UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Título: "Herramienta para evaluar la factibilidad técnico-comercial en proyectos de software haciendo uso de la lógica difusa".

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

Autores:

Bárbara Salinas Martínez

Isis Bertamí Barrios

Tutor:

Msc. Marieta Peña Abreu.

Ing. Carlos Rafael Rodríguez

Ciudad de La Habana, Febrero del 2014

"Año 56 de la Revolución"

Declaración de autoría

Declaramos que somos los únicos autores del trabajo “Herramienta para evaluar factibilidad técnica-comercial en proyectos de software haciendo uso de la lógica difusa” y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Autor: Bárbara Salinas Martínez Autor: Isis Bertamí Barrios

_____ _____

Tutor: Msc. Marieta Peña Abreu Tutor: Ing. Carlos Rafael Rodríguez

Dedicatoria

De Baby:

A mis padres por todos los sacrificios que hicieron para que yo me dedicara solamente al estudio sin ninguna otra preocupación, además de todo su amor y cariño

A mi hermano y a mi familia en general

A todos los amigos que me apoyaron de una forma u otra porque gracias a ellos estoy aquí hoy.

De Isis:

A mis padres por todo el amor y apoyo que me han dado todos estos años, sin su cuidado y confianza esto no sería posible.

A mi hermana por ser mi mejor amiga y darme siempre el mejor ejemplo a seguir.

A toda mi familia y mis amigos.

Resumen

La selección de proyectos de software es un proceso complejo debido a los acelerados cambios del entorno y a los múltiples criterios que entran en conflicto durante este. Una correcta selección de proyectos conlleva a la obtención de un producto con mayor calidad, menor costo y en menor período de tiempo. En la siguiente investigación se propone una herramienta para realizar análisis de factibilidad en el orden técnico y comercial, a proyectos de software, mediante el método Proceso Analítico Jerárquico Difuso por sus siglas en inglés (FAHP). A partir de la utilización del método se evalúan los proyectos a partir de criterios técnicos y comerciales ya definidos. La herramienta recibe como entrada un listado de proyectos y una vez terminado el proceso de evaluación se devuelve como salida el listado con los proyectos evaluados y ordenados según la factibilidad arrojada. La utilización de la herramienta tiene como objetivo contribuir a un perfeccionamiento en el proceso de selección permitiendo reducir la imprecisión presente en la toma de decisiones. El sistema se desarrolló a partir de tecnologías libres, se utilizó como metodología de desarrollo la programación extrema, el lenguaje de programación PHP junto al marco de trabajo Symfony. Todo el desarrollo fue verificado y probado mediante pruebas unitarias y de aceptación, obteniéndose una herramienta para evaluar factibilidad técnica y comercial en proyectos de software basada en la lógica difusa.

Palabras Claves: evaluación de proyectos, factibilidad técnica, factibilidad comercial, incertidumbre, lógica difusa.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la época moderna ha surgido la gestión de proyectos como un área de conocimiento interdisciplinaria que logra el control y la coordinación de un grupo de personas encaminadas a cumplir un objetivo común, guiados por un líder mediante la utilización de técnicas de dirección y la gestión de recursos humanos, potenciando la toma de decisiones (Institute 2013). La gestión de proyectos ha ocupado un lugar importante dentro del desarrollo de software por la forma en que varían los proyectos de software en magnitud, alcance, inversión, complejidad, costos y esfuerzos.

El acelerado desarrollo del mercado y los productos complejiza la toma de decisiones en los proyectos de software modernos, exigiendo un conocimiento de mayor confiabilidad en la industria del software. Debido a esto se hace necesario realizar estudios de factibilidad para la correcta selección de proyectos de software a ejecutar, estos estudios tienen como objetivo determinar el potencial de mercado de productos o servicios, teniendo en cuenta diferentes criterios que permitan que las proyecciones de la empresa sean cumplidas (Acevedo, Edna y Barrios 2010a).

Uno de los factores que determina el éxito de un proyecto de software es el grado de factibilidad que se presente luego de evaluar tres criterios fundamentales: el técnico, el económico y el comercial, centrándose la investigación en los técnicos y comerciales. La factibilidad técnica permite establecer la infraestructura necesaria para atender un mercado objetivo, así como la mejora del sistema actual, mientras la factibilidad comercial se encarga de estudiar el mercado a través de la información para identificar tanto las oportunidades como amenazas del entorno (Peña Abreu y Piñero Pérez 2013).

Existen personas encargadas de realizar los estudios de factibilidad dentro de una empresa. Estas son las responsables de tomar las principales decisiones y generalmente se sienten mejor presentando sus juicios como un intervalo, en vez de dar un valor puntual y fijo (Büyüközkan, Kahraman y Ruan 2004). Esto se debe a la vaguedad[1] presente en las respuestas de los seres humanos (Buckley 1985). Este problema es resuelto a través de la lógica difusa, que proporciona un mecanismo de inferencia que permite modelar la incertidumbre de los procesos cognitivos de forma que puedan ser tratados por una computadora (Zadeh 1974).

Cuba no se encuentra ajena a los cambios del entorno y busca nuevas alternativas para insertarse en el mercado mundial, una de ellas es la informatización de los principales procesos de la sociedad y el desarrollo de software. Cada vez se hace más compleja la correcta selección de proyectos a ejecutar, factor de vital importancia dentro de la industria de software por la cantidad de recursos que demandan. Provocando la necesidad de implantar modelos que permitan modelar la incertidumbre para realizar la

[1] **Vaguedad:** Se refiere a la falta de precisión o exactitud.

correcta selección de estos proyectos y se logre aumentar la competitividad a partir de una adecuada planificación.

La Universidad de las Ciencias Informáticas (en lo adelante UCI), se presenta como una institución destinada al desarrollo de la industria de software, mediante la formación de profesionales calificados en el tema y el desarrollo de soluciones informáticas con la vinculación de sus estudiantes y profesores a la producción. La UCI se encuentra abierta a nuevos proyectos informáticos lo que trae consigo que se haga necesario determinar la factibilidad técnica y comercial de los mismos para saber cuáles son los que deben ser puestos en ejecución.

Actualmente los principales jefes de proyectos no aplican modelos, procedimientos y herramientas para realizar estudios de factibilidad, debido a que estos por lo general no suelen ser exactos y para aplicarlos hay que realizar un estudio de cómo son utilizados para lo que requiere de tiempo y esfuerzo. Todo eso trae consigo que temas como la factibilidad técnica y comercial no son tratados adecuadamente. Además se decide la ejecución de algunos proyectos sin un estudio de factibilidad previo unido a la ausencia de una herramienta que tenga en cuenta la incertidumbre presente ante la toma de decisiones.

Los análisis de factibilidad de los proyectos en la UCI, algunos se realizan de forma automatizada a través de un paquete de gestión de proyectos denominado GESPRO, que actualmente cuenta con un grupo de funcionalidades referidas al ‘área económica, dejando atrás la evaluación de criterios técnicos y comerciales.

Por lo visto anteriormente se plantea el siguiente **problema a resolver**: ¿Cómo determinar la factibilidad técnica y comercial en proyectos informáticos bajo condiciones de incertidumbre?

Objeto de estudio:

Estudios de factibilidad técnica y comercial de proyectos

Objetivo General:

Desarrollar una herramienta informática que contribuya a realizar el análisis de factibilidad técnico y comercial basada en la lógica difusa que facilite la realización de estos estudios bajo condiciones de incertidumbre.

Campo de acción:

Estudios de factibilidad técnica y comercial en proyectos de software bajo condiciones de incertidumbre.

Objetivos específicos:

- Construir el marco teórico de la investigación relacionado con el análisis de factibilidad de proyectos y en particular los de software, así como las tendencias de la lógica difusa y las herramientas informáticas para el desarrollo de aplicaciones.
- Realizar el análisis y diseño de la solución propuesta, mediante la metodología de desarrollo de software seleccionada.
- Implementar todas las funcionalidades definidas para la obtención de la solución, mediante las tecnologías y herramientas elegidas.
- Analizar los resultados de la investigación mediante técnicas de verificación y validación y su aplicación a un caso de estudio.

Hipótesis:

Si se desarrolla una herramienta informática para realizar análisis de factibilidad técnica y comercial de proyectos de software mediante el uso de la lógica difusa permitirá tolerar la incertidumbre presente en la información.

Métodos teóricos:

Histórico Lógico: En la investigación se realiza un estudio de la problemática anunciada, revisando las ventajas y desventajas de los sistemas de evaluación de proyectos de software existentes, estableciendo así una conexión entre su concepción histórica y la actualidad.

Hipotético-Deductivo: Se realiza un análisis hipotético-deductivo ya que a partir de la problemática existente en la universidad, se plantean los objetivos generales y específicos de la investigación, proponiendo la hipótesis, a la cual se dará seguimiento dando respuesta en el transcurso de la investigación, llegando a introducir nuevos conocimientos que posteriormente serán evaluados.

Modelación: Permite la descripción de la propuesta de solución, basándose en la aplicación de restricciones y modelos matemáticos que posibilitaron la conceptualización y desarrollo de la herramienta para evaluar la factibilidad técnica y comercial.

Método Empíricos:

Entrevista: Se aplicó a varios especialistas en mercadotecnia del centro CEGEL, como se observa en el anexo 5, para comprobar la forma en que se realizaba el proceso de evaluación de los proyectos de software en este centro.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo son abordados los principales conceptos relacionados con la investigación. Se hace un estudio de los principales modelos multicriterio de apoyo a la toma de decisiones entre los que se encuentra el método AHP con lógica difusa, que es el que se propone a utilizar, y de las herramientas existentes en las que se basan estos modelos. Se hace un estudio de varias tecnologías existentes para el desarrollo de aplicaciones de este tipo y cuáles serán utilizadas para el desarrollo de la herramienta a realizar. Además se hace un análisis de la bibliografía existente para el tema de investigación, citando a los principales autores que demuestran dominio en el tema.

1.2 Análisis Bibliográfico

En esta sección se realiza un análisis de las bibliografías consultadas identificando las principales fuentes y escuelas del área. Dentro de la revisión se destaca fundamentalmente el estudio de artículos en revistas referenciadas donde se tratan fundamentalmente las aplicaciones del método FAHP, así como la utilización de la lógica difusa, para una mejor selección de alternativas. Las referencias son generalmente a partir del año 2000, encontrándose marcadas evidencias a partir de estos años la necesidad de aplicar técnicas para la selección, donde se tenga en cuenta la incertidumbre de la información así como la utilización de métodos cuantitativos y cualitativos para la selección de alternativas.

Tabla 1. Análisis bibliográfico. (Elaboración propia).

	Últimos 5 años	Años Anteriores
Libros y monografías	5	5
Tesis de Doctorados	1	1
Tesis de Maestría	1	1
Artículos en revistas científicas	22	18
Reportes técnicos y conferencias	-	1
Total de referencias	**29**	**26**

1.3 Principales Conceptos

Asumiendo la definición (Rodríguez Batista 2005) para proyecto, de (Perissé 2001) para proyecto informático, de (Serrano y Avilés 2014) para gestión de proyectos y de (Acevedo, Edna y Barrios 2010b) para la factibilidad, en la presente investigación para un mayor entendimiento y comprensión de la investigación se definen los conceptos relacionados con la temática abordada.

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(Bacca y others 2015).

Factibilidad comercial: El estudio de mercado se puede definir como la función que 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; para generar y evaluar las medidas de mercado así como para mejorar la comprensión del proceso. Éste, por su carácter preliminar, constituye un sondeo de mercado, antes de incurrir en costos innecesarios(Daniarys 2012).

Lógica Difusa: Forma de procesamiento de información en la que los datos podrían tener asociados un grado de pertenencia parcial a conjuntos(Zadeh 1974).

1.4 Métodos de decisión multicriterio

La toma de decisiones es un proceso crítico cuando se desea seleccionar la mejor vía para alcanzar un objetivo en un análisis multicriterio, aunque existan diferentes caminos, pueden ser difícilmente identificables, ya que en algunos casos el objetivo está rodeado de múltiples variables, denominadas criterios, que aumentan la complejidad de la decisión a tomar ya que los criterios suelen hacer referencia a elementos muy variados. En un proceso de decisión, las posibles soluciones, denominadas alternativas, deben ser factibles para la consecución de una solución óptima. No existe una única solución suficientemente argumentada aunque los métodos de evaluación y decisión multicriterio que se están utilizando en problemas actuales, ayudan a solventar los inconvenientes del proceso, resultando ser muy flexibles en la aceptación de modificaciones. Sin embargo siempre hay que tener en cuenta, que aunque estos métodos sirven eficazmente de apoyo a la toma de decisiones; la decisión final siempre queda en manos de un grupo de expertos(Iturbe y del Valle Campoamor 2011).

Entre los métodos de toma de decisión multicriterio se encuentran: el [2]AHP (Cascales 2009) , TOPSIS[3] (Aznar Bellver y Guijarro Martínez 2012), PROMETHEE[4] (Vincke y Brans 1985). Estos métodos ofrecen mecanismos a partir de los cuales se pueden ordenar las alternativas y brindar criterios para tomar la decisión. En la tabla 2 se muestra una comparación entre estos métodos de toma de decisión multicriterio según un conjunto de elementos propuestos por (Arza Pérez, Verdecia Martínez y Lavandero García 2013):

1. Punto de partida: información inicial para realizar la evaluación de las alternativas.
2. Permite múltiples criterios: (Sí/No) si el método permite el tratamiento de múltiples criterios.
3. Permite múltiples expertos: (Sí/No) si el método permite el trabajo con múltiples expertos, que facilite integrar las opiniones y tenga en cuenta las características de los expertos.
4. Tratamiento de la incertidumbre: (Sí/No) si el método da tratamiento a la incertidumbre.
5. Modelado lingüístico: (Sí/No) si el método usa variables lingüísticas.
6. Método de ordenamiento: método que se utiliza para el ordenamiento de las alternativas.
7. Uso de operadores de agregación: (Sí/No) si emplea operadores de agregación.

Tabla 2 Tabla comparativa de los métodos de toma de decisión multicriterio (Arza Pérez, Verdecia Martínez y Lavandero García 2013).

I	AHP	TOPSIS	PROMETHEE
1	Evaluaciones de los criterios para la decisión para cada alternativa.		
2	Sí	Sí	Sí
3	Se puede aplicar el método con múltiples expertos, no deja explícitamente establecida la forma de integrar los criterios de los expertos, no da tratamiento al consenso (el propio operador de agregación define el consenso) y la decisión en grupo.		
4	Sí	Sí	Sí
5	Sí	Sí	Sí
6	Se establece una jerarquía de los criterios y su relación con las alternativas; aplicando un operador de agregación se consolida la información.	Se establece un ideal positivo y un ideal negativo a partir de un índice que combina la distancia de cada alternativa a los ideales y se ordenan en función de ese índice.	Se propone un orden entre las alternativas utilizando medidas de distancia para determinar la influencia negativa y positiva de una alternativa en función de cuáles son mejores o peores que ella.
7	Sí	Sí	Sí

[2] **AHP:** Analytic Hierarchy Process
[3] **TOPSIS:** Technique for Order Preference by Similarityto Ideal Solution
[4] **PROMETHEE**: Preference Ranking Organization Methods for Enrichement Evaluations

Luego de comparar cada uno de los métodos estudiados según un grupo de criterios y teniendo en cuenta la tesis de maestría de la Master en Ciencia Marieta Peña Abreu “Modelo para análisis de factibilidad en la evaluación de Proyectos de software”, donde se plantea el uso del método AHP para realizar estudios de factibilidad técnica y comercial en proyectos de software, y contribuir a la selección de los mejores proyectos atendiendo a los resultados dados por el método, se puede afirmar que el AHP resuelve las necesidades planteadas, debido a que presenta un sustento matemático que garantiza su efectividad y permite desglosar y analizar un problema por partes, además dicho método tiene como una de sus aplicaciones: la evaluación de proyectos de software siguiendo criterios técnicos, comerciales ya definidos.

Si se considera que al realizar un análisis que incluya la opinión de expertos se debe tener en cuenta que los mismos se sienten más cómodos representando los juicios de forma difusas con valores como (“entre 6 y 7”, “no más de 5”, etc.) o lingüísticos (bueno, medio, malo, etc.), ya que ellos son incapaces de explicar sus preferencias, dado el carácter difuso de comparación(Zapata Cortés, Arango Serna y Adarme Jaimes 2012). Se puede decir que esta condición humana hizo necesario que se le añadiera una matriz borrosa al método AHP, permitiendo integrar la vaguedad presente en la respuesta de las personas involucradas en la toma de decisiones, acercarse a la realidad humana y proporcionar análisis de toma de decisiones con más validez.

