Implementar una funcionalidad que permita a un especialista agregar un criterio técnico a una evaluación.
	Añadir valor de importancia de los criterios	Implementar una funcionalidad que permita a un especialista añadir los valores de importancia de los criterios seleccionados.
	Evaluar criterio por parte del experto	Implementar una funcionalidad que permita a un experto emitir una evaluación para cada criterio seleccionado.
	Aplicar modelo 2-tuplas para la evaluación de criterios técnicos y comerciales	Implementar en plpgsql los operadores de agregación del modelo lingüístico 2-tuplas.
	Generar listado ordenado de proyectos evaluados	Implementar una funcionalidad para generar el listado de los proyectos evaluados.
2	Adicionar usuario	Implementar una funcionalidad que permita a un administrador adicionar un usuario en el sistema.
	Modificar usuario	Implementar una funcionalidad que permita a un administrador modificar un usuario adicionado en el sistema
	Eliminar usuario	Implementar una funcionalidad que permita a un administrador eliminar un usuario del sistema.
	Adicionar rol	Implementar una funcionalidad que permita a un administrador adicionar un rol en el sistema.
	Modificar rol	Implementar una funcionalidad que permita a un administrador modificar un rol adicionado en el sistema.
	Eliminar rol	Implementar una funcionalidad que permita a un administrador eliminar un rol del sistema.
	Modificar ficha de proyecto	Implementar una funcionalidad que permita a un especialista modificar una ficha de proyecto del sistema.

	Eliminar ficha de proyecto	Implementar una funcionalidad que permita a un especialista eliminar una ficha de proyecto del sistema.
	Mostrar ficha de proyecto	Implementar una funcionalidad que permita mostrar los datos de una ficha de proyecto adicionada en el sistema.
	Mostrar criterios técnicos de una ficha de proyecto	Implementar una funcionalidad que permita mostrar los criterios técnicos de una ficha de proyecto adicionada en el sistema.
	Mostrar criterios comerciales de una ficha de proyecto	Implementar una funcionalidad que permita mostrar los criterios comerciales de una ficha de proyecto adicionada en el sistema.
	Eliminar evaluación de proyectos	Implementar una funcionalidad que permita eliminar una evaluación del sistema.
	Eliminar proyecto de una evaluación	Implementar una funcionalidad que permita eliminar un proyecto de evaluación del sistema.
	Eliminar experto de una evaluación	Implementar una funcionalidad que permita eliminar un experto de una evaluación del sistema.
	Eliminar criterio comercial de una evaluación	Implementar una funcionalidad que permita eliminar un criterio comercial de una evaluación del sistema.
	Eliminar criterio técnico de una evaluación	Implementar una funcionalidad que permita eliminar un criterio comercial de una evaluación del sistema.
	Exportar resultado de la evaluación	Implementar una funcionalidad que permita exportar el resultado de la evaluación en formato .pdf.
	Listar usuarios	Implementar una funcionalidad que permita listar los usuarios adicionados en el sistema.
	Buscar usuario	Implementar una funcionalidad que permita buscar un usuario adicionado en el sistema.
	Listar roles	Implementar una funcionalidad que permita listar los roles adicionados en el sistema.
	Buscar rol	Implementar una funcionalidad que permita buscar un rol adicionado en el sistema.

3	Listar fichas de proyectos	Implementar una funcionalidad que permita listar las fichas de proyectos adicionadas en el sistema.
	Buscar ficha de proyecto	Implementar una funcionalidad que permita buscar una ficha de proyecto adicionada en el sistema.
	Listar evaluaciones de proyectos	Implementar una funcionalidad que permita listar las evaluaciones adicionadas en el sistema.
	Buscar evaluación de proyectos	Implementar una funcionalidad que permita buscar una evaluación adicionada en el sistema.

3.2.1.1 Tareas detalladas

A continuación se muestran las tareas de ingeniería detalladas correspondientes a las historias de usuario *Adicionar evaluación de proyectos* y *Mostrar resultado de la evaluación de proyectos*, el resto de las tareas detalladas se encuentran en el Anexo 5".

Tabla 16. Tarea de ingeniería detallada 3.

TAREA DE INGENIERÍA	
Número tarea: 2	**Historia de usuario:** Adicionar evaluación de proyectos
Nombre Tarea: Implementar funcionalidad adicionar evaluación de proyectos	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 3
Fecha inicio: 6/03/2015	**Fecha fin:** 11/03/2015
Programador responsable: Alejandro Álvarez	
Descripción: Implementar una funcionalidad que permita a un especialista adicionar una ficha de proyecto en el sistema.	

TAREA DE INGENIERÍA	
Número tarea: 3	**Historia de usuario:** Mostrar resultado de la evaluación de proyectos
Nombre tarea: Implementar funcionalidad mostrar resultado de la evaluación de proyectos	
Tipo de Tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 12/03/2015	**Fecha fin:** 16/03/2015

Programador responsable: Alejandro Alvarez
Descripción: Implementar una funcionalidad que permita mostrar el resultado de la evaluación realizada por un especialista, una vez evaluados los criterios por parte del experto.

3.2.2 Estándares de codificación

En función de lograr estandarización en el código del sistema, se definieron un conjunto de pautas a seguir durante la implementación.

- **Definición de clases**

  Las clases se encuentran en archivos independientes que solo contendrán el código de esta. Se utiliza el estilo de codificación "*UpperCamelCase"*, el cual establece que los nombres inician con letra mayúscula y si poseen más de una palabra, la primera letra de estas deberá ser mayúsculas también.

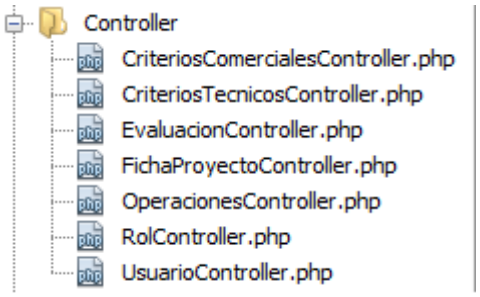


Figura 12. Definición de clases.

- **Definición de variables**

  Los nombres de las variables deben ser descriptivos y concisos. No se usan abreviaciones. Se utiliza el estilo de codificación "lowerCamelCase", el cual establece que los nombres inician con letra minúscula y cada nueva palabra debe iniciar con mayúscula.

```
33          * @var integer
34          *
35          * @ORM\Column(name="idClasificacion", type="integer")
36          */
37         private $idClasificacion;
38
39         /**
40          * @var integer
41          *
42          * @ORM\Column(name="prioridad", type="integer")
43          */
44         private $prioridad;
```

Figura 13. Definición de variables.

3.3 Fase de pruebas

El proceso de prueba se realiza de forma continua para asegurar durante todo el proceso de desarrollo, el éxito del producto. Esto permite detectar los errores en un plazo de tiempo corto.

3.3.1 Pruebas de caja blanca

Las pruebas de caja blanca son las encargadas de verificar el código y son diseñadas por los programadores. Para llevar a cabo esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba, que pongan a prueba los algoritmos implementados. Las pruebas de caja blanca se realizan a los principales algoritmos y procedimientos (Pressman, 2010).

Una técnica de las pruebas unitarias es la prueba del camino básico. Esta técnica permite al diseñador de casos de prueba obtener la complejidad lógica de un procedimiento o algoritmo y usar esta como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2010).

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes[7] del conjunto básico[8] de un programa y proporciona un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2010).
Se selecciona para la realización de las pruebas unitarias o pruebas de caja blanca el método o técnica prueba del camino básico, junto a la métrica complejidad ciclomática. Para llevar a cabo esta técnica se deben realizar los siguientes pasos:

- ✓ Representar el programa en un grafo de flujo
- ✓ Calcular la complejidad ciclomática
- ✓ Determinar el conjunto básico de caminos independientes
- ✓ Derivar los casos de prueba que fuerzan al ejecución de cada camino

A continuación se detallan cada uno de estos pasos.

[7] **Camino independiente:** cualquier camino del programa que introduce un nuevo conjunto de sentencias de proceso o una nueva condición.

[8] **Conjunto básico:** conjunto de caminos independientes.

Se selecciona como ejemplo el método createAction() de la clase EvaluacionController, el cual crea una evaluación a partir de los datos introducidos por el usuario.

```
public function createAction(Request $request) {
    $entity = new Evaluacion(); //1
    $form = $this->createCreateForm($entity); //1
    $form->handleRequest($request); //1

    if ($form->isValid()) { //2
        $em = $this->getDoctrine()->getManager(); //3
        $entity->setUsuario($this->getUser()); //3
        $criteriosTecnicos = $request->request->get('criteriosTecnicos'); //3
        $criteriosComerciales = $request->request->get('criteriosComerciales'); //3
        $formulario = $request->request->get('tesis_tesisbundle_evaluacion'); //3
        $exp = $formulario['usuariosExpertos']; //3
        $proy = $formulario['fichas']; //3
        $em = $this->getDoctrine()->getManager(); //3
        $proyectos = array(); //3
        $expertos = array(); //3
        foreach ($exp as $id) { //4
            $expertos[] = $em->getRepository('TesisBundle:Usuario')->find($id); //5
        }
        foreach ($proy as $id) { //6
            $proyectos[] = $em->getRepository('TesisBundle:FichaProyecto')->find($id); //7
        }
        $this->get('session')->set('ct', $criteriosTecnicos); //8
        $this->get('session')->set('cc', $criteriosComerciales); //8
        $this->get('session')->set('exp', $expertos); //8
        $this->get('session')->set('proy', $proyectos); //8
        $this->get('session')->set('exp_id', $exp); //8
        $this->get('session')->set('proy_id', $proy); //8
        $em->persist($entity); //8
        $em->flush(); //8
        $this->get('session')->set('eval_id', $entity->getId()); //8
        return $this->redirect($this->generateUrl('importancia_criterios', array('id' => $entity->getId()))); //8
    }
    else{
        return $this->render('TesisBundle:Evaluacion:new.html.twig', array('entity' => $entity,'form' => $form->createView(), )); //9
    }
} //10
```

Figura 14. Código del método createAction().