Esta modificación del método se llevó a cabo utilizando la lógica difusa, esta proporciona un mecanismo de inferencia que permite simular los procedimientos de razonamiento humano en sistemas basados en el conocimiento. La teoría de la lógica difusa proporciona un marco matemático que permite modelar la incertidumbre de los procesos cognitivos humanos de forma que pueda ser tratable por un computador(Zadeh 1974).

La unión de la lógica difusa y el método AHP dio como resultado al Proceso Jerárquico Analítico Difuso (en lo adelante FAHP[5]) el cual se definió en tres pasos. "La primera implica el uso de números difusos triangulares para transformar las ideas de expertos en una matriz de reciprocidad positiva. En segundo lugar, un método de media geométrica que pesa los valores parciales para cada opción, con la conexión jerárquica establecida y, finalmente, una función de permanencia para cada opción que desarrolló la clasificación de prioridades"(Díaz et al. 2012)

[5] **FAHP**: Proceso Jerárquico Analítico Difuso

1.4.1 FAHP

El desarrollo de este método se lleva a cabo de forma muy parecida a la del método tradicional, pero utilizando los números borrosos triangulares que son los más usados en la práctica por su relativa comodidad de manipulación.

Este método ha tenido un gran auge en los últimos años haciendo que su estudio y aplicación gane gran importancia en diferentes campos como la administración siendo uno de los más populares así como el medio ambiente, la construcción, el comercio entre otros.

También debemos destacar que la importancia del estudio de este método ha sido mundial destacando entre los principales países que investigan acerca del mismo a Taiwán, China, Irán, y en el Continente Americano destacan países como Colombia, México y Los Estados Unidos de América.

En la siguiente tabla se puede apreciar como las publicaciones acerca de este método han ido en aumento en la última década.

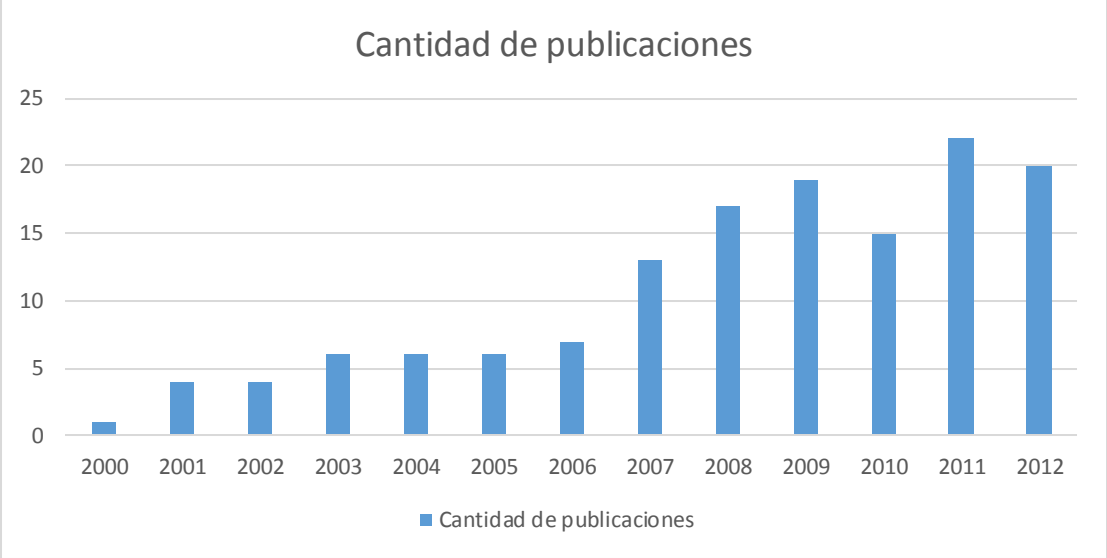

Ilustración 1 Cantidad de publicaciones acerca del método FAHP en la última década(Sánchez Reig 2015).

Si se realiza una comparación con respecto al método tradicional AHP y al FAHP se puede afirmar que los estudios realizados con respecto a este último como plantean (Huang y Wu 2005) demuestran como con la ayuda de la teoría de conjuntos difusos, se resuelven algunos defectos encontrados en el método tradicional de AHP, tales como la aplicación de escalas limitadas para la explicación de las consideraciones de expertos, la correlación entre los atributos para la toma de decisiones, la imprecisión, la ambigüedad y la incertidumbre de expertos para encontrar los valores de las comparaciones.

Los principales pasos para el desarrollo del método son:

1. Desarrollo de la estructura jerárquica para los criterios y alternativas
2. Representación difusa de los juicios, en una comparación por pares para criterios de evaluación y alternativas
3. Construcción de las matrices de juicio difuso para el AHP
4. Operaciones matemáticas:

Una vez se construyen las matrices de comparación por pares, se deben hacer los cálculos pertinentes para el desarrollo de la metodología, los cuales son (Cascales 2009):

- El cálculo del índice de consistencia
- El cálculo de los vectores de peso para cada nivel de la jerarquía mediante el análisis extendido y los principios de comparación de números difusos.

Para el desarrollo de esta investigación se estudiaron diferentes modelos que hacen uso de este método como lo son:

1. Modelo para la gestión de proveedores utilizando AHP Difuso(Herrera Umaña y Osorio Gómez 2006).
2. Aplicación del Proceso Analítico Jerárquico con Lógica Difusa para la selección de software para logística(Zapata Cortés, Arango Serna y Adarme Jaimes 2012).

El estudio de estos documentos permitió desarrollar un conocimiento de cómo se lleva a cabo el método según diferentes autores. Cada uno aplicado a un área diferente y con resultados palpables que permiten ver como son manejadas las diferentes variables que intervienen en el Proceso Analítico Jerárquico en cada paso del mismo.

Se puede señalar como desventaja que en ningún documento de los estudiados se hace referencia a la selección de proyectos por tanto no se cuenta con una herramienta que haga uso de este método para apoyar la toma de decisiones y la selección de las mejores alternativas.

1.5 Herramientas informáticas para la evaluación de factibilidad de proyectos

Para la ayuda en la toma de decisiones existen actualmente varias herramientas que hacen el trabajo de los usuarios más fáciles respecto a este tema, a continuación mencionaremos algunas de ellas que fueron objeto de estudio en la investigación.

1.5.1 Expert Choice

El software de Expert Choice está diseñado para ayudar en la toma de decisiones, de forma que se puedan sobrepasar los límites de la mente humana para sintetizar datos, intuición, recursos limitados y objetivos. Está basado en el AHP, el software ofrece un acercamiento estructurado y un proceso

probado para prioridades y toma de decisiones. Expert Choice no solo ayuda a personas en la toma de decisiones para llegar a la mejor solución, sino que ofrece un claro razonamiento del porque tomar esa decisión en particular. Permite integrarse con Microsoft Project y bases de datos de Oracle, especificar los roles de los participantes y desarrollar una jerarquía de alternativas. También ofrece un conjunto de planillas que le permiten guardar múltiples escenarios de recursos(Ishizaka y Labib 2009).

1.5.2 EasyPlanEx

EasyPlanEx es un software especializado en proyectos financieros, el cual facilita el desarrollo de modelos matemáticos y reportes. Permite realizar análisis de sensibilidad de forma automática medir el riesgo del proyecto a través de la técnica de Montecarlo, generar de forma automática gráficos y la documentación del proyecto. Además realiza análisis de impacto de forma automática y genera múltiples reportes e informes. Actualmente se encuentra disponible en dos idiomas inglés y español, es compatible solo con el sistema operativo windows, este software se entrega en siete versiones similares en funcionalidad pero con capacidades diferentes y limitaciones de uso, propias de cada una, es un herramienta privativa que tiene un costo de uso por su licencia, posee una versión de prueba de 15 días sin costo de licencia(Chain 2007).

1.5.3 DECIDE

Es una herramienta diseñada para entrenar de modo rápido y eficaz las habilidades requeridas en el proceso relativo a la toma de decisiones, haciendo referencia a aquellas decisiones requeridas en situaciones complejas, que cambian constantemente. Entre sus características principales se encuentra que permite realizar estudios de factibilidad tanto técnica como comercial, también es capaz de elaborar planes de negocio y de tomar varias decisiones a la vez, en lugar de una sola. Facilita el establecimiento de un ambiente de comunicación entre los miembros del equipo de trabajo generando aptitudes como la colaboración, el liderazgo y la negociación(Chain 2007).

1.5.4 GESPRO

GESPRO se presenta como una solución informática que posibilita la informatización de las organizaciones orientadas a proyectos. Posibilita no sólo la informatización de las organizaciones, sino que contribuye a la mejora integral de los procesos de la organización alineado con las mejores prácticas para la dirección estratégica. Entre sus características principales se encuentran (PIÑERO-PÉREZ 2013):

- Permite el control y seguimiento de proyectos a diferentes niveles: nivel de proyecto, nivel de sucursal y nivel gerencial de una corporación.
- Incorpora para el control y seguimiento de proyectos un cuadro de mando integral y un sistema de indicadores que permita la toma de decisiones en cascada, dosificando la información y permitiendo llegar en caso requerido hasta detalles del funcionamiento de las organizaciones y sus proyectos.
- Permite el control y seguimiento de programas de proyectos y de sucursales de una organización.

1.5.5 Valoración de las herramientas

A continuación se muestra una tabla comparativa entre las diferentes herramientas estudiadas para la evaluación de proyectos:

Tabla 3. Comparación de herramientas para evaluación de proyectos.

Herramientas	Evalúan criterios técnicos-comerciales	Tratamiento de Incertidumbre	Utilizan AHP	Formato de reporte	Plataforma	Propiedad	Costo (USD)
Decide	Si	No	No	Pdf, Excel, Word	Windows	Privativa	$ 2900 para un mes
EasyPlanEx	No	Si(Técnica de Montecarlo)	No	Pdf	Windows	Privativa	$ 1960 para un año
Expert Choice	No	No	Si	-	Windows	Privativa	$ 2500 para un mes
Gespro	No	No	Si	Pdf, XML	Windows/Linux	Libre Costo	Libre Costo

Las herramientas antes estudiadas presentan un conjunto de ventajas para la toma de decisiones durante el proceso de evaluación de un proyecto. Están basadas en el uso de métodos y modelos

utilizados en la gestión de proyectos. A pesar de esto también cuentan con un conjunto de desventajas como que no tienen en cuenta la factibilidad técnica y comercial, son herramientas privativas por lo que hay que pagar un costo por su licencia, tienen un tiempo de utilización restringido y una cantidad de usuarios determinados para utilizarlas, son dependientes de una plataforma porque están diseñadas para un sistema operativo en específico. Además la mayoría de estas herramientas están destinadas a ser usadas en campos como las ciencias económicas y financieras, las inversiones productivas, la agricultura, no en las ciencias informáticas por lo que no satisfacen las necesidades de la Universidad de las Ciencias Informáticas de acuerdo a los proyectos que se realizan en la misma. Por lo antes expuesto se hace necesario la realización de una nueva herramienta que elimine las desventajas expuestas con anterioridad.

1.6 Ingeniería de requisitos

La ingeniería de requisitos (IR) del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Cuenta con los mecanismos adecuados para comprender las necesidades del cliente, especificando la solución sin ambigüedad. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto el software sobre el negocio(PRESSMAN 2005).

1.6.1 Actividades de la ingeniería de requisitos

Existen cuatro actividades básicas que se tienen que llevar a cabo para completar el proceso. Estas actividades ayudan a reconocer la importancia que tiene para el desarrollo de un proyecto de software realizar una especificación y administración adecuada de los requisitos de los clientes.

Las cuatro actividades son: extracción, análisis, especificación y validación, y serán explicadas a continuación cada una de ellas (Chaves 2011).

Extracción de los requisitos

Esta fase representa el comienzo de cada ciclo y en esta se realizan las actividades involucradas en la captura de los requisitos del sistema. Aquí, los analistas de requisitos deben trabajar junto al cliente para descubrir el problema que el sistema debe resolver, los diferentes servicios que el sistema debe prestar, las restricciones que se pueden presentar. Es de suma importancia, que la extracción sea efectiva, ya que la aceptación del sistema dependerá de que esta satisfaga las necesidades del cliente. En esta fase se realizan las siguientes tareas:

- Revisión documental: Es una técnica de revisión y de registro de documentos que fundamenta el propósito de la investigación y permite el desarrollo del marco teórico y/o conceptual.
- Entrevistas: Los requisitos son identificados muchas veces mediante respuestas que emiten los clientes a preguntas realizadas en entrevistas, que permiten obtener una información general

de lo que hace el cliente y de cómo podría interactuar con el sistema. Se utiliza también para identificar los requisitos no funcionales del sistema.

- Identificación de las reglas del negocio: Se realiza en conjunto con el cliente donde se recogen en un documento las normas y políticas por las que se rige el negocio.

Los artefactos que se generan son:

- Descripción de procesos de negocio.
- Reglas del negocio.
- Especificación de requisitos de software.

Análisis de los requisitos

Sobre la base de la extracción realizada previamente, comienza esta fase, que se enfoca en descubrir problemas con los requisitos del sistema identificados hasta el momento. Usualmente se hace un análisis luego de haber producido un bosquejo inicial del documento de requisitos; en esta etapa se leen los requisitos, se conceptúan, se investigan, se intercambian ideas con el resto del equipo, se resaltan los problemas, se buscan alternativas y soluciones, y luego se van fijando reuniones con el cliente para discutir los requisitos.

Tareas que se realizan:

- Analizar requisitos del cliente: Se identifican ambigüedades, inconsistencias, requisitos comunes y escenarios arquitectónicos requeridos para dar soporte a los requisitos funcionales.
- Talleres de análisis: El propósito de estos talleres es analizar los requisitos para identificar otros o modificar los existentes.
- Reuniones con el cliente: En estas se analizan los requisitos identificados con el propósito de aprobarlos luego de llegar a un acuerdo con el cliente.

El artefacto que se actualiza durante esta actividad es:

- Especificación de requisitos de software.

Especificación de los requisitos:

En esta fase se documentan los requisitos acordados con el cliente, en un nivel apropiado de detalle. Estos requisitos deben ser escritos en un lenguaje natural.

Los artefactos que se generan son:

- Especificación de requisitos de software (actualizado).
- Especificación de historias de usuarios.
- Prototipos de interfaz de usuario.

Validación

La validación es la etapa final de la IR. Su objetivo es, ratificar los requisitos, es decir, verificar todos los requisitos para asegurarse que representan una descripción aceptable del sistema que se debe implementar. Esto implica verificar que los requisitos sean consistentes y que estén completos.
Durante esta actividad se aplican métricas de software como:

- Métricas de la calidad de la especificación de requisitos.
- Modelo de métricas orientadas a objeto aplicadas al diagrama de casos de uso.

Los artefactos que se generan son:

- Acta de validación de la especificación de casos de uso.
- Acta de validación de la especificación de requisitos.

1.7 Ambiente de desarrollo

1.7.1 Metodología de desarrollo de software

Para garantizar la correcta gestión del desarrollo de un sistema están definidas varias metodologías, con el objetivo de llevar un proceso guiado por pasos y procedimientos para lograr obtener clientes satisfechos y artefactos correctamente elaborados. Pero no solo se trata de seguir los pasos que proponga una metodología sino de seleccionar la adecuada, cada una de ellas está definida para proyectos con características específicas, por lo que se debe analizar bien cuál de las existentes va a resultar más factible emplear, pues de esto depende el éxito de un proyecto. Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las metodologías tradicionales y las ágiles. Las tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Pero adaptarse a la agitada sociedad actual implica ser “ágil” dándole mayor importancia a proveer respuestas rápidas y ser adaptables al cambio(Canós, Letelier y Penadés 2003).

Programación Extrema

La programación extrema (XP[6]) es un enfoque de la ingeniería del software formulado por (Beck 2000). Esta se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos.

[6] **XP:** Acrónimo en inglés de extreme Programming.

Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (Joskowicz 2008).
Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto y aplicarlo de manera dinámica durante el ciclo de vida del software. Sus fases son la planificación, el diseño, la codificación y las pruebas. Los artefactos que se generan con el uso de esta metodología son:

- Historias de usuarios
- Tareas de Ingeniería
- Pruebas de Aceptación
- Pruebas Unitarias y de integración
- Plan de la Entrega
- Código

Se selecciona para el desarrollo de la herramienta propuesta XP por ser una metodología ágil, diseñada para equipos de trabajo pequeños, centrada en vincular al cliente en el ciclo de desarrollo tratando de lograr un retroalimentación entre clientes y desarrolladores, incrementando la posibilidad de éxito, minimizando los riesgos de no conformidades y de obtener un producto final rechazado por el cliente por no cumplir con los objetivos y especificaciones trazadas. La metodología XP responde de manera eficiente a los cambios que se puedan presentar durante todo el desarrollo de la herramienta, proponiendo un ciclo de vida dinámico. El proceso de prueba de XP posibilita probar cada funcionalidad al finalizar cada iteración comprobando si cumple con los requisitos de la herramienta.