Representar el procedimiento en un grafo de flujo

A continuación se muestra el grafo de flujo correspondiente al código del método seleccionado:

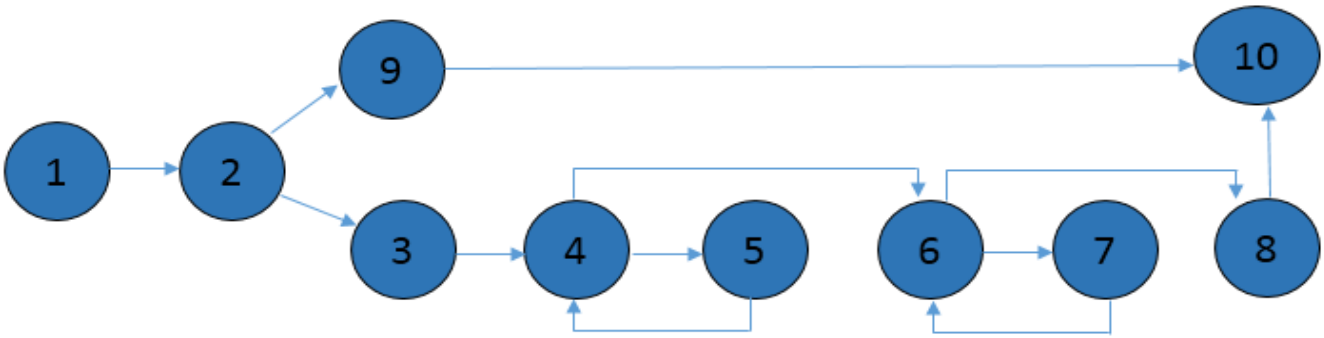


Figura 15. Grafo de flujo.

Calcular la complejidad ciclomática

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

1. V(G) = Aristas – Nodos + 2
   A: número de aristas N: número de nodos.
   V(G)= (12 - 10) + 2 = 4
2. V(G) = Nodos Predicado + 1
   P: número de nodos predicados (nodos de los cuales parten dos o más aristas).
   V(G)= 3 + 1 = 4
3. V(G)= R
   R: número de regiones del grafo.
   V(G)= 4

Luego de calculada la complejidad ciclomática mediante las tres fórmulas se evidencia que la complejidad es 4, lo cual indica que existen 4 caminos independientes, representando este valor el mínimo número de casos de pruebas.

Conjunto básico de caminos independientes

Los caminos independientes resultantes son:

Camino 1: 1-2-3-4-5-4-6-7-6-8-10

Camino 2: 1-2-3-4-6-7-6-8-10

Camino 3: 1-2-3-4-5-4-6-8-10

Camino 4: 1-2-9-10

Casos de prueba

Se ejecutan los casos de prueba para cada camino independiente determinado en el grafo de flujo.

Tabla 17. Caso de prueba camino 1.

CASO DE PRUEBA CAMINO 1	
Condición de ejecución	Se seleccionan todos los datos necesarios para la evaluación, luego se presiona el botón Guardar
Entrada	Proyectos, expertos, criterios técnicos y criterios comerciales, fecha de inicio y fecha de fin de la evaluación
Resultados esperados	Se crea una nueva evaluación, seguidamente se pasa a la interfaz Importancia Criterio

Tabla 18. Caso de prueba camino 2.

CASO DE PRUEBA CAMINO 2	
Condición de ejecución	No es seleccionado al menos un proyecto
Entrada	Proyectos, expertos, criterios técnicos y criterios comerciales, fecha de inicio y fecha de fin de la evaluación
Resultados esperados	Se muestra un mensaje de error informando al usuario que debe seleccionar al menos un proyecto

Tabla 19. Caso de prueba camino 3.

CASO DE PRUEBA CAMINO 3	
Condición de ejecución	No es seleccionado al menos un experto
Entrada	Proyectos, expertos, criterios técnicos y criterios comerciales, fecha de inicio y fecha de fin de la evaluación
Resultados esperados	Se muestra un mensaje de error informando al usuario que debe seleccionar al menos un experto

Tabla 20. Caso de prueba camino 4.

CASO DE PRUEBA CAMINO 4	
Condición de ejecución	Formulario no válido
Entrada	Proyectos, expertos, criterios técnicos y criterios comerciales, fecha de inicio y fecha de fin de la evaluación
Resultados esperados	Se informa al usuario las causas por las que no es válido el formulario

Análisis de los resultados

Para la validación del código generado en el desarrollo de la herramienta se seleccionaron los tres métodos principales del sistema, el análisis de los demás métodos se encuentra en el Anexo 6. A estos se le realizaron las pruebas para evaluar si el funcionamiento de cada uno de ellos se comportó de la manera esperada. Las pruebas realizadas a estas funcionalidades resultaron satisfactorias, comprobándose la estabilidad de la lógica aplicada en el código.

3.3.2 Pruebas de caja negra

Las pruebas de caja negra se centran en los requisitos funcionales del sistema. Con estas pruebas se intentan encontrar funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos

o en acceso a base de datos externas y errores de rendimiento. Se centran en qué hace el software y no en cómo lo hace. (Pressman, 2010)

Según (Pressman, 2010) existen varias técnicas para realizar este tipo de pruebas, se seleccionó la técnica partición equivalente, la cual permite comprobar los valores válidos e inválidos de las entradas existentes en la aplicación.

- *Partición equivalente:* método que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

La Figura 16 muestra el resultado de aplicación de esta prueba.

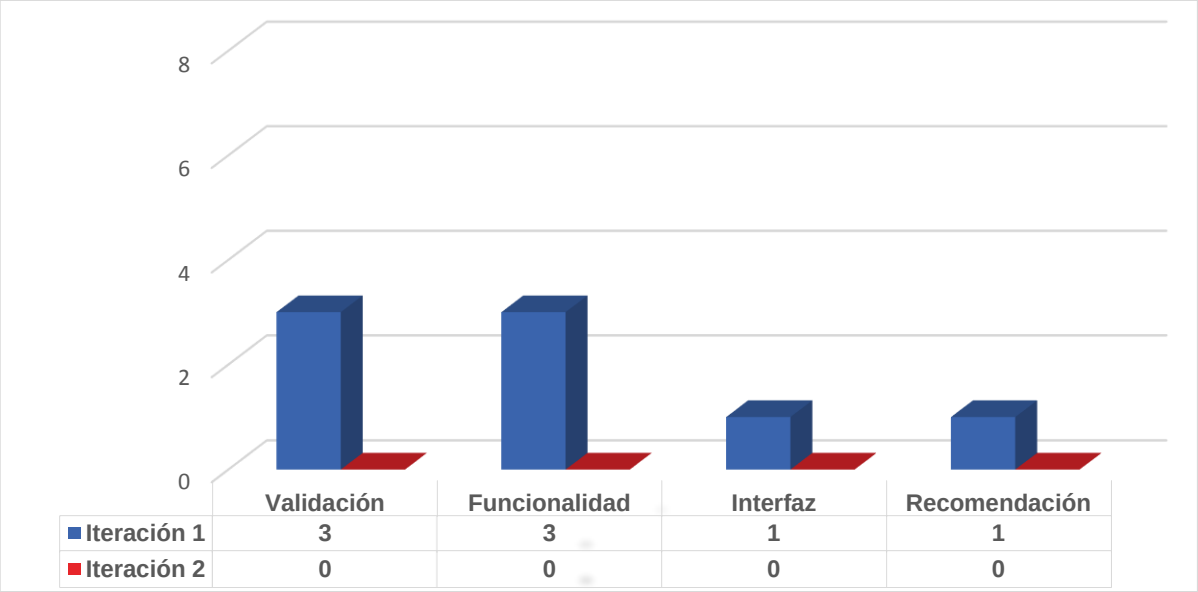


Figura 16. No conformidades detectadas.

Análisis de los resultados

Se realizaron dos iteraciones y un caso de prueba para cada requisito. En la primera iteración se detectaron 8 no conformidades, siendo las 8 de aplicación, desglosadas en 3 de validación, 3 de funcionalidad, 1 de interfaz y 1 de recomendación. En una segunda iteración estas no conformidades fueron solucionadas obteniéndose resultados satisfactorios.

3.3.3 Pruebas de aceptación

Las pruebas de aceptación están destinadas a evaluar si al final de una iteración se consiguió la funcionalidad que se esperaba y que esté en función de los requisitos establecidos inicialmente.

A continuación se muestra el Casos de Prueba de Aceptación (CPA en lo adelante) de los requisitos *Adicionar evaluación* y *Mostrar resultado de la evaluación de proyectos*, el resto de los CPA se encuentran en Anexo 7.

Tabla 21. CPA requisito funcional *Adicionar evaluación de proyectos*

CASO DE PRUEBA DE ACEPTACIÓN		
Código: CPA-19	**Historia de Usuario:** HU-19 Adicionar evaluación de proyectos	
Funcionalidad que se prueba: Adicionar evaluación de proyecto		
Condiciones de Ejecución: El usuario con permisos de especialista se autentica en el sistema. En la página principal en el menú lateral izquierdo se selecciona **Evaluación**.		
Acción	**Datos de entradas**	**Resultado esperado**
El especialista selecciona la opción **Adicionar.**		El sistema muestra un formulario para introducir los datos para adicionar una evaluación.
El especialista introduce los datos para crear una evaluación dejando campos vacíos.	Algunos datos de la evaluación.	El sistema valida los datos, al haber campos vacíos muestra un mensaje indicando existen campos vacíos. El sistema resalta los campos vacíos
Evaluación de la Prueba: Satisfactoria		

Tabla 22. CPA requisito funcional *Mostrar resultado de la evaluación de proyectos*

CASO DE PRUEBA DE ACEPTACIÓN	
Código: CPA-21	**Historia de Usuario:** HU-21 Mostrar resultado de la evaluación de proyectos
Funcionalidad que se prueba: Mostrar resultado de la evaluación de proyectos	
Condiciones de Ejecución: El usuario con permisos de especialista se autentica en el sistema.	