1.7.1 Herramienta CASE de modelado

Herramientas CASE[7] son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Desde el análisis de datos y procesos integrados mediante un repositorio hasta la generación de interfaces entre el análisis y el diseño, así como la generación del código y el mantenimiento y control(Quintero et al. 2012).

Visual Paradigm

[7] **CASE**: Acrónimo en inglés de Computer Aided Software Engineering

Visual Paradigm 8.0 ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de diagramas UML[8] (Rumbaugh, James, Jacobson, & Ivar, 2007). Constituye una herramienta privada disponible en varias ediciones, es multiplataforma, tiene un diseño centrado en casos de uso y enfocados al negocio para concebir un software de mayor calidad. Está diseñada para una amplia gama de usuarios, posee un enfoque orientado a objetos es fácil de instalar y usar. Fue diseñada para automatizar los procesos de software desde el análisis, así como el diseño y la implementación. Permite crear una base de datos a partir del esquema de clases y viceversa(Paradigm 2010).

1.8 Herramientas de desarrollo

Sistema gestor de Bases de Datos

Como sistema gestor de bases de datos se selecciona PostgreSQL ya que es un sistema distribuido bajo licencia BSD[9] y su código fuente está disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Entre las principales características por las que se elige son la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Además de que funciona bien con grandes cantidades de datos (Caldeira 2015).

PHP 5.3

Como lenguaje de programación del lado del servidor se escoge PHP[10] debido a que es un lenguaje de secuencia de comandos para la creación dinámica de documentos HTML[11]. Permite el desarrollo rápido de aplicaciones web, unido al hecho de que es un proyecto de código abierto y con una comunidad de desarrollo con amplia experiencia. Además ofrece múltiples funciones para la explotación de bases de datos lo que permite llevar a cabo proyectos web de gran envergadura. Se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy (Heurtel 2011).

JavaScript 1.8.5

Dentro de los lenguajes a utilizar del lado del cliente está JavaScript, es un lenguaje de script u orientado a documento, para añadir interactividad a las páginas web. Es compacto y basado en objetos,

[8] **UML:** Acrónimo en inglés de Unified Modeling Language.
[9] **BSD:** Acrónimo en inglés de Berkeley Software Distribution
[10] **PHP:** Acrónimo en inglés de HyperText Preprocessor.
[11] **HTML:** Acrónimo en inglés de Hyper Text Markup Language,

diseñado para el desarrollo de aplicaciones cliente-servidor (Eguíluz 2008). Por estas características se elige para el desarrollo de la herramienta y por las facilidades de ser interpretado e independiente de la plataforma, lo que permite incluir macros en páginas web, soporta varios tipos de datos y no es necesario declarar el tipo de las variables, este cambia implícitamente cuando es necesario, lo que ayuda a programar con rapidez macros sencillas.

HTML 5

Otro de los lenguajes del lado del cliente es HTML en su versión 5, el mismo es usado para describir la estructura y el contenido en forma de texto. Este puede describir hasta un cierto punto, la apariencia de un documento. Entre las características por las que es seleccionado está que facilita la creación de páginas web y no hay que compilar el código para ver si funciona. Se puede ver en forma inmediata el resultado del trabajo y también es usado para complementar el texto con objetos tales como imágenes (Eguíluz 2008).

CSS[12] 3

Se selecciona CSS en su versión 3 por ser un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML[13] y por extensión en XHTML[14]. Permite separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML(Eguíluz Pérez 2008). Las ventajas de utilizar CSS es que tienen un control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo. Por otro lado los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad ya que una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Facilitando que el documento HTML en sí mismo sea más claro de entender y se consiga reducir considerablemente su tamaño.

1.9 Marco de trabajo

Un marco de trabajo web se puede definir como un conjunto de componentes (por ejemplo: clases, descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web y es precisamente el que se utiliza en estos estudios. Entre los

[12] **CSS**: Acrónimo en inglés de Cascading Style Sheets
[13] **XML:** Acrónimo en inglés de Extensible Markup Language
[14] **XHTML:** Acrónimo en inglés de Extensible HyperText Markup Language

objetivos principales de un marco de trabajo se encuentran: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo, como el uso de patrones.

Symfony 2.3.7

Symfony es un marco de trabajo de PHP basado en la arquitectura MVC (Modelo-Vista-Controlador) y un proyecto de software libre. Su elección está condicionada por ser diseñado para optimizar el desarrollo de aplicaciones web, proporcionando herramientas para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto. Reutiliza conceptos y desarrollos exitosos de terceros y los integra como librerías para ser utilizados por otros desarrolladores. Permite la comunicación y control casi total de los datos sin importar si se está hablando de MySQL, PostgreSQL, SQL server, Oracle, entre otros (Eguiluz 2012). Además cuenta con el motor de plantillas Twig y Doctrine como ORM[15] por defecto.

ORM Doctrine 2

Como mapeador de objetos relacionales ORM que provee una persistencia transparente para los objetos PHP. Este usa el patrón de mapeo de datos, lo que proporciona una completa separación entre la lógica del dominio/negocio de la persistencia en un sistema de gestión de bases de datos relacionales. El beneficio de Doctrine para los programadores es la habilidad de concentrarse únicamente en la lógica de negocio orientada a objetos y no por la persistencia de los datos (Wage y Vesterinen 2009).

Twig 2.1

Twig es un motor de plantillas, integrado con Symfony2 para crear las plantillas de la aplicación, las cuales tienen una sintaxis concisa, fáciles de leer y de escribir (Porebski, Przystalski y Nowak 2011). Su característica más importante es que está implementado con herencia entre plantillas, lo que permite crear un “esqueleto” base que contiene todos los rasgos comunes de las interfaces ahorrando así código y tiempo en el trabajo. Se caracteriza por ser rápido, seguro y flexible:

Rápido: Compila las plantillas hasta código PHP regular optimizado. El costo general en comparación con código PHP regular se ha reducido al mínimo.

[15] **ORM**: Acrónimo en inglés de Object Relational Mapping.

Seguro: Tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Esto permite utilizarlo como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.

Flexible: Es alimentado por flexibles analizadores léxico y sintáctico. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados.

Twitter Bootstrap 2.1

Twitter Bootstrap es una colección de herramientas de software libre para la creación de sitios y aplicaciones web el cual es compatible con gran parte de los navegadores web. Bootstrap 2.1 es un marco de trabajo diseñado para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas CSS y de ficheros JavaScript, los cuales permiten obtener (Otto y Thornton 2013):

- ✓ Interfaces que funcionen de forma perfecta en los navegadores actuales y correctamente en los no tan actuales.
- ✓ Un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones.
- ✓ Una mejor integración con las bibliotecas que se suelen usar habitualmente, como por ejemplo jQuery.
- ✓ Un diseño sólido basado en herramientas actuales y potentes.

1.10 Servidor web

Un servidor web es un programa informático que procesa aplicaciones realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. Es quien gestiona la mayor parte de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores web son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Servidor HTTP Apache v2.2

Como servidor web se selecciona Apache ya que implementa el protocolo HTTP, es software libre y soporta los sistemas operativos Windows y Linux, es de fácil configuración, pues su estructuración en módulos permite al usuario utilizar los servicios y funcionalidades que ofrece (Apache 2011).

Se eligió Apache como software para el servidor de aplicaciones en su versión 2.2, por su flexibilidad, rapidez, porque es gratuito, multiplataforma e interpreta varios lenguajes como PHP, características

estas que promueven la eficiencia y rendimiento del sistema que debe ser capaz de dar respuestas a las peticiones con un nivel aceptable de desempeño, teniendo en cuenta la concurrencia que pueda existir, debe prestar servicios sin que se amplíen los rangos de tiempo de repuesta.

1.11 Entorno de desarrollo Integrado

Un entorno de desarrollo integrado, llamado también IDE[16] , es un programa informático compuesto por un conjunto de herramientas de programación. Existen varios entornos de desarrollo para el lenguaje PHP entre los más destacados se encuentran Eclipse y Netbeans, los cuales serán descritos a continuación.

NetBeans 7.3

Se selecciona como IDE Netbeans ya que es un entorno de desarrollo gratuito, multiplataforma y de código abierto que permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a tecnologías como Java, PHP, Groovy, C/C++, HTML5, entre otras. Entre sus principales características se encuentran que cuenta con un editor de código, multilenguaje muy potente, simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario, cuenta con una herramienta para la depuración de los errores, permite la conexión a diferentes gestores de bases de datos, se integra con diversos servidores de aplicaciones y es fácilmente extensible a través de plugins (Lim et al. 2008).

1.12 Patrones de diseño

El patrón es una descripción del problema y su solución que recibe un nombre y puede emplearse en otros contextos, ofreciendo sugerencias de cómo utilizarlos según el problema y ante situaciones nuevas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas (Tabares 2011).

Los patrones GRASP [17] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Tabares 2011) . De este grupo se utilizan en la investigación los siguientes patrones:

Experto: Plantea que la responsabilidad debe ser asignada al experto en la información, es decir a la clase que cuenta con la información necesaria para cumplir esta responsabilidad. Indica que la

[16] **IDE:** Acrónimo en inglés de integrated development environment.
[17] **GRASP:** Patrones Generales de Software para Asignación de Responsabilidades

responsabilidad de un objeto o la implementación de un método deben recaer sobre la clase que conoce toda la información necesaria para crearla.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos como el sistema que se desea implementar. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte también al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.
Alta Cohesión: Plantea asignar una responsabilidad de manera que la cohesión siga siendo alta. Permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión.

Bajo Acoplamiento: Está estrechamente relacionado con los patrones Experto y Alta cohesión, y plantea la baja dependencia que debe existir entre las clases.

Controlador: El patrón que sirve como intermediario entre las peticiones o acciones que se llevan a cabo dentro de la aplicación.

Los patrones GOF[18] que se emplean en este trabajo son (Guerrero, Suárez y Gutiérrez 2013):
Decorador: Este patrón posibilita extender la funcionalidad de un objeto dinámicamente de tal modo que sea transparente a sus clientes.

Inyección de dependencias: Patrón de diseño orientado a objetos en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto.

1.13 Arquitectura de software

La arquitectura del software es la estructura u organización de los componentes del programa (módulos), la manera en qué estos componentes interactúan, y la estructura de datos que utilizan (Pressman 2005). La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de: definir los módulos principales, las responsabilidades que tendrá cada uno de estos módulos y la interacción que existirá entre dichos módulos, constituye un artefacto de la actividad de diseño, que servirá de medio de comunicación entre los miembros del equipo de desarrollo, los clientes y usuarios finales, dado que contempla los aspectos

[18] **GOF**: Acrónimo en inglés de Gang of Four.

que interesan a cada uno. ¨ Una arquitectura de software es el producto del trabajo de desarrollo que ofrece el mayor rendimiento de la inversión con respecto a la calidad, el tiempo y el costo¨ (Bass 2007).

Se propone utilizar una arquitectura basada en capas (Capa de Presentación, la Capa de Negocio, la Capa de Acceso a Datos y la Capa de Datos), con el uso del patrón Modelo-Vista-Controlador en lo adelante (MVC). Su elección está condicionada porque es el patrón que utiliza el marco de trabajo Symfony y para cuando surjan problemas durante el desarrollo del sistema. EL patrón MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo y la escalabilidad de las aplicaciones (Espinosa et al. 2012).

El modelo: Mantiene el estado de la aplicación o sea es el responsable de la persistencia de los datos que esta maneja. Esta capa es la encargada de hacer cumplir todas las reglas del negocio que se le aplican a estos datos.

La vista: Genera las vistas de la aplicación generalmente muy estrechamente relacionados con los datos del modelo.

El controlador: Organiza la aplicación, es el intermediario de la comunicación entre la capa del modelo y la vista.

1.14 Métricas de Validación

Las métricas de software van a ayudar a la evaluación de los modelos de análisis y diseño, donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, ayudando a que diseño de pruebas se más efectivo (Pressman 1997).

Métricas de la calidad de la Especificación de requisitos:

Se emplea con el objetivo de determinar la especificidad de los requisitos. Esta se basa en la consistencia de la interpretación de los revisores para cada requisito. El equipo de revisores debe estar integrado por especialistas funcionales (no menos del 50%), analistas de sistemas y programadores (Sakipova 2011).

Métricas basadas en clases

Tamaño Operacional de las Clases (TOC): Está dada por el número de operaciones asignados a una clase, donde un resultado alto de su aplicación indica que la clase posee bastante responsabilidad, implicando un bajo nivel de reutilización de la clase y complicando la implementación y la realización de las pruebas. Los atributos de calidad que evalúa esta métrica son los siguientes (Grey y Viltres 2011):

- Responsabilidad: Un aumento del TOC está ligado a un incremento de la responsabilidad asignada a una clase.
- Complejidad de las pruebas: Un aumento del TOC indica un incremento de la complejidad de implementación de una clase y por tanto complejiza las pruebas.
- Reutilización: Un incremento del TOC implica una disminución del grado de reutilización de una clase.

Relaciones entre clases (RC): La métrica RC permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas por unidad para probar una clase teniendo en cuenta las relaciones existentes entre ellas.

1.15 Pruebas

La metodología XP divide las pruebas en dos grupos: las pruebas unitarias y las pruebas de aceptación. A continuación se describen ambas:

Pruebas unitarias

Las pruebas unitarias constituyen una de las piedras angulares de la metodología XP y se basan en realizar pruebas al código fuente del sistema. Dichas pruebas no se le pueden realizar a todo el código de la aplicación, debido a que el número de caminos lógicos puede crecer de forma exponencial, imposibilitando realizar casos de prueba para todos estos caminos, por tal motivo las pruebas unitarias se realizan solo a los principales requisitos o funcionalidades.

Una de las técnicas que utilizan las pruebas unitarias, es la técnica del camino básico. Esta técnica está basada en una mediada cuantitativa del software denominada complejidad ciclomática, que define el número de caminos independientes del programa, siendo éste una cuota superior para el número de casos de prueba que se debe realizar para asegurar la cobertura de sentencias. En este contexto dos caminos son diferentes si tienen sentencias o condiciones diferentes. La técnica propone, por tanto, generar casos de pruebas que ejecuten todos los caminos independientes (Pressman 2005). Esta será la técnica y la métrica utilizada para desarrollar los casos de pruebas unitarias a la herramienta desarrollada.

Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada (Joskowicz 2008).

1.16 Conclusiones del capítulo

Luego de la investigación realizada se pudo llegar a las conclusiones siguientes:

- Luego del estudio realizado a los diferentes métodos de selección multicriterio se eligió el método FAHP, ya que el mismo garantiza una mayor certeza en las evaluaciones emitidas por los expertos.
- Las herramientas existentes actualmente para realizar estudios de factibilidad no tratan de forma integrada la factibilidad técnica y comercial.

- La selección de la metodología, así como las herramientas, y lenguajes permitieron sentar las bases, desde el punto de vista técnico, para realizar un desarrollo de software eficiente.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En el presente capítulo se realiza una descripción del sistema a desarrollar y los roles que interactúan con el mismo. Además se describe detalladamente cómo será el desarrollo de la herramienta durante todas las fases que propone la metodología XP. Se generan todos los artefactos correspondientes a cada fase como es el caso de las historias de usuario y plan de iteraciones en la fase de planificación, las tarjetas clase responsabilidad colaborador (CRC) en la fase de diseño y finalmente las tareas de programación y la obtención del código funcional de la herramienta en la fase de codificación.

2.2 Objeto de Informatización

En la presente investigación se pretende obtener una herramienta que permita la informatización del proceso de estudio de factibilidad de un proyecto informático en la UCI, teniendo en cuenta criterios técnicos y comerciales, ayudando a los directivos de centros y gerentes de proyectos a tomar las decisiones en cuanto a que proyecto debe ser seleccionado para ser puesto en ejecución, lo que propicia que aumente el grado de certeza y el éxito de los proyectos de software antes de acometerlos. En la ilustración 2 se puede visualizar el proceso macro de cómo se lleva a cabo el método FAHP que es el que se propone para la realización de la herramienta. El mismo está condicionado por un conjunto de pasos que garantizan una vez terminados, que se realice una correcta selección de proyectos de software. Estos pasos son descritos a continuación.