En la página principal en el menú lateral izquierdo se selecciona **Evaluación**.		
Acción	**Datos de entradas**	**Resultado esperado**
El especialista selecciona la opción **Listar.**		El sistema muestra un listado con las evaluaciones adicionadas en el sistema.
Si la evaluación está finalizadoa el especialista selecciona presiona el botón **Ver Resultados** en la columna **Acciones**.		El sistema muestra el resultado final de la evaluación seleccionada.
Evaluación de la Prueba: Satisfactoria		

Análisis de los resultados

Para validar que el resultado obtenido por el sistema corresponde con el resultado esperado por el cliente se diseñaron un total de 36 casos de prueba de aceptación. En una primera iteración en la ejecución de las pruebas se detectaron un total de 10 no conformidades entre interfaz y validación, las cuales fueron tratadas en una segunda iteración, para un resultado exitoso debido a que no se encontraron no conformidades. Terminadas estas pruebas y corregidas las no conformidades el cliente avaló la solución con la entrega del Certificado de Aceptación del Producto, el cual se encuentra en el Anexo 2.

3.4 Validación de la solución

Para la validación de la solución se consideraron los proyectos que se describen a continuación:

Servicio Despertador Matutino (SDM): Este proyecto tuvo como objetivo mejorar el servicio de despertador automático brindado por ETECSA en Cuba, para poder así reutilizar el personal subutilizado que tenían brindando este servicio, mejorando la eficiencia y productividad del trabajo además de introducir en el país la tecnología de Asterisk que es una de las tecnologías líderes en el mundo como solución de telefonía IP.

Sistema Penitenciario Cubano (SPC): Este proyecto tuvo como objetivo informatizar el sistema penitenciario cubano, aprovechando el desarrollo de las tecnologías que se llevaban a cabo en el país, para apoyar los procesos de trabajo que se ejecutan para el control, tratamiento y atención a los internos en los establecimientos penitenciarios y los correspondientes a los tres niveles de mando.

Para la evaluación de los proyectos con el modelo lingüístico 2-tuplas se seleccionaron dos expertos y un total de 6 criterios, 3 técnicos y 3 comerciales.

Para el análisis de los resultados obtenidos por la herramienta se determina el grado de aceptación de los proyectos con las siguientes medidas propuestas por (Abreu, 2012):

- *Casos positivos*: casos donde el resultado de la herramienta y la realidad coinciden.
- *Casos falsos positivos*: casos donde el resultado de la herramienta es factible y en la realidad el proyecto fue cancelado o no exitoso.
- *Casos falsos negativos*: casos donde el resultado de la herramienta es no factible y en la realidad fue factible.

Análisis de los resultados

A continuación se muestra el resultado obtenido basado en los criterios técnicos y comerciales, la tabla muestra el resultado arrojado por el sistema, así como el resultado real y el tipo de caso que representa, además de los valores α, β y S_i del modelo 2-tuplas.

Criterios evaluados

Criterios comerciales

- grado de comercialización
- demanda actual del producto
- impacto entre los productos existentes

Criterios técnicos

- hardware existente en la organización
- herramientas de software existentes
- soporte de software

Tabla 23. Comparación real contra sistema.

Proyectos	Lugar por el sistema	Lugar real	β	S_i	α	Factibilidad	Tipo de caso
SPC	1	2	3.75	4	-0.25	Muy alta	Falso positivo
SDM	2	1	3.35	3	0.35	Alta	Positivo

El proyecto SPC en cuanto al grado de comercialización es un producto que puede ser adaptable a otros países que posean un sistema judicial parecido, en cuando a la demanda actual e impacto, en su

momento el proyecto resolvería un problema que enfrentaban los centros penitenciarios cubanos. En el caso del proyecto SDM es un producto realizado a la medida, atendiendo a las necesidades de un cliente específico, en este caso ETECSA, por lo que no posee un alto grado de comercialización, de igual forma este resolvía un problema con el que contaba ETECSA. Con respecto a la factibilidad técnica los dos proyectos hacían uso de un grupo de herramientas de hardware y software con los que la universidad contaba, así como para brindar soporte una vez concluido y desplegado los sistemas. En el caso de SPC el sistema arrojó un falso positivo, dado que este fue cancelado luego de haber comenzado su desarrollo, por problemas internos de la universidad, no porque no fuese factible. El proyecto SDM arrojó un caso positivo lo que coincide con los resultados reales.