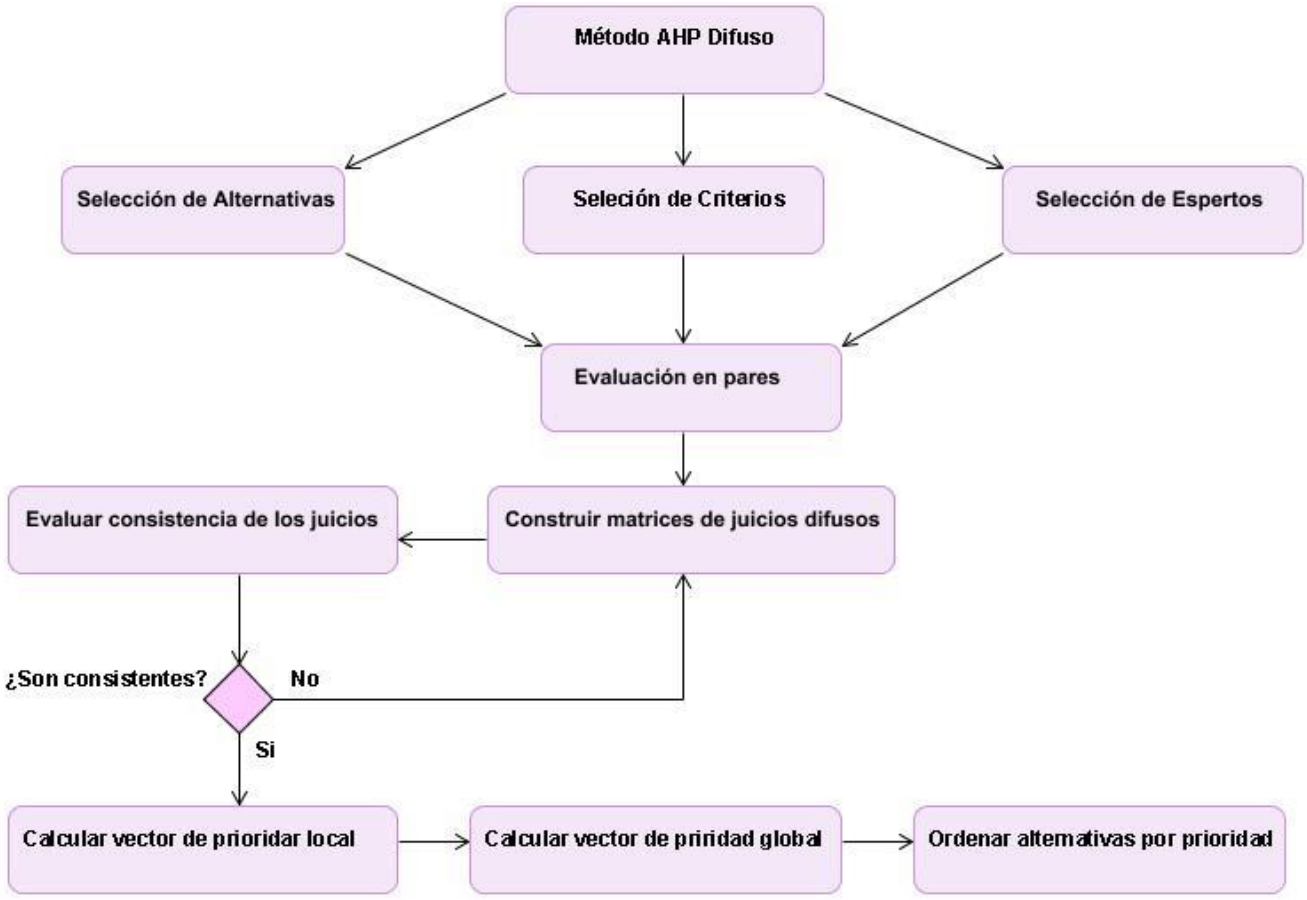

Ilustración 2. Proceso macro aplicando el método FAHP para la selección de proyectos.

Los principales pasos para el desarrollo del método son:

1. Desarrollo de la estructura jerárquica para los criterios y alternativas:

Los criterios de evaluación (Ver anexo #1) y las alternativas deben ser estructurados en diferentes niveles de jerarquía. Estos criterios y alternativas deben ser asignados a los expertos que deben haber sido seleccionados por los especialistas. Estos expertos antes de haber evaluado su nivel de conocimiento a tevés de las fuentes de argumentación (Ver anexo 3).

2. Representación difusa de los juicios, en una comparación por pares para criterios de evaluación y alternativas:

Una vez se construye la jerarquía, se debe hacer la conversión de la escala de Saaty en una escala de números triangulares difusos (Ver anexo #2).

3. Construcción de las matrices de juicio difuso para el AHP:

Con base en la jerarquía construida en el paso uno y a la escala difusa del paso dos se procede a la construcción de las matrices de juicio. La jerarquía de criterios y alternativas es el objeto de comparación por pares para el AHP.

4. Operaciones matemáticas:

Una vez se construyen las matrices de comparación por pares, se deben hacer los cálculos pertinentes para el desarrollo de la metodología, los cuales son (Cascales 2009):

- El cálculo del índice de consistencia

La consistencia se puede medir mediante el índice de consistencia (en lo adelante CI), que tiene la siguiente expresión:

$$CI = \frac{\lambda max - n}{n - 1}$$

Dónde:

n: Es el número de elementos que se comparan

λmax: Es el valor propio de la matriz, que se calcula de la siguiente forma:

$$\lambda max = \frac{\frac{1}{n}\sum_1^n AWi}{Wi}$$

Esta medida puede utilizarse para mejorar la consistencia de los juicios si se la compara con el número apropiado de la tabla que recoge el índice de consistencia aleatorio (en lo adelante IA) (Ver Anexo#3):

$$RC = \frac{IC}{IA}$$

- ✓ Si RC = 0, la matriz es consistente.
- ✓ Si RC ≤0,10, la matriz R tiene una inconsistencia admisible, lo que significa que se la considera consistente y el vector de pesos obtenidos se admite como válido.
- ✓ En caso de que RC > 0,10, la inconsistencia es inadmisible y se aconseja revisar los juicios.

- El cálculo de los vectores de peso para cada nivel de la jerarquía mediante el análisis extendido y los principios de comparación de números difusos, para su realización se llevan a cabo los siguientes pasos:

1- Se calcula el número difuso triangular Si para cada elemento (Ertuğrul y Karakaşoğlu 2009):

$$Si = \frac{\sum_{j=1}^n lij}{\sum_{j=1}^n \sum_{i=0}^n uij}, \frac{\sum_{j=1}^n mij}{\sum_{j=1}^n \sum_{i=1}^n mij}, \frac{\sum_{j=1}^n uij}{\sum_{j=1}^n \sum_{i=1}^n lij}$$

2- Se comparan los distintos números difusos triangulares Si dos a dos; el resultado se evalúa asignando los siguientes valores:

$$(S2 \geq S1) = \begin{cases} 1 \ si \ \ m2 \geq m1 \\ 0 \ \ si \ \ l1 \geq u2 \\ \dfrac{l1 - u2}{(m2 - u2) - (m1 - l1)} \ \ en \ cualquier \ otro \ caso \end{cases}$$

3- De todos los valores obtenidos de las comparaciones dos a dos de un elemento con los demás se elige el valor mínimo.

4- Se suman todos los valores mínimos y se asigna a cada elemento la normalización de su valor mínimo respecto de esa suma:

$$\frac{\min(Sn \geq Si)}{\sum_{j=1}^{n} \min(Sj \geq Si)}$$

5- Se repiten los pasos 1 al 5 para todos los niveles de una misma rama de la jerarquía.

6- Se evalúa cada alternativa con un valor que se corresponderá con el producto de todos los pesos de la rama de la jerarquía.

2.3 Roles relacionadas con el sistema

Las personas relacionadas con el sistema son aquellas que estás vinculadas al proceso de selección de proyectos, estos son:

Tabla 4 Roles relacionadas con el sistema.

Personas	Descripción
Administrador	Puede efectuar cualquier funcionalidad de sistema, tiene acceso a todo el proceso de selección.
Especialista	Es el encargado de hacer el proceso de selección y controla el desarrollo de todos los procesos que estén relacionados con él.
Experto	Se encarga de evaluar los proyectos atendiendo a los criterios que sean seleccionados por el especialista, para ello debe realizar una evaluación de competencia previa.

2.4 Ingeniería de requisitos

La ingeniería de requisitos es el conjunto de actividades implicadas en descubrir, documentar y mantener un conjunto de requisitos del software a desarrollar. La captura de los mismos es un proceso

en el cual los datos son extraídos a partir de la aplicación de técnicas de recopilación de información (Canós, Letelier y Penadés 2003).

2.4.1 Captura de requisitos

Para la captura de requisitos se utilizaron las entrevistas con el cliente, para conocer sus necesidades y la revisión documental para ver cómo se realizaba el método FAHP en otras investigaciones. Se identificaron un total de 44 requisitos funcionales y 15 requisitos no funcionales. Los requisitos funcionales identificados se clasificaron en tres grupos según su prioridad, fueron identificados 22 de prioridad muy alta, 15 de prioridad alta y 7 de prioridad media.

2.4.2 Definición de los requisitos

Los requisitos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares y como debe de comportarse ante tal situación (Escalona y Koch 2002). Es un conjunto de funcionalidad que permitirá informatizar el sistema que desea llevar a cabo. A continuación se muestra un listado de los requisitos:

Tabla 5: Requisitos funcionales.

Código	Descripción	Prioridad
RF-#1	Agregar usuario	Muy Alta
RF-#2	Eliminar usuario	Muy Alta
RF-#3	Modificar usuario	Muy Alta
RF-#4	Buscar usuario	Muy Alta
RF-#5	Agregar nomenclador	Media
RF-#6	Modificar nomenclador	Media
RF-#7	Eliminar nomenclador	Media
RF-#8	Crear Ficha de proyecto	Muy Alta
RF-#9	Modificar Ficha de proyecto	Muy Alta
RF-#10	Mostrar Ficha de proyecto	Muy Alta

RF-#11	Eliminar Ficha de proyecto	Muy Alta
RF-#12	Crear análisis de factibilidad	Muy Alta
RF-#13	Modificar análisis de factibilidad	Muy Alta
RF-#14	Eliminar análisis de factibilidad	Muy Alta
RF-#15	Buscar análisis de factibilidad	Muy Alta
RF-#16	Agregar criterio de evaluación	Muy Alta
RF-#17	Eliminar criterio de evaluación	Muy Alta
RF-#18	Mostrar criterio de evaluación	Muy Alta
RF-#19	Modificar criterio de evaluación	Muy Alta
RF-#20	Agregar expertos al proceso de evaluación	Muy Alta
RF-#21	Eliminar expertos al proceso de evaluación	Muy Alta
RF-#22	Buscar experto	Muy Alta
RF#23	Evaluar nivel de conocimiento de experto	Muy Alta
RF#24	Obtener el coeficiente de argumentación de cada experto	Muy Alta
RF#25	Calcular el coeficiente de competencia del experto	Muy Alta
RF-#26	Evaluar Método AHP para la valoración de criterio técnicos	Alta
RF-#27	Evaluar Método AHP para la valoración de criterio comercial	Alta
RF-#27.1	Seleccionar Objetivo	Alta
RF-#27.2	Seleccionar Criterios	Alta

RF-#27.3	Seleccionar Alternativas	Alta
RF-#27.4	Establecer prioridad entre criterios	Alta
RF-#27.5	Establecer prioridades locales	Alta
RF-#27.6	Establecer prioridades globales	Alta
RF-#27.8	Establecer prioridades locales entre alternativas	Alta
RF-#27.9	Representar los juicios de forma difusa	Alta
RF-#27.10	Verificar consistencia de los juicios	Alta
RF-#27.11	Construir las matrices de juicio difuso	Alta
RF-#27.12	Establecer prioridades totales asociadas a cada alternativa	Alta
RF#27.13	Defuzzificar	Alta
RF-#27.14	Ordenar alternativas por prioridad	Alta
RF-#28	Generar listado ordenado de proyecto	Media
RF-#29	Listar criterios técnicos y comerciales	Media
RF-#30	Listar especialistas	Media
RF-#31	Visualizar resultados	Media

Los requisitos no funcionales (RNF) se refieren a las características o cualidades que debe tener el sistema para que sea atractivo, usable, rápido y confiable. Para el desarrollo de la herramienta propuesta se definieron los RNF que se muestran a continuación, los que se clasificaron en RNF de apariencia o interfaz externa, usabilidad, fiabilidad, eficiencia, seguridad, software y restricciones de diseño e implementación.

Tabla 6: Requisitos no funcionales.

Código	**Clasificación**

	Apariencia o Interfaz Externa
RNF-#1	La interfaz de usuario en cada una de las páginas de la aplicación debe ser sencilla, intuitiva, que le permita al usuario que interactúa con el sistema realizar cualquier labor en el menor tiempo posible y a través de pocas acciones.
RNF-#2	Las interfaces contarán con menús que le permitan al usuario acceder desde cualquier lugar de la aplicación a otro lugar que desee con la menor cantidad posible de acciones.
RNF-#3	En el momento que ocurra un error con los datos de entrada a la aplicación le será informado al usuario de manera detallada.
	Usabilidad
RNF-#4	Los usuarios deben tener una navegación constante por lo que no pueden existir diferentes caminos para realizar una misma acción y los pasos para realizar la misma no deben ser complejos.
RNF-#5	El servidor de la aplicación deberá estar encendido para que esta siempre esté disponible, asegurando el acceso en todo momento desde cualquier lugar de red a todos los usuarios que estén autorizados.
	Fiabilidad
RNF-#6	Para asegurar la consistencia de los datos que se procesan en la aplicación y la fiabilidad de estos, se restringirá el nivel de acceso por roles y usuarios al gestor de base de datos y a la aplicación.
RNF-#7	En caso de fallos durante la ejecución de la aplicación el sistema tiene que ser capaz de recuperarse volviendo a su último estado.
	Eficiencia
RNF-#8	En caso que el sistema esté realizando mayor carga de trabajo por encontrarse procesando y almacenando datos, los tiempos de respuesta no deben exceder los 5 segundos.

	Seguridad
RNF-#9	La aplicación debe estar protegida mediante una política de usuarios y roles que no permitan el acceso no autorizado a esta. De esta forma se asegura la integridad y confiabilidad de los datos que en esta se procesan. Configurándose en el marco de trabajo los roles definido para cada usuario y la encriptación de las contraseñas.
RNF-#10	A cada usuario se le debe mostrar y dar acceso a las funcionalidades a las cuales está autorizado a acceder. Asegurando el acceso a los datos que corresponde para cada usuario en el sistema. En el marco de trabajo se debe configurar mediante el uso de las ACL para el acceso a las rutas de acuerdo a lo mencionado anteriormente.
	Software
RNF-#11	En el servidor de aplicación para el correcto funcionamiento de la aplicación es necesario tener instalado alguna distribución del Sistema Operativo GNU/Linux o Windows, el servidor web Apache en su versión 2.2, Marco de trabajo Symfony2.
RNF-#12	El servidor de datos, debe estar instalado el Gestor de Base de Datos PostgreSQL 9.1 o superior.
RNF-#13	Para el uso en las PC clientes es necesario tener instalado un navegador Web (Mozilla Firefox v.26.x, Google Chrome v.30.x Opera v.20.x, Internet Explorer v.7.x)
	Hardware
RNF#14	Para garantizar un buen desempeño de la aplicación en los servidores se necesita una computadora con 2 GB de memoria RAM o superior, un disco duro 320 GB o superior por el volumen de datos que se generan, un procesador Dual Core 2.0 GHz o superior.
RNF#15	En las PC clientes se necesitan para garantizar un funcionamiento correcto 1 GB de memoria RAM mínimo, un procesador Pentium 3 o superior.

2.5 Fase de planificación

Para el desarrollo de la herramienta se dividirá el trabajo en cuatro fases como lo propone la metodología XP.

En la fase de planificación el cliente y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuarios, para obtener una descripción detallada de las funcionalidades que realiza el sistema, y asociadas a estas, las entregas. Una vez obtenidas las historias de usuarios, se evalúa el tiempo de desarrollo de cada una a través de un plan de iteraciones.

2.5.1 Historias de usuarios

Durante la fase de planificación se describen historias de usuario, las mismas son escritas por el cliente en un lenguaje no técnico, con descripciones cortas del sistema sin hacer mucho hincapié en los detalles. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. A continuación se describe la historia de usuario Gestionar criterio del sistema a desarrollar, las otras se encuentran en el artefacto que contiene las historias de usuario.

Tabla 7: Descripción de la historia de usuario HU#5.

Historia de Usuario	
Número: *HU_5*	**Nombre Historia de Usuario**: Gestionar criterio
Modificación de Historia de Usuario Número: Ninguna.	
Programador: Isis Bertamí Barrios	**Iteración Asignada**: primera
Prioridad en Negocio: Muy Alta.	**Puntos Estimados**: 3 días
Riesgo del Desarrollo: Alto.	**Puntos Reales**: 3 días
Descripción: La funcionalidad gestionar criterio, debe permitir agregar, mostrar, modificar y eliminar un criterio de evaluación del sistema. Inicialmente se debe mostrar una página donde aparezcan todos los criterios que han sido adicionados con anterioridad en una tabla, con la información del nombre, tipo de criterio y las opciones para escoger las acciones que se puedan realizar sobre estos criterios. **Agregar:** Se debe mostrar una página para solicitar todos los datos necesarios de un criterio **Modificar**: Cuando se selecciona el criterio que se desea modificar, se muestra la información que este contiene, permitiendo realizar cambios en este y finalmente permitiendo actualizarlo.	

Mostrar: Se deben mostrar los criterios existentes hasta el momento. **Eliminar:** Se elimina el criterio seleccionado, siempre preguntando la confirmación de la acción a realizar al usuario para evitar pérdidas de datos accidentales.
Observaciones: Las acciones gestionar criterios solo pueden ser realizadas por un usuario que tenga privilegios de especialista en el sistema.

2.5.2 Plan de iteraciones

Luego de identificar las historias de usuarios y establecer la prioridad entre ellas se prosigue a confeccionar el plan de iteraciones. Este plan define que historias de usuarios serán implementadas en cada iteración y que tiempo se estima que durará cada una de ellas. El sistema estará dividido en tres iteraciones como se muestra a continuación:

Primera: Se implementan las historias de usuario que tienen una prioridad muy alta para el negocio.

Segunda: Se implementan las historias de usuario que tienen una prioridad alta para el negocio.

Tercera: Se implementan las historias de usuario que tienen una prioridad media para el negocio.

Tabla 8: Plan de iteraciones.

Iteración	Orden de la HU a implementar	Duración de HU (días)	Duración Total (semanas)
1ra	Gestionar Usuarios	7	6
	Gestionar Fichas de proyectos	7	
	Gestionar ficha para solicitar información técnica y comercial	7	
	Gestionar análisis de factibilidad	7	
	Gestionar criterios a evaluar	3	
	Realizar evaluación de las competencias de un experto	7	

2da	Evaluar método FAHP para la valoración de criterios técnicos y comerciales	15	**2**
3ra	Generar listado ordenado de proyecto	7	**2**
	Gestionar nomencladores	7	
Total			**10**

2.5 Fase de Diseño

Durante la fase de diseño se confeccionan las tarjetas clase responsabilidad colaborador para la descripción de cada una de las entidades, se realiza el diagrama de clases para tener mejor visión de la estructura del sistema y se diseña el modelo de datos.

2.5.1 Tarjetas clase-responsabilidad-colaborador (CRC)

El uso de las tarjetas C.R.C (Class, Responsabilities and Collaboration) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. El uso de esta técnica de modelado permite identificar las clases, los atributos y responsabilidades. A continuación se muestran algunas de las tarjetas del sistema a desarrollar.

Tabla 9: Tarjeta CRC "Criterio".

Criterio	
Descripción: Guarda información de los criterios que serán utilizados durante el proceso de evaluación.	
Atributos	
Nombre	**Descripción**
id	Campo identificador
nombre_criterio	
tipo_criterio	

Tabla 10: Tarjeta CRC "Valor_Criterio".

Valor _Criterio	
Descripción: Guarda información del valor del criterio proporcionado por el experto que realiza el análisis	
Atributos	
Nombre	**Descripción**
id_valor_criterio	Campo identificador
valor_l	
valor_m	

valor_criterio	
descripcion	
Responsabilidades	
Nombre	**Colaborador**
Crear	
Modificar	
Mostrar	
Eliminar	

valor_u	
Responsabilidades	
Nombre	**Colaborador**
Agregar	
Modificar	
Eliminar	

2.5.2 Patrones de diseño

En el diseño del sistema es importante asignar correctamente las responsabilidades, para hacerlo de forma eficaz se utilizan los patrones GRASP, que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El marco de trabajo Symfony utiliza varios patrones, situándolos en las capas de Modelo y Control que plantea el patrón arquitectónico Modelo Vista Controlador. A continuación son descritos los principales patrones que se utilizaron en el desarrollo de la solución.

Experto: En la solución esto se puede evidenciar en las entidades las cuales contienen los datos que las caracterizan y en ellas mismas se implementan los métodos para poder acceder o modificarlos.

Creador: Al aplicar este patrón en la solución es posible precisar las responsabilidades entre las clases del negocio y cuál tiene la responsabilidad de instanciar nuevos objetos y clases, además de definir la visibilidad que existe entre ellas. A continuación se pone un ejemplo de la clase controladora criterioController.

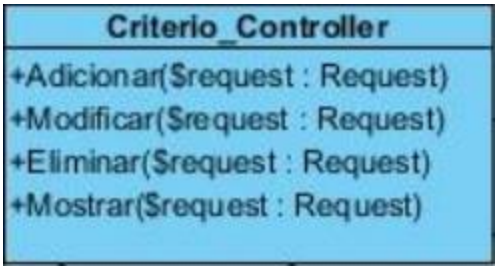

Ilustración 3. Patrón Creador.

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

Bajo Acoplamiento: Su aplicación se puede evidenciar en la separación de las clases según la arquitectura MVC, las clases que implementan la lógica del negocio y de acceso a datos se encuentran separadas y no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja. Además mediante el empleo de *bundles* se propicia un bajo acoplamiento y reutilización de los componentes de la aplicación.

Controlador: En Symfony todas las peticiones Web son manipuladas por un solo controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado y la existencia de controladores adicionales que actúan como intermediarios entre el modelo y la vista. En la aplicación es utilizado a través del controlador frontal **app.php**, siendo el encargado de manejar todas las peticiones web. El controlador app.php es el punto de entrada único de toda la aplicación.

Otro conjunto de patrones que son utilizados por el marco de trabajo Symfony son los llamados patrones GOF, los mismos son descritos a continuación.

Decorador: En Symfony existe el método decorador que pertenece al motor de plantitas Twig, que contienen un decorador para permitir agregar funcionalidades dinámicamente a las vistas. Se utiliza principalmente en el diseño o definición de las plantillas o *layout* de las cuales pueden heredar las demás vistas de la aplicación.

Inyección de dependencias: Se implementa por defecto en las aplicaciones desarrolladas en el marco de trabajo Symfony2. Su uso se pone en práctica en la llamada que se realiza desde las clases controladoras a los objetos o componentes de la solución desarrollada.

2.5.3 Arquitectura utilizada

La arquitectura a la que responde la herramienta, es una arquitectura en capas que implementa el patrón MVC y posee la característica de tener varios complementos transversales a las capas para garantizar seguridad, tratamiento de excepciones, entre otros aspectos. Cada uno de los módulos o bundles se estructuran arquitectónicamente igual. Las entidades del dominio son accesibles en la sub-

capa servidor de la Capa de Presentación, la Capa de Negocio y la Capa de Acceso a Datos. En la siguiente ilustración se evidencia como se estructuran cada una de estas capas.

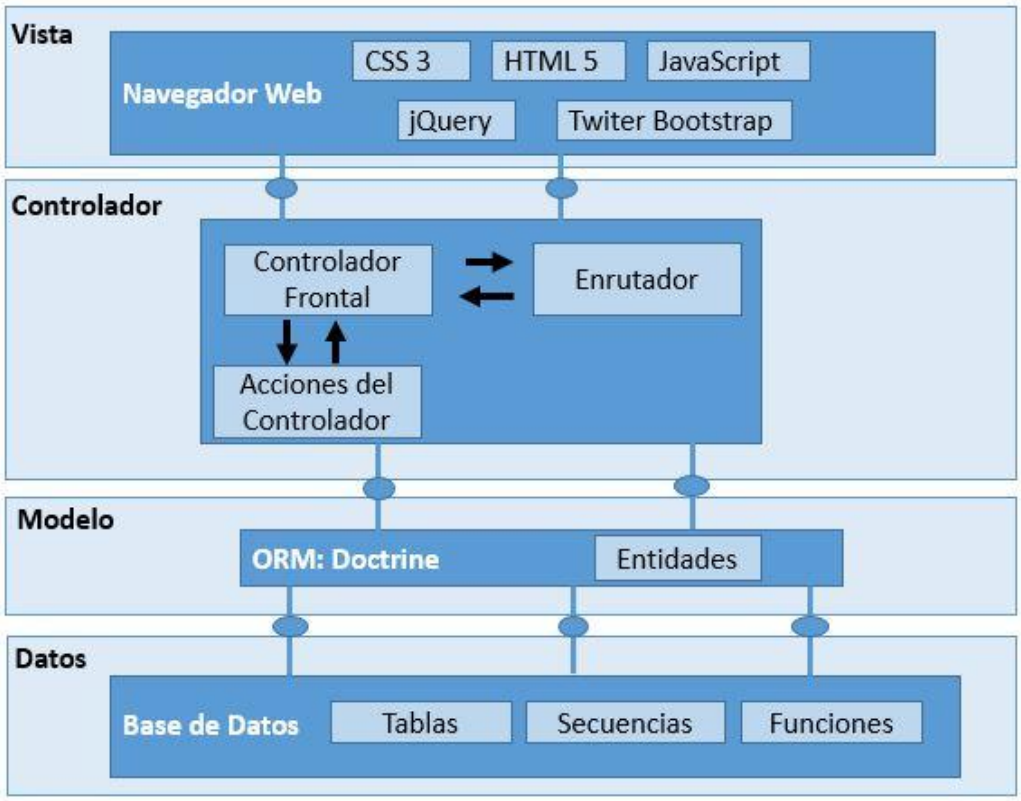

Ilustración 4. Arquitectura en capas.

Capa de Presentación: contiene los componentes con los que el usuario va a interactuar (páginas). Permite alcanzar las funcionalidades que brinda el controlador y mostrar o capturar la información a través de los diferentes elementos que comprende, es decir: twigs, formularios, entre otros.

Capa de Negocio: Su diseño depende directamente del negocio específico al que se refiera cada sistema. Esta capa recibe una petición del nivel superior y gestiona o procesa la misma. Finalmente envía una respuesta continuando el proceso en el punto donde se inició dicha petición. Contiene las clases gestoras, encargadas del manejo de la lógica de negocio.

Capa de Acceso a datos: Contiene las entidades y repositorios que mapea Doctrine como marco de trabajo para la comunicación con el servidor de datos. Gestiona las peticiones de la capa de negocio consultando la BD. Es importante destacar que las entidades del dominio se encuentran en esta capa pero son accesibles además desde las capas superiores.

Capa de datos: En esta capa se encuentra el gestor de base de datos PostgreSQL y en él un conjunto de esquemas, tablas, vistas y procedimientos almacenados que permiten persistir la información con la que trabaja la aplicación y manejar su almacenamiento.

2.5.3 Diagrama Entidad a partir de las tarjetas CRC

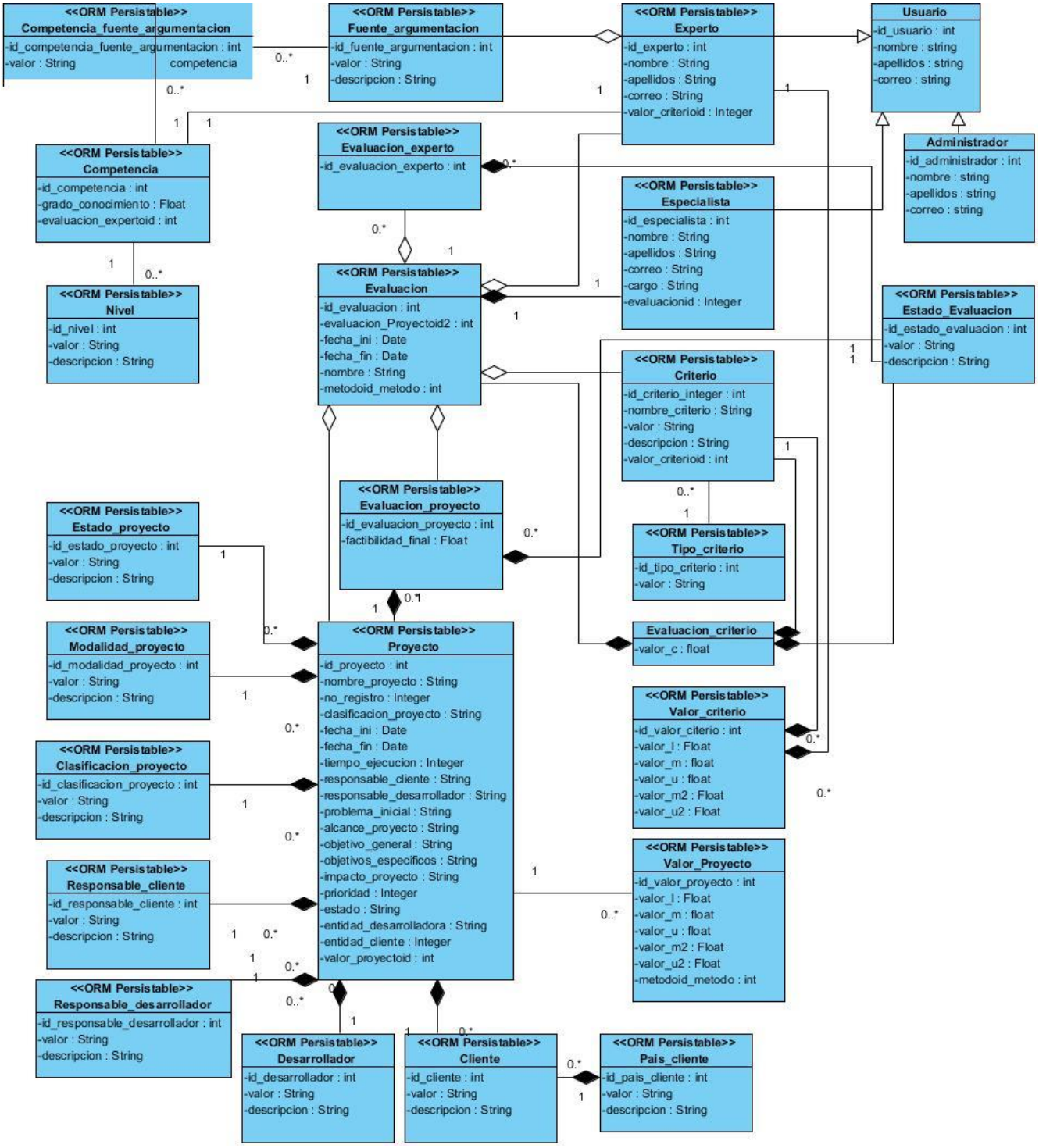

Ilustración 5: Diagrama de clases persistentes.