El valor β es el resultado del operador de agregación OWA, este redondeado es el valor correspondiente de i en S_i. El valor de S_i representa la factibilidad del proyecto, tomando el valor de la etiqueta lingüística correspondiente a la posición del valor de i, es mediante este resultado que el sistema ordena los proyectos evaluados, el proyecto SPC posee una factibilidad con un valor de muy alto, mientras que SDM posee un valor de factibilidad alto. El valor α representa la traslación simbólica, siendo esta la diferencia o incertidumbre de las valoraciones emitidas por los expertos en la evaluación de los criterios, en el resultado de factibilidad del proyecto SPC existe menor incertidumbre que en el resultado del proyecto SDM.

3.5 Conclusiones parciales

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- La confección de las tareas de ingeniería por iteración y de las tareas detalladas permitieron simplificar la implementación de las historias de usuario.
- El uso de estándares de codificación permitieron proyectar una calidad en el código generado.
- Las pruebas de caja blanca realizadas a varios algoritmos del sistema posibilitaron la detección de errores, obteniéndose resultados satisfactorios.
- Las pruebas de aceptación y de caja negra, permitieron comprobar el correcto funcionamiento de la herramienta luego de realizadas varias iteraciones.
- La validación de la solución demostró que la herramienta desarrollada contribuye al tratamiento de la incertidumbre en las valoraciones de los expertos en el análisis de factibilidad técnica y comercial.

CONCLUSIONES

Finalizada la presente investigación se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, resaltando que:

- La elaboración del marco teórico referencial de la investigación permitió el estudio del estado del arte y la selección de la técnica de soft computing: computación con palabras, así como las herramientas, lenguajes y la metodología de desarrollo a emplear en la implementación.
- Durante el análisis y diseño de la solución se obtuvieron los artefactos propuestos por la metodología de desarrollo seleccionada, lo que permitió facilitar la implementación de las funcionalidades definidas.
- Las pruebas realizadas permitieron garantizar un correcto funcionamiento de la herramienta y cumplimiento de las necesidades y requisitos del cliente, avalado mediante el certificado de aceptación.
- La herramienta informática obtenida contribuye al tratamiento de la incertidumbre en las valoraciones de los expertos en el análisis de factibilidad técnica y comercial.

RECOMENDACIONES

- Implementar una funcionalidad que permita la creación de fichas de proyectos cargando los datos desde ficheros Excel.
- Añadir criterios ambientales y sociales para el análisis de factibilidad.
- Integrar la solución con estudios de factibilidad económica y social.

REFERENCIAS BIBLIOGRÁFICAS

Abreu, Marieta Peña. 2012. *Modelo para análisis de factibilidad en la evaluación de proyectos de software.* 2012.

Almaguer, Daniarys Ramírez. 2009. *Etapas del análisis de factibilidad. Compendio bibliográfico.* 2009.

Almeira, Adriana Sandra y Pérez, V. 2007. *Arquitectura de Software: Estilos y Patrones.* Argentina: s.n., 2007.

Apache. 2014. Ibrugor. [En línea] 11 de junio de 2014. http://www.ibrugor.com/blog/apache-http-server-que-es-como-funciona-y-para-que-sirve/.

Castro, Maylé Díaz. 2010. *Método de evaluación de proyectos para decidir su aceptación.* 2010.

Choice, Expert. 2015. IOSA: Investigación de Operaciones SA. [En línea] 2015. http://www.iosa.com.pe/productos/expert-choice.

Christiansson, Benneth. 2008. *GoF Design Patterns-with examples using Java and UML2.* 2008.

Cleger, Eliober. 2015. *El proceso de aprobacion de un nuevo proyecto en la UCI.* La Habana, 2 de Abril de 2015.

DECIDE. 2013. ITESO. [En línea] 2013. http://www.iteso.mx/web/general/detalle?group_id=57482.

EasyPlanEx. 2014. BoraSystem. [En línea] 2014. http://www.borasystems.com/es/productos-software/easyplanex/ .

Eguiluz, Javier. 2012. *Desarrollo web ágil con Symfony2.* 2012.

Escalona, María José y Koch, Nora. 2002. *Ingeniería de Requisitos en Aplicaciones para la Web. Un estudio comparativo.* España: s.n., 2002.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. 2010. *Metodologías tradicionales vs. metodologías ágiles.* 2010.

Hernández, Rolando Alfredo. 2002. *Gestión de Proyectos para Informáticos.* Cuba, Universidad de las Ciencias Informáticas: s.n., 2002.

—. 2010. *Procedimiento para contratar proyectos informáticos.* La Habana: s.n., 2010. pág. 118.

Herrera, F. y Herrera-Viedma, E. 2000. *Linguistic decision analysis: Steps for solving decision problems under linguistic information.* s.l. : Fuzzy Sets and Systems, 115, 2000: s.n., 2000. págs. 67-82.

Herrera, F., Martínez, L. y Sánchez, P. J. 2005. *Managing non-homogeneous information in group decision making.* s.l. : European Journal of Operational Research: s.n., 2005. págs. 115-132.

IPMA, International Project Management Association. 2006. The Standard for Portfolio Management. Newtown Square Pennsylvania. 2006.