2.5.5 Modelo de Datos

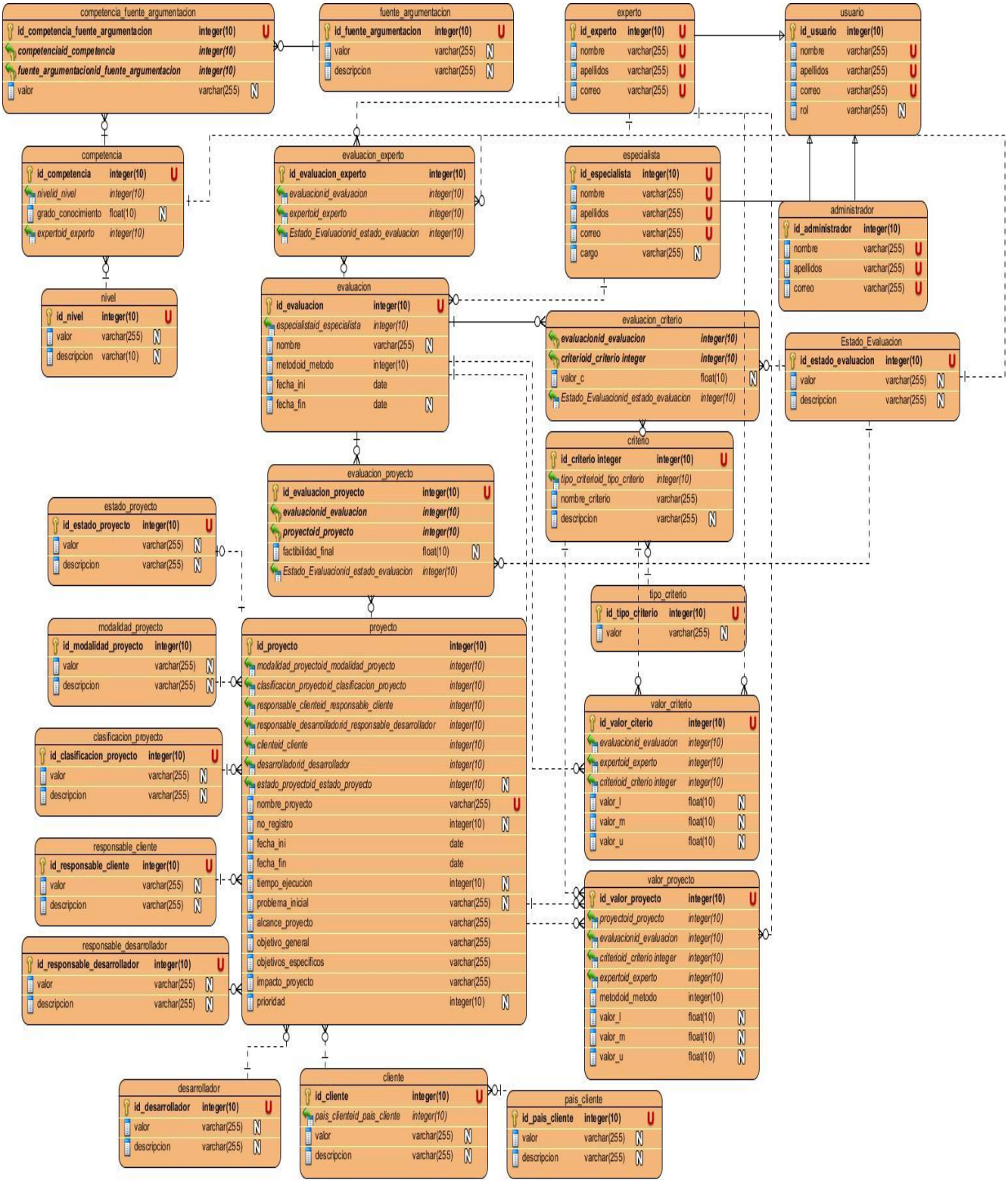

Ilustración 6: Modelo de Datos.

Patrones de diseño de base de datos:

Llaves subrogadas: El uso de este patrón es bastante generalizado en la base de datos pues la mayoría de las entidades contienen una llave única entera autoincremental, lo que facilita reducir el costo de las búsquedas en la base de datos. EL uso de este patrón constituye una protección ante los cambios debido a que la lógica del negocio no está en las llaves y evita la contención (bloqueo) de la base de datos, debido a la rapidez de los mecanismos de generación que provee el sistema.

Control de Acceso Basado en Roles (RBAC, por sus siglas en inglés): este patrón se implementa mediante la asignación de roles a los diferentes usuarios de la aplicación. Cada rol posee permisos para realizar determinadas funciones de acuerdo a las restricciones del negocio. Por lo que cada usuario tiene estrictamente los permisos que les son conferidos a través del rol que desempeña.

2.5.4 Diagrama de despliegue

El diagrama de despliegue modela la distribución en tiempo de ejecución de los elementos de procesamiento y componentes de software, junto a los procesos y objetos asociados. En dicho diagrama se modelan los nodos y la comunicación entre ellos y cada nodo puede contener instancias de componentes(Sarmiento).

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. El siguiente diagrama de despliegue tiene la intención de describir cómo va a ser desplegada la aplicación

Ilustración 7: Diagrama de Despliegue.

2.6 Fase de codificación

Durante la fase de codificación se planifican y ejecutan las tareas de programación, además de que se codifica el sistema para obtener como resultado la herramienta .La codificación debe hacerse ateniendo a estándares de codificación ya creados para mantener el código consistente y facilitar su comprensión y escalabilidad.

2.6.1 Tareas de programación

Las tareas de programación son actividades que se derivan de las historias de usuarios y son divididas entre los miembros del equipo de desarrollo para simplificar su implementación, logrando una programación en pareja. Cada una de estas tareas podrá ser comprobada a través de los casos de prueba.

A continuación se relacionan las tareas a desarrollar para la implementación de la herramienta, y la descripción de alguna de ellas.

Tabla 11: Tareas de programación por cada historia de usuario.

Historia de Usuario	Tareas de programación
Gestionar usuarios	• Agregar usuarios • Buscar usuarios • Eliminar usuarios
Gestionar nomencladores	• Agregar cada uno de los elementos que pueden ser listados en nomencladores • Modificar cada uno de los elementos que pueden ser listados en nomencladores • Eliminar cada uno de los elementos que pueden ser listados en nomencladores
Gestionar fichas de proyectos	• Crear ficha de proyecto • Mostrar ficha de proyecto • Modificar ficha de proyecto • Eliminar ficha de proyecto
Gestionar análisis de factibilidad	• Crear análisis de factibilidad • Buscar análisis de factibilidad • Eliminar análisis de factibilidad • Enviar alertas
Gestionar criterios a evaluar durante una evaluación	• Agregar criterio • Mostrar criterio • Modificar criterio • Eliminar criterio

Realizar evaluación de las competencias de un experto	• Crear evaluación de las competencias de un experto • Estudiar métodos para evaluar expertos • Desarrollar los métodos para evaluar los expertos
Evaluar método AHP para la valoración de criterios técnicos y comerciales	• Mostrar Interfaz para seleccionar criterios, expertos, metas y alternativas. • Mostrar Interfaz para ponderar criterios y alternativas. • Representar los juicios de forma difusa • Construir matriz de juicio difuso • Realizar Desfuzzificación • Procesar datos de las ponderaciones
Generar listado ordenado de proyectos	• Generar listado • Exportar listado

Tareas para HU#5: Gestionar criterios a evaluar durante una evaluación

Tabla 12: Descripción de la Tarea de Programación HU#5-1.

Tarea	
Número de Tarea: HU#5-1	**Historia de Usuario:** HU#5- Gestionar criterios a evaluar durante una evaluación
Nombre de Tarea: Agregar criterio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 3/04/2015	**Fecha Fin:** 8/04/2015
Programador Responsable: Isis Bertamí y Bárbara Salinas	
Descripción: Se debe mostrar una página para solicitar todos los datos necesarios de un criterio.	

Tabla 13: Descripción de la Tarea de Programación HU#5-2.

Tarea	
Número de Tarea: HU#5-2	**Historia de Usuario:**

	HU#5- Gestionar criterios a evaluar durante una evaluación
Nombre de Tarea: Mostrar criterio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 3/04/2015	**Fecha Fin:** 8/04/2015
Programador Responsable: Isis Bertamí y Bárbara Salinas	
Descripción: Cuando se selecciona la opción mostrar criterio, se muestran los criterios existentes hasta el momento.	

Tabla 14: Descripción de la Tarea de Programación HU#5-3.

Tarea	
Número de Tarea: HU#5-3	**Historia de Usuario:** HU#5- Gestionar criterios a evaluar durante una evaluación
Nombre de Tarea: Modificar criterio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 3/04/2015	**Fecha Fin:** 8/04/2015
Programador Responsable: Isis Bertamí y Bárbara Salinas	
Descripción: Cuando se selecciona el criterio que se desea modificar, se muestra la información que este contiene, permitiendo realizar cambios en este y finalmente permitiendo actualizarlo	

Tabla 15: Descripción de la Tarea de Programación HU#5-4.

Tarea	
Número de Tarea: HU#5-4	**Historia de Usuario:** HU#5- Gestionar criterios a evaluar durante una evaluación

Nombre de Tarea: Eliminar criterio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 3/04/2015	**Fecha Fin:** 8/04/2015
Programador Responsable: Isis Bertamí y Bárbara Salinas	
Descripción: Se elimina el criterio seleccionado, siempre preguntando la confirmación de la acción a realizar al usuario para evitar pérdidas de datos accidentales.	

2.6.2 Estándares de codificación

Con el objetivo de lograr una estandarización en la programación de la aplicación web se decide aplicar el estándar de codificación CamelCase, específicamente la variante lowerCamelCase. El cual facilitará la lectura, comprensión y mantenimiento del código.

A continuación se describen a grandes rasgos las convenciones de nomenclatura.

General:

- ✓ Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- ✓ En todo momento se utilizarán nombres para las columnas de las tablas que sean claros, concretos, libres de ambigüedades y expresen su significado. Ejemplo: "id_proyecto" y no solamente "id" para hacer referencia al id correspondiente a la tabla proyecto.
- ✓ El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula siguiendo el patrón lowerCamelCase. Ejemplo: “buscarCriterioTecnico()”.

Identación:

- ✓ El contenido siempre se identará con tabs, nunca utilizando espacios en blanco.

Clases:

- ✓ El nombre de las clases comenzará con mayúscula si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula también, siguiendo el patrón lowerCamelCase. Ejemplo: “criterioController”.
- ✓ Mantener los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML.

2.7 Conclusiones del capítulo

En el presente capítulo se definió la propuesta de solución mediante el uso de la metodología seleccionada, propiciando la generación de cada uno de los artefactos, lo que posibilitó la organización del trabajo durante el proceso de desarrollo.

En la fase de planificación se realizó la captura de requisitos mediante técnicas propuestas por la ingeniería de requisitos y estos fueron descritos mediante las historias de usuario, de las que se elaboró el plan de iteraciones para definir en qué momento iban a ser implementadas y cuánto tiempo iba a demorar la realización del sistema.

Se realizó una descripción de la fase del diseño donde se elaboraron las tarjetas CRC a partir del modelo de datos, y análisis de los patrones de diseño y arquitectónicos de acuerdo al marco de trabajo Symfony que fue el seleccionado para el desarrollo de la herramienta.

Durante la fase de codificación se elaboraron las tareas de codificación derivadas de las historias de usuario y se definieron estándares de codificación para lograr que el código esté organizado y legible.

CAPÍTULO 3: VALIDACIÓN Y PRUEBA DE LA SOLUCIÓN

3.1 Introducción

En el presente capítulo se describe la validación de la herramienta desarrollada con el objetivo de evaluar los resultados obtenidos a partir del diseño y la implementación. Esto se lleva a cabo mediante la utilización de diferentes métricas, como las que miden la calidad de la especificación de requisitos, las que corresponden al tamaño operacional de las clases y las relaciones entre ellas. Además se aplicaron las pruebas de software unitarias y de aceptación con el objetivo de que los requisitos especificados se realicen adecuadamente y alcanzar la aceptación del cliente.

3.2 Validación mediante métricas

La aplicación de estas métricas durante el análisis y el diseño proporcionan una forma cuantitativa de medir los atributos de calidad internos del software, para expresar una valoración sobre la calidad de la solución. A continuación se expone el resultado de la aplicación de estas métricas.

3.2.1 Métricas de la calidad de la Especificación de requisitos:

Una vez identificados los requisitos se realizó una comprobación para identificar especificidad basada en la consistencia de la interpretación del equipo de revisores, constituido por los dos miembros del equipo de desarrollo y el cliente.

Para realizar este proceso se determina el toral de requisitos mediante la suma de los requisitos funcionales y no funcionales identificados.

Ilustración 8 Fórmula para requisitos funcionales.

Al aplicar la fórmula se obtiene:

NR = RF + RNF

NR = 44 + 15

NR = 59

Para determinar, finalmente, la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación: $Er = Nu/Nr$

Donde Nu es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad. Para el caso de los requisitos obtenidos para la aplicación solo 3 produjeron contradicción en las interpretaciones. Sustituyendo las variables se obtiene:

ER = Nu / Nr

ER = 56 / 59

ER = 0,95

El grado de ambigüedad fue bajo y los requisitos ambiguos fueron verificados para una correcta comprensión.

3.2.2 Métricas basadas en clases

Tamaño Operacional de las Clases (TOC): Esta métrica permite medir el tamaño general de una clase tomando el valor de la cantidad de operaciones y el número de atributos que están encapsulados dentro de dicha clase. Es importante destacar que en esta métrica, la responsabilidad y la complejidad son inversamente proporcional a la reutilización, por tanto, a mayor responsabilidad y complejidad de una clase, menor será su nivel de reutilización.

A continuación se muestra el resultado de la métrica TOC a 12 clases que juegan un papel importante en el sistema.

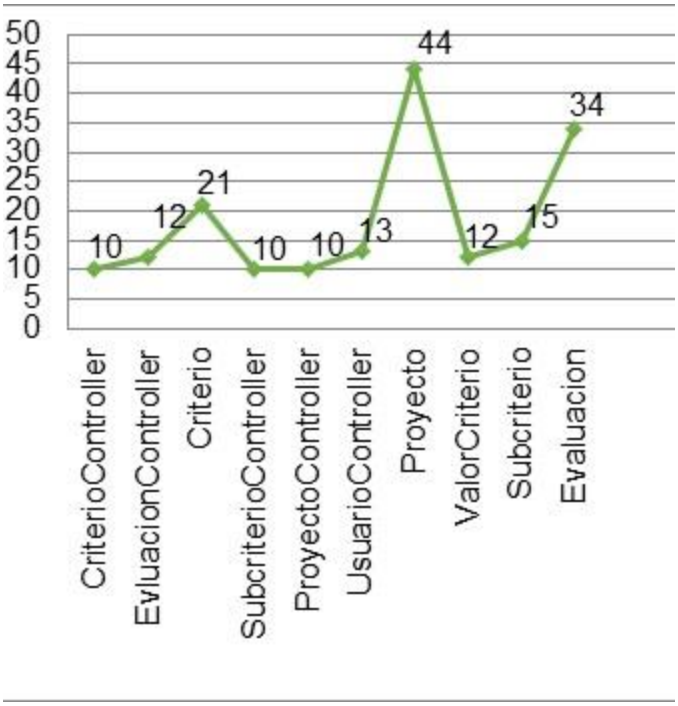

Ilustración 9 Cantidad de procedimientos por clases.

Aplicando como umbral un promedio de 15 procedimientos, se obtienen los valores de los atributos de calidad: responsabilidad, complejidad y reutilización.

Tabla 16. Valores de las variables de calidad: Responsabilidad, Complejidad y Reutilización.

	Responsabilidad		**Complejidad**		**Reutilización**	
Nivel	Cantidad de clases	Promedio	Cantidad de clases	Promedio	Cantidad de clases	Promedio
Baja	3	30	3	30	2	16
Media	4	40	4	40	6	60
Alta	3	30	3	30	8	68

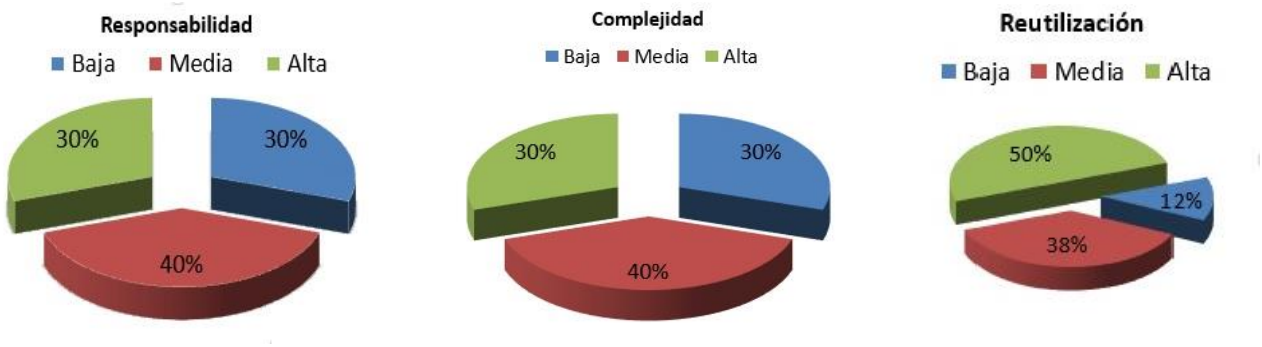

Ilustración 10. Resultados de las variables: Responsabilidad, Complejidad, Reutilización.

Luego de aplicada la métrica TOC, se puede observar en la ilustración 10 que en las clases la responsabilidad y la complejidad se encuentran en un nivel medio, lo que favorece que las mismas puedan ser reutilizadas.

Relaciones entre clases (RC): Luego de aplicar la métrica RC, se arrojaron los siguientes resultados.