Joskowicz, José. 2008. *Reglas y prácticas en eXtreme Programing.* 2008.

Kidd, Mark Lorenz y Jeff. 1994. *Object-Oriented Software Metrics.* New Jersey: s.n., 1994.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* [ed.] Pablo Eduardo Roig Vázquez. [trad.] Luz María Hernández Rodríguez. Primera. México: PRENTICE HALL, 1999. págs. 189-209. ISBN: 970-17-0261-1.

Malfará, Dayvis. 2006. *Testing en eXtreme Programming.* 2006.

Martínez, L. Martínez y Ruan, D. y Herrera F. 2010. *Computing with Words in Decision support Systems: An overview on Models and Applications.* s.l.: International Journal of Computational Intelligence Systems : s.n., 2010. págs. 382-395. Vol. 3. ISSN 1875-6891..

Martínez, L. 1999. *Un nuevo modelo de representacion de informacion lingüistica basado en 2-tuplas para la agregacion de preferencias lingüisticas. .* España, Universidad de Granada: Tesis para optar al grado de Doctor en Informática: s.n., 1999.

Martínez, L. y Herrera, F. 2012. *An overview on the 2-tuple linguistic model for computing with words in decision making: Extensions, applications and challenges.* 2012. págs. 1-18. ISSN 0020-0255.

Mateu, Carles. 2004. *Desarrollo de aplicaciones web.* 2004.

NetBeans. 2015. NetBeans. [En línea] 2015. https://netbeans.org/index_es.html.

Palenzuela, Filiberto López. 2013. *Modelo para la toma de decisiones en los proyectos de software basado en los criterios de expertos.* 2013.

Paradigm, Visual. 2015. Knowledge Reuse Group. [En línea] 2015. http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

PHP. 2015. PHP.net. [En línea] 2015. http://www.php.net/manual/es/intro-whatis.php.

Piñero Pérez, P.Y. and autores, colectivo de. 2013. GESPRO Paquete para la gestión de proyectos. *Nueva Empresa.* La Habana: GECYT, 2013. Vol. 9, 1, pp. 45-53. ISSN: 1682-2455.

PostgreSQL. 2009. PostgreSQL. [En línea] 2009. www.postgresql.org.es.

Pressman, Roger S. 2010. *Ingeniería del Software. Un Enfoque Práctico.* Séptima Edición. 2010. pág. 646.

Recaman, Hernando Chaux. 2012. Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias. 2012.

Rodriguez, Diana. 2015. *¿Se realizan estudios de factibilidad tecnica y comercial en la UCI?* La Habana, 7 de Abril de 2015.

Romillo Tarke, Antonio and Oropesa Méndez, Daisy. 2013. *Nuevo modelo de universidad-empresa. El sistema UCI.* La Habana: Editorial Universitaria Félix Varela, 2013. 2306-918X.

Rosa, Martín M. 2005. *Guía Práctica para el Diseño, Administración, y Evaluación de Proyectos Sociales.* 2005.

Rouse, M. 2007. *What is integrate development environment (IDE).* 2007.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2000. *El Lenguaje Unificado de Modelado. Manual de referencia.* [trad.] Addison Wesley Longman. Madrid. España : Pearson Educación, 2000. ISBN: 84-7829-037-0.

SGBD. 2007. Desarrollo Web. [En línea] 31 de julio de 2007. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

Sommerville, Ian. 2007. *Ingeniería del Software.* Octava Edición. Madrid, España: s.n., 2007.

Vega, José I. 2015. *Los estudios de viabilidad para negocios.* Mayagüez: s.n., 2015.

Vega, Yunier. 2015. *La contratación de un nuevo proyecto en la uci. El papel de la factibilidad tecnica y comercial.* La Habana, 13 de Abril de 2015.

Veliz, Yeleny Zulueta. 2014. *Modelos de evaluación de la importancia del impacto ambiental en contextos complejos bajo incertidumbre.* Cuba: s.n., 2014.

Yager, R. R. 1988. *On ordered weighted averaging aggregation operators in multicriteria decisionmaking.* s.l. : IEEE Transactions on Systems, Man and Cybernetics: s.n., 1988. págs. 183-190.

Zadeh, Lotfi A. 1996. *Nacimiento y evolución del la lógica borrosa, el soft computing y la computación con palabras. .* 1996. págs. 421-429. Vol. 8.

ANEXOS

Anexo 1. Entrevista realizada a especialistas del Centro de Negocio UCI y de los Centros Productivos CEIGE y CEGEL, facultad 3.

Entrevista

(Realizada a especialista del Centro de Negocio UCI, y especialistas de los centros CEIGE y CEGEL)

Objetivo: comprobar cómo se desarrollan los análisis de factibilidad técnica y comercial para la ejecución de proyectos en la UCI.

Compañero o compañera.

Se necesita su colaboración para la realización de esta investigación.