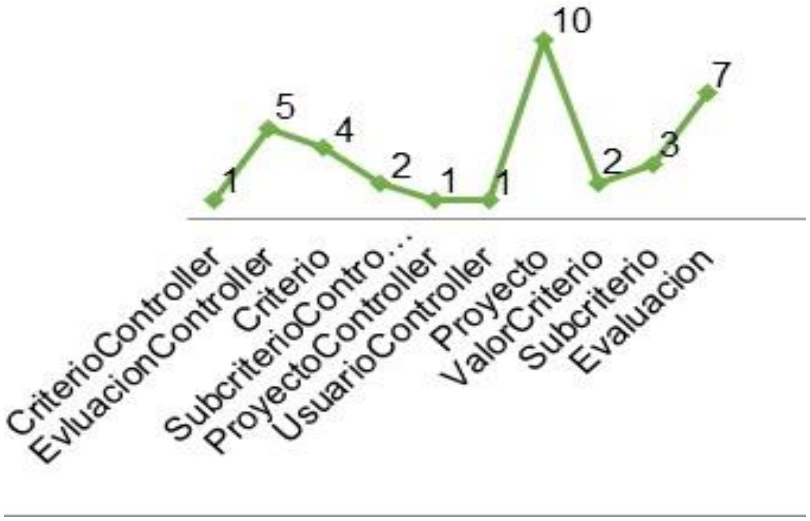

Ilustración 11. Cantidad de relaciones de usos por clases.

Luego de calcular los valores de las variables de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas, aplicando como umbral un promedio, mayor a 3 asociaciones de uso por clase, obteniendo los siguientes resultados.

Tabla 17. Valores para las variables: Acoplamiento, Complejidad, Reutilización, Cantidad de Pruebas.

	Acoplamiento	Complejidad	Reutilización	Cantidad de Pruebas

Nivel	Cantidad de clases	Promedio	Cantidad de clases	Promedio	Cantidad de clases	Promedio	Cantidad de clases	Promedio
Ninguno	0	0	-	-	-	-	-	-
Baja	5	50	2	20	6	60	2	20
Media	1	10	2	20	2	20	2	20
Alta	4	40	6	60	2	20	6	60

Una vez aplicada la métrica RC y teniendo en cuenta el umbral definido para validar el diseño, se obtiene como resultado que en las cases existe un predominio de un bajo acoplamiento. Además el indicador cantidad de pruebas fue bajo evidenciando que el número o grado de esfuerzo para realizar las pruebas de calidad va a ser bajo.

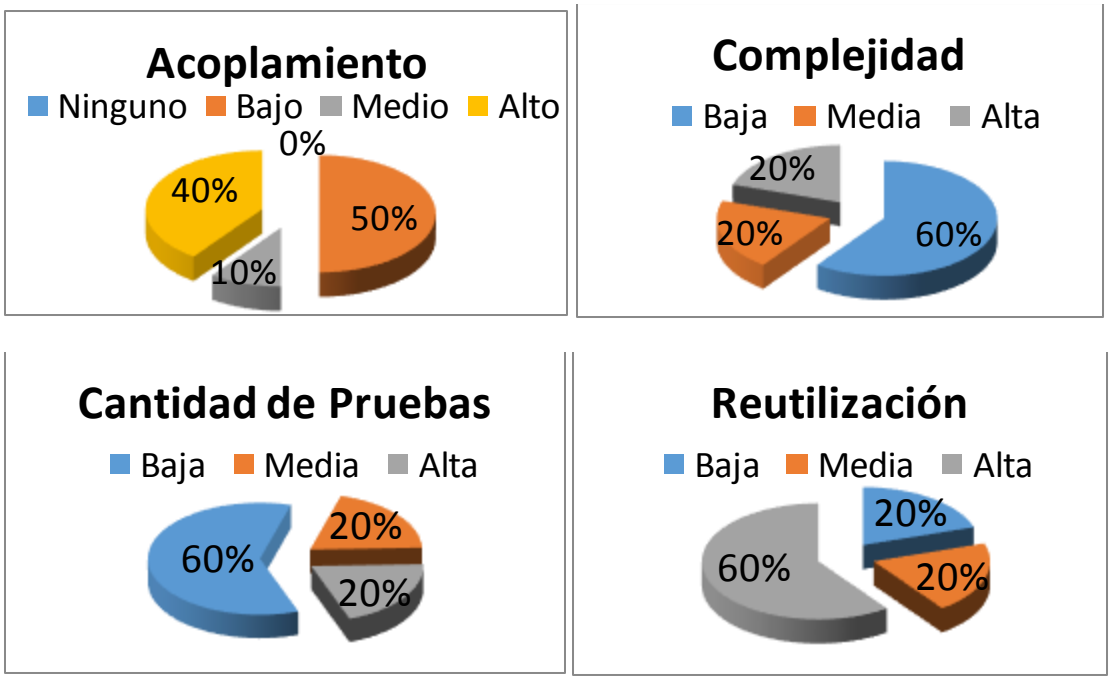

Ilustración 12. Resultado de las variables: Acoplamiento, Complejidad y Cantidad de pruebas.

3.3 Fase de Pruebas

Las pruebas del sistema tienen como objetivo verificar las funcionalidades del sistema a través de sus interfaces externas comprobando que dichas funcionalidades sean las esperadas en función de los requisitos del sistema (Abran et al. 2001). Uno de los pilares de la metodología XP es el proceso de

pruebas debido a que se realiza de forma continua y tanto como sea posible, esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.
XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y estas son diseñadas por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida, estas generalmente son diseñadas por el usuario o cliente final.

3.3.1 Pruebas unitarias

Para realizar las pruebas unitarias al código fuente de la herramienta se selecciona el algoritmo calcularC() que es el más complejo del sistema, el mismo se encarga de calcular la prioridad de las alternativas evaluadas. A continuación se explica todo el procedimiento de la obtención de los casos de prueba, utilizando la técnica de camino básico con la métrica complejidad ciclomática. Para esto se hará uso de una notación para la representación de flujo de control. En la siguiente ilustración se muestra el código fuente del algoritmo.

```
BEGIN
for criterio_id in select unnest( criterio_array )loop    //1
select into aux * from public."calcularS"("criterio",evaluacion_id,criterio_id);    //2
s_array[i] := aux;    //2
i:=i+1;    //2
end loop;
    for s_val in select unnest (s_array) loop    //3
    arreglo_s:=unnest(string_to_array(s_val, ','));    //4
    i:=1;    //4
          for sval_v in select unnest (s_array) loop    //5
          arreglo_sv:=unnest(string_to_array(sval_v, ','));    //6
                  if arreglo_s[2]>= arreglo_sv[2] then    //7
                  arreglo_comp[i]:=1;    //8
                  elsif  arreglo_sv[1]>=arreglo_s[3] then    //9
                  arreglo_comp[i]:=0;    //10
                  else    //11
                  arreglo_comp[i]:=(arreglo_sv[1]-arreglo_s[3])/((arreglo_s[2]-arreglo_s[3])-(arreglo_sv[2]-arreglo_sv[1]));    //12
                  end if;
          i:=i+1;    //13
          end loop;


       SELECT into aux1 min(valor_c) FROM unnest(arreglo_comp) AS valor_c;    //14
       arreglo_c[pos]:=aux1;    //14
       pos:=pos+1;    //14
    end loop;

       SELECT into sum_c sum(valor_c) FROM unnest(arreglo_c) AS valor_c;    //15
       i:=1;    //15
    for valor_ultimo in select unnest (arreglo_c) loop    //16
    arreglo_c[i]:= valor_ultimo/sum_c;    //17
    i:=i+1;    //17
    end loop;

return arreglo_c;    //18
END;
```

Ilustración 13 Código fuente del algoritmo calcularC().

El primer paso para aplicar el método del camino básico es obtener el grafo del flujo, a partir del código de la función anterior.

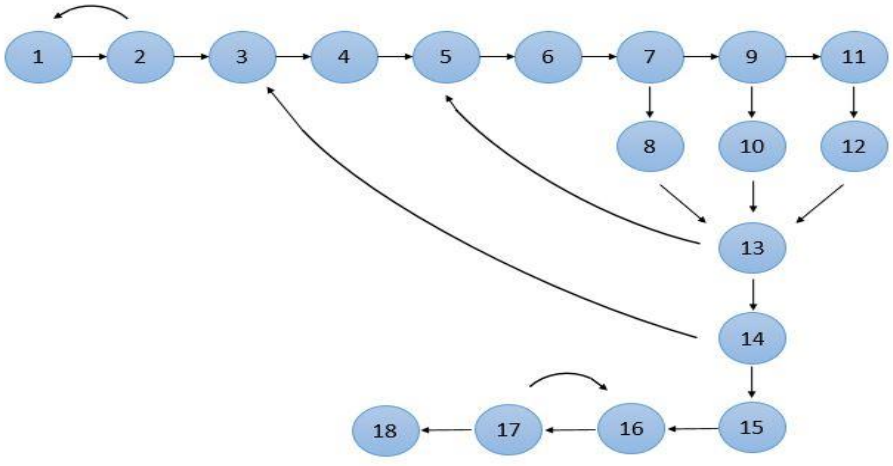

Ilustración 14. Grafo de flujo correspondiente a la funcion claculaC().

Una vez construido el grafo se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres fórmulas, de manera tal que sea el mismo resultado en cada caso:
La complejidad ciclomática V (G) de un grafo de flujo G se define como:

- V(G) = cantidad de aristas(A) – cantidad de nodos(N) +2
  V (G) = 23–18+2 = 7
- V(G) = cantidad de nodos predicados[19](P) + 1

V (G) = 6+1 = 7

- V(G) = número de regiones del grafo de flujo(R)

V (G) = 7

Luego de efectuar el cálculo se obtuvo una complejidad ciclomática de valor 7, de manera que existen 7 posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento analizado. A continuación se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución.

- Camino básico #1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 13 – 14 – 15 – 16 – 17 – 18
- Camino básico #2 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 13 – 14 – 15 – 16 – 17 – 18
- Camino básico #3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18
- Camino básico #4: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 13 – 5 – 7 – 8 – 13 – 14 – 15 – 16 – 17 – 18
- Camino básico #5: 1 – 2 – 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 13 – 14 – 15 – 16 – 17 – 18
- Camino básico #6: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 13 – 14 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 13 – 14 – 15 – 16 – 17 – 18
- Camino básico #7: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 13 – 14 – 15 – 16 – 17 – 16 – 17 – 18

Posteriormente se procede a visualizar la ejecución de un caso de prueba.

Tabla 18. Caso de prueba para el camino básico #1.

Función	CalcularC
Descripción	Se calculan los vectores de prioridad local para cada uno de los elementos ya evaluados con anterioridad.
Entradas	Nombre de la tabla, id de evaluación, id del elemento ya evaluado (criterio o proyecto).

[19] **Nodos predicados:** Son los nodos de los cuales parten dos o más aristas.

Salidas	De acuerdo valores previamente introducidos se construyó el vector de prioridad local dando como resultado que los valores de C son igual a 1.
Resultado	Satisfactorio

3.3.2 Pruebas de aceptación

A continuación se muestran algunos Casos de Prueba de Aceptación (CPA en lo adelante) que se realizaron para validar el sistema partir de las historias de usuarios y los requisitos del sistema, el resto se encuentra en documentos anexos a este trabajo.

Tabla 19. Descripción del Caso de Prueba de Aceptación HU#3.

Caso de Prueba de Aceptación		
Código: 15	**Historia de Usuario (Nro. Y nombre):** HU #5 Gestionar criterio	
Funcionalidad que se prueba: Agregar criterio.		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista o administrador.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona la opción agregar nuevo criterio.		El sistema debe mostrar un formulario para introducir los datos necesarios para agregar un nuevo criterio.
Se introducen los datos necesarios para agregar el nuevo criterio.	Datos obligatorios y opcionales.	
Se guardan los datos introducido en cada uno delos campos.	Nuevo criterio.	El sistema debe mostrar el listado con todos los criterios agregados anteriormente junto al que se acaba de agregar.

Resultado de la prueba: Satisfactoria.

Tabla 20 Descripción del Caso de Prueba de Aceptación HU#3.

Caso de Prueba de Aceptación		
Código: 16	**Historia de Usuario (Nro. Y nombre):** HU # Gestionar criterio	
Funcionalidad que se prueba: Modificar criterio.		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista, administrador o experto. Debe existir al menos un criterio.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona el criterio que se desea modificar y se selecciona la opción modificar.	El criterio a modificar.	El sistema debe mostrar un formulario con los valores actuales y permitir que estos sean editados.
Se introducen los nuevos valores de los campos que se desean modificar.	Los valores de los campos se modifican.	
Se guardan los cambios.	El criterio modificado.	El sistema debe mostrar el listado con todos los criterios existentes anteriormente mostrando además al que se acaba de modificar.
Resultado de la prueba: Satisfactoria.		

Tabla 21. Descripción del Caso de Prueba de Aceptación HU #3.

Caso de Prueba de Aceptación	
Código: 17	**Historia de Usuario (Nro. Y nombre):** HU #5 Gestionar criterio
Funcionalidad que se prueba: Mostrar criterio.	

Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista, administrador o experto. Debe existir criterio para que sea mostrado.		
Acción	**Datos de entrada**	**Resultado esperado**
Se accede a la funcionalidad para mostrar el criterio.		El sistema debe mostrar un listado con los criterios existentes hasta el momento.
Resultado de la prueba: Satisfactoria.		

Tabla 22. Descripción del Caso de prueba de Aceptación HU #3.

Caso de Prueba de Aceptación		
Código: 18	**Historia de Usuario (Nro. Y nombre):** HU #5 Gestionar criterio	
Funcionalidad que se prueba: Eliminar criterio		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista o administrador.		
Acción	**Datos de entrada**	**Resultado esperado**
Se selecciona el criterio que se desea eliminar, se selecciona la opción eliminar.	El criterio que se desea eliminar.	Mostrar listado con todos los criterios restantes.
Resultado de la prueba: Satisfactoria.		

Además de las pruebas de aceptación realizadas por el cliente, también se llevó a cabo un proceso de pruebas de caja negra mediante la técnica de particiones equivalentes, obteniéndose como resultado en la primera iteración un total de 65 no conformidades, siendo solo 40 significativas. En una segunda iteración se encontraron solo 12, las cuales fueron corregidas en su totalidad y al efectuar una tercera no se detectó ninguna.

3.4 Validación de la investigación

Para comprobar el correcto funcionamiento de la propuesta de solución se utilizaron datos históricos acerca de dos proyectos ya sometidos a un proceso de evaluación de factibilidad técnica y comercial a través del método AHP tradicional. Estos fueron evaluados por un experto en la materia y utilizando 6 criterios de evaluación.

Al realizar la evaluación a través del proceso analítico jerárquico se debe tomar en cuenta los niveles de incertidumbre arrojados por el método, si estos se encuentran por debajo del 10% entonces dichos resultados son aceptables.

Elementos que intervienen en el proceso de evaluación:

- Proyectos:
  1. Servicio Despertador Matutino para ETECSA en lo adelante Proy_SDM.
  2. Generador de Reportes en lo adelante Proy_GRP.

- Criterios:
  1. Innovación del producto
  2. Productividad del equipo de desarrollo
  3. Integración del equipo de trabajo
  4. Consumidores potenciales
  5. Hardware existente en la organización
  6. Impacto social del producto

Resultados obtenidos:

Tabla 23 Comparación entre los resultados arrojados por ambos métodos.

Proyectos	AHP	FAHP
Proy_SDM	62,86%	85.42%
Proy_GRP	37,14%	14.57%
Nivel de incertidumbre	8.58%	5.42%

Como se puede observar en la tabla ambos métodos arrojaron resultados parecidos en cuanto a cuál de los proyectos evaluados posee mejor factibilidad. Los resultados observados (analizados a través de FAHP) posen un nivel de mejora significativo con respecto a los resultados esperados (analizados a través de AHP). Los niveles de incertidumbre en ambos casos se encuentran por debajo del 10%, pero la certeza que se tiene utilizando el método FAHP es superior en un 3.16% que la que se posee si se realiza el análisis a través del AHP.

Para demostrar el nivel de significación de los resultados obtenidos a través del FAHP se realizó la prueba de chi cuadrado a los valores de la tabla 22 utilizando la herramienta SPSS[20]. Los valores arrojados por la herramienta se muestran en la ilustración 15. El valor de chi cuadrado es de 6.4 lo que está por en encima de 5 que es el valor mínimo esperado, esto asegura que la mejor vía de evaluación es la llevada a cabo a través del método FAHP. Además estadísticamente los resultados de significación se indican con Asymp. Sig[21] en valores por debajo de 0.05, en este caso dicho valor es de 0.011. Este análisis demuestra que los resultados que se obtienen al aplicar el Proceso Analítico Jerárquico Difuso son más certeros que al realizar el proceso de la forma tradicional y por tanto se reduce la incertidumbre presente en la toma de decisiones de forma significativa.