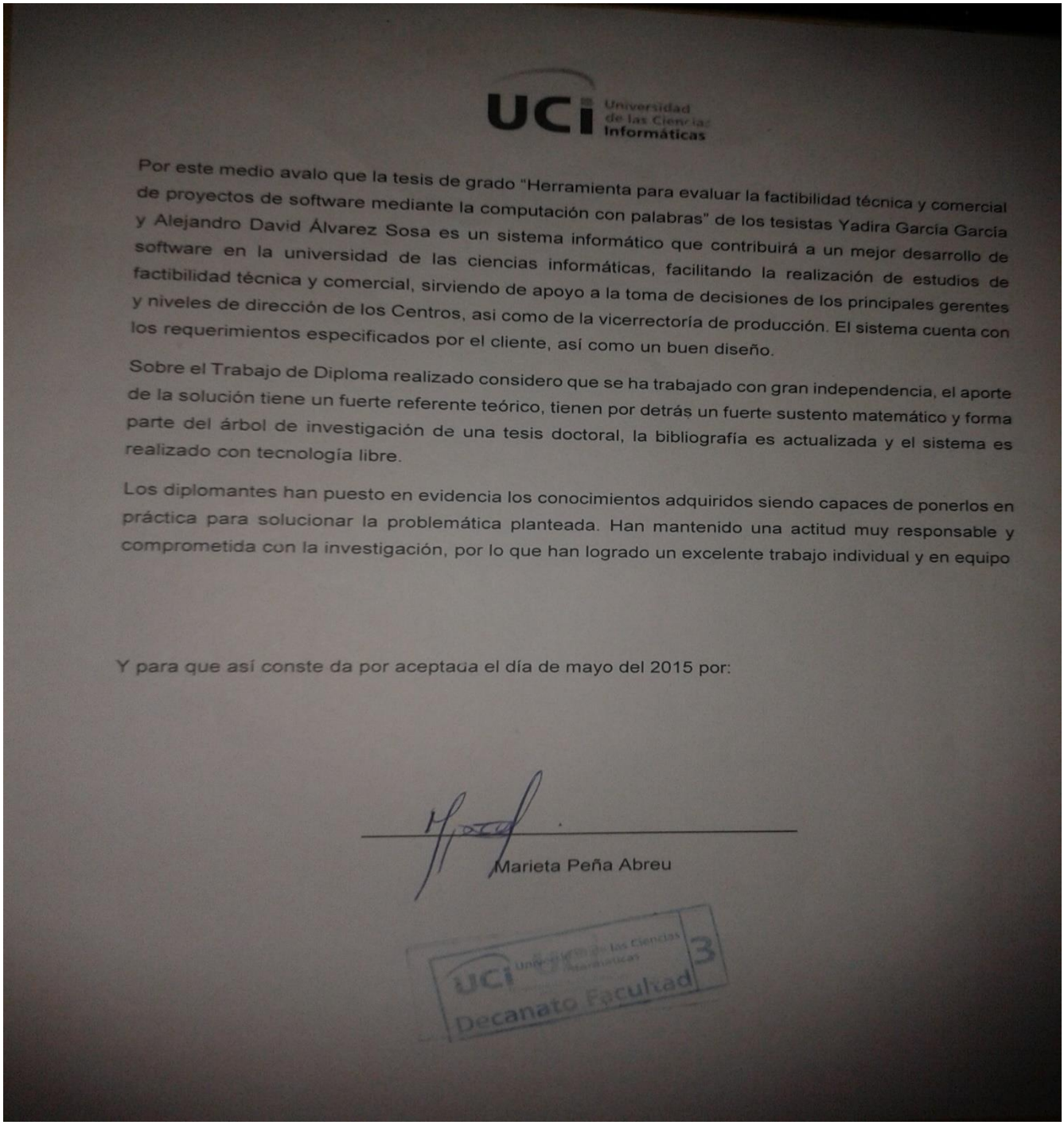
Datos generales

- Nombre y Apellidos
- Rol que desempeña

Aspectos a entrevistar

1. ¿Se realizan en el centro análisis de factibilidad antes de dar comienzo a un proyecto?
2. ¿Se tienen en cuenta criterios técnicos y comerciales para el análisis de factibilidad técnica y comercial? Menciones algunos de ellos en caso positivo.
3. ¿Existe actualmente en el centro personal capacitado para llevar a cabo estos análisis de factibilidad?
4. ¿Conoce usted acerca de la existencia de modelos y herramientas que contribuyen al análisis de factibilidad técnica y comercial para la evaluación de proyectos?

Anexo 2. Certificado de Aceptación del Producto



UCI Universidad de las Ciencias Informáticas

Por este medio avalo que la tesis de grado "Herramienta para evaluar la factibilidad técnica y comercial de proyectos de software mediante la computación con palabras" de los tesistas Yadira García García y Alejandro David Álvarez Sosa 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 técnica y comercial, 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.

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 fuerte sustento matemático 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 el día de mayo del 2015 por:

Marieta Peña Abreu

UCI Universidad de las Ciencias Informáticas Decanato Facultad 3