Test Statistics

	Incertidumbre
Chi-Square[a]	6.400
df	1
Asymp. Sig.	.011

a. 0 cells (.0%) have expected frequencies less than 5. The minimum expected cell frequency is 5.0.

Ilustración 15 Resultados arrojados por la herramienta SPSS.

3.5 Conclusiones del capítulo

La aplicación de las métricas TOC y RC demostraron que el diseño del sistema no es complejo y que el predominio de niveles bajos de responsabilidad y complejidad, favorecen un alto grado de reutilización.

[20] **SPSS:** Acrónimo del inglés Statistical Product and Service Solutions.
[21] **Asymp. Sig:** significación asintótica

La realización de pruebas a la herramienta desarrollada aseguró la adecuada correspondencia entre las funcionalidades implementadas y los requisitos funcionales planteados inicialmente.

La ejecución de pruebas unitarias a varias funcionalidades del sistema posibilitó evaluar en buena medida la implementación, pues se obtuvieron los resultados esperados en cada caso.

Finalmente la realización de las pruebas de aceptación permitió evidenciar la satisfacción del cliente con la herramienta desarrollada.

CONCLUCIONES

- El estudio de las herramientas informáticas y modelos que se utilizan para llevar a cabo estudios de factibilidad de proyectos, demostró la necesidad de desarrollar un sistema que tuviera en cuenta la incertidumbre, haciendo uso del método: Proceso Analítico Jerárquico Difuso (FAHP).
- La realización del análisis sobre la metodología de desarrollo de software, los lenguajes de programación y las herramientas, permitió justificar la utilización de las mismas y sentar las bases desde el punto de vista técnico, para iniciar el desarrollo.
- Con la definición de los requisitos se lograron identificar las actividades a informatizar. Además, su verificación posibilitó la construcción del modelo de diseño.
- La aplicación de métricas al diseño permitió valorar positivamente el mismo para dar paso a la implementación, con la cual se lograron satisfacer las necesidades del cliente y resolver los problemas por los que se decidió comenzar el desarrollo de la herramienta.
- La aplicación de pruebas al software permitió evaluar en buena medida el nivel de calidad de la implementación de la solución.
- El desarrollo del sistema proporcionó al cliente una herramienta para evaluar la factibilidad técnica y comercial en los proyectos de software haciendo uso de la lógica difusa.

RECOMENDACIONES

1. Se recomienda la utilización de la herramienta en los centros de producción de la universidad para hacer un correcto estudio de los proyectos antes de que estos sean iniciados.
2. Seguir perfeccionado los estudios de factibilidad en los diferentes momentos del ciclo de vida de los proyectos, así como proponer nuevos indicadores en función de esto.

BIBLIOGRAFÍA

ABRAN, A., BOURQUE, P., DUPUIS, R. y MOORE, J.W. 2001. *Guide to the software engineering body of knowledge-SWEBOK* [en línea]. S.l.: IEEE Press. [Consulta: 7 mayo 2015]. ISBN 0769510000. Disponible en: http://dl.acm.org/citation.cfm?id=580192.

ABREU, PEÑA MARIETA, 2012. *Modelo para análisis de factibilidad en la evaluación de proyectos de software.* 2012. S.l.: s.n.

ACEVEDO, K., EDNA, A. y BARRIOS, J. 2010. Estudio de Factibilidad de un Proyecto. *Universidad del Atlántico. Disponible e n www. slideshare. net*,

ACEVEDO, K., EDNA, A. y BARRIOS, J. 2010. Estudio de Factibilidad de un Proyecto. *Universidad del Atlántico. Disponible e n www. slideshare. net*,

AGUARON, J. y MORENO-JIMÉNEZ, J.M. 2003. The geometric consistency index: Approximated thresholds. *European Journal of Operational Research*, vol. 147, no. 1, pp. 137–145.

APACHE, U. 2011. Apache Software Foundation. *URL http://java. apache. org*,

ARZA PÉREZ, L., VERDECIA MARTÍNEZ, E.Y. y LAVANDERO GARCÍA, J. 2013. MODELO COMPUTACIONAL PARA LA RECOMENDACIÓN DE ROLES EN EL PROCESO DE UBICACIÓN DE ESTUDIANTES EN LA INDUSTRIA DE SOFTWARE. [en línea], [Consulta: 16 junio 2015]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/7925.

AZNAR BELLVER, J. y GUIJARRO MARTÍNEZ, F. 2012. *Nuevos métodos de valoración. Modelos Multicriterio* [en línea]. 2da. S.l.: Editorial Universitat Politècnica de València. [Consulta: 29 abril 2015]. ISBN 978-84-8363-982-5. Disponible en: http://riunet.upv.es/handle/10251/19181.

BACCA, A. y OTHERS 2015. Metodologías para la implementación de proyectos de tecnología. Un Caso de estudio en la virtualización de aplicaciones y hardware. [en línea], [Consulta: 17 mayo 2015]. Disponible en: http://repository.unimilitar.edu.co/handle/10654/13220.

BASS, L. 2007. *Software architecture in practice*. S.l.: Pearson Education India. ISBN 0-321-15495-9.

BECK, K. 2000. *Extreme programming explained: embrace change* [en línea]. S.l.: Addison-Wesley Professional. [Consulta: 15 junio 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=G8EL4H4vf7UC&oi=fnd&pg=PR13&dq=Kent+Beck&ots=j9vHvtfSzq&sig=Co7MWa387ixeMAu2nWJnA-zIZSE.

BUCKLEY, J.J. 1985. Fuzzy hierarchical analysis. *Fuzzy sets and systems*, vol. 17, no. 3, pp. 233–247.

BÜYÜKÖZKAN, G., KAHRAMAN, C. y RUAN, D. 2004. A fuzzy multi-criteria decision approach for software development strategy selection. *International Journal of General Systems*, vol. 33, no. 2-3, pp. 259–280.

CALDEIRA, C. 2015. PostgreSQL: Guia Fundamental. [en línea], [Consulta: 17 mayo 2015]. Disponible en: http://www.silabo.pt/Conteudos/7950_PDF.pdf.

CANÓS, J., LETELIER, P. y PENADÉS, M.C. 2003. Metodologías Ágiles en el desarrollo de Software. *Universidad Politécnica de Valencia, Valencia* [en línea], [Consulta: 29 abril 2015]. Disponible en: http://www.willydev.net/Willydev_old/Root/descargas/prev/TodoAgil.Pdf.

CASCALES, M. del S.G. 2009. *Métodos para la comparación de alternativas mediante un sistema de ayuda a la decisión: SAD y« Soft computing»* [en línea]. S.l.: Universidad Politécnica de Cartagena. [Consulta: 29 abril 2015]. Disponible en: http://dialnet.unirioja.es/servlet/tesis?codigo=22138.

CHAIN, N.S. 2007. *Proyectos de inversión: formulación y evaluación* [en línea]. S.l.: Pearson Educación. [Consulta: 29 abril 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=pIS1QnFYt5IC&oi=fnd&pg=PT8&dq=Easy PlanEx&ots=3vcaN66V4H&sig=1_Fy-Euo7e79CSL0VcMjeVMaTfU.

CHAVES, M.A. 2011. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes* [en línea], vol. 6, no. 10. [Consulta: 27 mayo 2015]. Disponible en: http://revistas.ucr.ac.cr/index.php/intersedes/article/view/790.

CRUZ RAMÍREZ, M., CEPENA, M. y CARIDAD, M. 2012. Perfeccionamiento de un instrumento para la selección de expertos en las investigaciones educativas. *Revista electrónica de investigación educativa*, vol. 14, no. 2, pp. 167–179.

DANIARYS, R.A. 2012. *Etapas del Análisis de Factibilidad. Compendio bibliográfico. Centro universitario Vladimir Ilich Lenin, Las Tunas, Cuba*. S.l.: s.n.

DÍAZ, P.A.B., BALLESTER, V.A.C., ALCARAZ, J.L.G. y INIESTA, A.A. 2012. El Proceso Jerárquico Analítico y Lógica Difusa: Sus Aplicaciones. [en línea], [Consulta: 29 abril 2015]. Disponible en: http://celaya.academiajournals.com/downloads/Tomo%2003.pdf.

EGUÍLUZ, J. 2008. Introducción a JavaScript. *Versión electrónica consultada en www. librosweb. es el* [en línea], vol. 20. [Consulta: 7 mayo 2015]. Disponible en: http://librosweb.es/javascript/pdf/introduccion_javascript_2caras.pdf.

EGUILUZ, J. 2012. Desarrollo Web Ágil con Symfony 2. *easybook*,

EGUÍLUZ PÉREZ, J. 2008. Introducción a CSS. *España. URL: http://librosweb. es/css*,

ERTUĞRUL, İ. y KARAKAŞOĞLU, N. 2009. Performance evaluation of Turkish cement firms with fuzzy analytic hierarchy process and TOPSIS methods. *Expert Systems with Applications*, vol. 36, no. 1, pp. 702–715.

ESCALONA, M.J. y KOCH, N. 2002. Ingeniería de Requisitos en Aplicaciones para la Web–Un estudio comparativo. *Universidad de Sevilla* [en línea], [Consulta: 30 abril 2015]. Disponible en: https://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

ESPINOSA, A.T., SAGREDO, J.G.C., REYES, M.M. y GARCÍA, M. de L.L. 2012. Automatización de la codificación del patrón modelo vista controlador (MVC) en proyectos orientados a la Web. *CIENCIA ergo-sum*, vol. 19, no. 3, pp. 239–250.

GREY, L.G. y VILTRES, Y.M. 2011. PROCESO DE MEJORA DEL SISTEMA DE GESTIÓN DE PROYECTOS PARA CUBA Y VENEZUELA. IMPROVEMENT PROCESS OF PROJECTS MANAGEMENT SYSTEM FOR CUBA AND VENEZUELA. [en línea], [Consulta: 26 mayo 2015]. Disponible en: http://www.informatica-juridica.com/trabajos/Improvement_process_of_projetcs_management_system_for_Cuba_and_Venezuela.pdf.

GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E. 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*, vol. 24, no. 3, pp. 103–114.

HERRERA UMAÑA, M.F. y OSORIO GÓMEZ, J.C. 2006. Modelo para la gestión de proveedores utilizando AHP difuso. *Estudios Gerenciales*, vol. 22, no. 99, pp. 69–88.

HEURTEL, O. 2011. *PHP 5.3: desarrollar un sitio Web dinámico e interactivo* [en línea]. S.l.: Ediciones ENI. [Consulta: 29 abril 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=GcymrdA9lZoC&oi=fnd&pg=PA10&dq=php+5.3&ots=8HYwLibOzq&sig=s3y2yBZfqhD6Rudyl1OMORrVYvY.

HUANG, L.-C. y WU, R.Y.-H. 2005. Applying fuzzy analytic hierarchy process in the managerial talent assessment model–an empirical study in Taiwan's semiconductor industry. *International Journal of Technology Management*, vol. 30, no. 1, pp. 105–130.

HURTADO, T. y BRUNO, G. 2005. El Proceso de análisis jerárquico (AHP) como herramienta para la toma de decisiones en la selección de proveedores. *Trabajo de grado (Licenciado en Investigación Operativa), Universidad Nacional de San Marcos. Facultad de Ciencias Matemáticas. EAP de Investigación Operativa., Lima*,

INSTITUTE, P.M. 2013. *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK).* S.l.: s.n.

ISHIZAKA, A. y LABIB, A. 2009. Analytic hierarchy process and expert choice: Benefits and limitations. *OR Insight*, vol. 22, no. 4, pp. 201–220.

ITURBE, M.C.E. y DEL VALLE CAMPOAMOR, E.A.G. 2011. MÉTODOS DE DECISIÓN MULTICRITERIO. [en línea], [Consulta: 27 mayo 2015]. Disponible en: http://tesiuami.uam.mx/revistasuam/Denarius/include/getdoc.php?id=318&.

JOSKOWICZ, J. 2008. Reglas y prácticas en eXtreme Programming. *Universidad de Vigo*, pp. 22.

LIM, H.-S., LEE, J.-S., CHON, H.-T. y SAGER, M. 2008. Heavy metal contamination and health risk assessment in the vicinity of the abandoned Songcheon Au–Ag mine in Korea. *Journal of Geochemical Exploration*, vol. 96, no. 2, pp. 223–230.

OTTO, M. y THORNTON, J. 2013. Bootstrap. *Twitter Bootstrap*,

PARADIGM, V. 2010. Visual paradigm for uml. *Visual Paradigm for UML-UML tool for software application development*,

PEÑA ABREU, M. y PIÑERO PÉREZ, P.Y. 2013. Modelo para análisis de factibilidad en la evaluación de Proyectos de Software. [en línea], [Consulta: 27 mayo 2015]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/7987.

PERISSÉ, M.C. 2001. Proyecto informático: una metodología simplificada. *Buenos Aires, Argentina: Ciencia y Técnica Administrativa* [en línea], [Consulta: 29 abril 2015]. Disponible en: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/ficha/ficha.htm.

PIÑERO-PÉREZ, P. 2013. COLECTIVO DE AUTORES. GESPRO. Paquete para la gestión de proyectos. *Revista Nueva Empresa*, vol. 9, no. 1, pp. 45–53.

POREBSKI, B., PRZYSTALSKI, K. y NOWAK, L. 2011. *Building PHP Applications with Symfony, CakePHP, and Zend Framework* [en línea]. S.l.: John Wiley and Sons. [Consulta: 29 abril 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=gTEl2mWGNtAC&oi=fnd&pg=PR27&dq=Twig+in+Symfony&ots=KOJL7cdBeB&sig=2M-6ncokas8YyhXieMuQrkNLnqk.

PRESSMAN, R.S. 1997. *Ingeniería del Software: Un enfoque práctico* [en línea]. S.l.: Mikel Angoar. [Consulta: 26 mayo 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=8UV5jxkuBZIC&oi=fnd&pg=PP3&dq=Ingenier%C3%ADa+de+software.+Un+enfoque+pr%C3%A1ctico,+vol.+1.+pressman&ots=wJQp0RPlGJ&sig=HAl4C9peSjLOjRprnJJXPkRBqS0.

PRESSMAN, R.S. 2005. *Ingeniería de Software, Sexta Edición, Ed*. S.l.: Mc Graw Hill, Mexico. ISBN 10: 84-7829-074-1.

QUINTERO, J.B., DE PÁEZ, R.A., MARÍN, J.C. y LÓPEZ, A.B. 2012. Un estudio comparativo de herramientas para el modelado con UML. *Revista universidad eafit*, vol. 41, no. 137, pp. 60–76.

RODRÍGUEZ BATISTA, A. 2005. Impacto social de la ciencia y la tecnología en Cuba: una experiencia de medición a nivel macro. *Revista iberoamericana de ciencia tecnología y sociedad*, vol. 2, no. 4, pp. 147–171.

SAATY, T.L. 1994. How to make a decision: the analytic hierarchy process. *Interfaces*, vol. 24, no. 6, pp. 19–43.

SAKIPOVA, D.Y.T. 2011. Modelado del Negocio, Sistema y Análisis de los Requisitos del Módulo Inventario del Proyecto de Software ERP. [en línea], [Consulta: 21 mayo 2015]. Disponible en: http://www.laccei.org/LACCEI2011-Medellin/published/IT246_Torres.pdf.

SÁNCHEZ REIG, M.D.C. 2015. *Evaluación de los impactos ambientales derivados de nanomateriales aplicados al envase y embalaje* [en línea]. S.l.: s.n. [Consulta: 27 mayo 2015]. Disponible en: https://riunet.upv.es/handle/10251/48554.

SARMIENTO, J. 2014. UML: Diagrama de Despliegue [en línea]. *Bogotá: La Empresa [citado 20 septiembre, 2014]. Disponible en Internet:< URL: http://umldiagramadespliegue. blogspot. com*,

SERRANO, S.C. y AVILÉS, R.A. 2014. La gestión de proyectos en los estudios universitarios de Información y Documentación en Iberoamérica. *BiD: Textos universitaris de biblioteconomia i documentació*, no. 32, pp. 6-.

TABARES, R.B. 2011. Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. *Entre Ciencia e Ingeniería*, no. 8, pp. 161–173.

WAGE, J.H. y VESTERINEN, K. 2009. Doctrine ORM for PHP. *Sensio SA, April*,

ZADEH, L.A. 1974. Fuzzy Logic and Its Application to Approximate Reasoning. *IFIP Congress*. S.l.: s.n.,

ZAPATA CORTÉS, J.A., ARANGO SERNA, M.D. y ADARME JAIMES, W. 2012. Aplicación del proceso de análisis jerárquico extendido con lógica difusa para la selección de software para logística. *Ingeniería e Investigación*, vol. 32, no. 1, pp. 94–99.