Universidad Central “Marta Abreu” de Las Villas

Facultad Matemática, Física y Computación

INGENIERÍA INFORMÁTICA

TRABAJO DE DIPLOMA

“AIChoose: Plataforma para la definición y solución de sistemas de toma de decisiones basados en problemas con múltiples criterios.”

Autor:

Francisco Javier Peña Veitía

Tutores:

Dr. Rafael Esteban Bello Pérez
Dr. Daniel Gálvez Lio

Santa Clara, Cuba 2016

A mi madre, por darme su amor, guía y apoyo incondicional.

A mi familia, por ayudarme a alcanzar mis metas.

A Beatriz, por brindarme su amor inmenso.

A mis tutores, por compartir sus conocimientos y experiencia.

A mis amigos, por su ayuda.

A mis profesores, por sus enseñanzas.

Gracias a todos los que me apoyaron.

Resumen

Es común enfrentarse a situaciones donde existen un conjunto de alternativas o decisiones posibles de las cuales se debe escoger la mejor entre ellas o establecer un orden prioritario. Aunque la resolución de problemas de toma de decisiones basados en problemas de múltiples criterios y el estudio de métodos y algoritmos con este propósito ha tenido un gran auge en la actualidad, el acceso a herramientas computacionales que resuelvan estos problemas es prácticamente nulo. Este proyecto persigue integrar bajo una misma plataforma varios métodos de resolución a estos tipos de problemas, utilizando como lenguaje de programación Java en su versión estándar. AIChoose o Artificial Intelligence Choose, la Inteligencia Artificial elige, en español; es una herramienta desarrollada bajo software libre que pretende asimilar los conceptos fundamentales de los problemas de toma de decisiones e implementarlos correctamente mediante el uso de diferentes patrones de arquitectura y patrones de diseño, asegurando la flexibilidad, extensibilidad e integración con otras herramientas. La plataforma resultante le permite al usuario definir y resolver problemas de toma de decisiones de una manera fácil, rápida y segura a través de una interfaz amigable, fácil de usar y potente. El diseño de la misma permite la adición de nuevos métodos que resulten de interés considerar.

Abstract

It is common to face situations where they exist a group of alternative or decisions possible of which the best should be chosen among them or to establish a high-priority order. Although the resolution of problems of taking of decisions based on problems of multiple approaches and the study of methods and algorithms has had a great peak at the present time, the access to computational tools that solve these problems is practically null. This project pursues to integrate under oneself platform several resolution methods to these types of problems, using as programming language Java in its standard version. AIChoose or Artificial Intelligence Choose, it is a tool developed under free software that seeks to assimilate the fundamental concepts of the problems of taking of decisions and to implement them correctly by means of the use of different architecture patterns and design patterns, assuring their flexibility, extensibility and integration with other tools. The resulting platform allows to the user to define and to solve problems of taking of decisions in an easy, quick and sure way through a friendly, easy to use and potent interface.

Tabla de contenidos

Introducción

Contexto

Es común enfrentarse a situaciones donde se tiene un conjunto de alternativas y se debe elegir la mejor entre ellas; en este caso, se está en presencia de un problema de toma de decisiones. Este tipo de problemática, tiene como finalidad la selección de la mejor solución ante un conjunto de posibles alternativas, utilizando para ello determinados juicios y/o la verificación del cumplimiento de requisitos en las alternativas (Cables Pérez, 2011).

La revisión de la literatura científica acerca de la solución a problemas de toma de decisión con múltiples criterios y expertos ha permitido identificar los siguientes elementos:

- Existencia de múltiples métodos para solucionar este tipo de problemas.
- Los autores proponen nuevos métodos o variaciones de los existentes regularmente.
- El acceso a herramientas que resuelven este tipo de problemas se hace complicado ya que tienen altos costos en el mercado.
- La mayoría de las herramientas son de licencia privada y carecen de mecanismos para su extensión sin tener que hacer cambios significativos en la arquitectura del sistema.

Problema de investigación

- Existen varias herramientas de ayuda a la toma de decisiones las cuales son difíciles de obtener y son propietarias.

- Las herramientas existentes carecen de mecanismos para su extensión.

Objetivo general

Desarrollar una herramienta computacional flexible y extensible que permita gestionar el diseño y prueba de sistemas para la toma de decisiones con múltiples criterios mediante varios métodos de solución integrados bajo una misma plataforma.

Objetivos específicos

1- Modelar los conceptos fundamentales de los sistemas de toma de decisiones de modo que queden correctamente representados dentro de la herramienta computacional.

2- Definir estrategias para la implementación computacional de la herramienta que garantice su flexibilidad y extensibilidad.

3- Implementar e integrar en la herramienta varios métodos de solución a problemas de toma de decisiones.

4- Diseñar y probar distintos casos de estudio en la herramienta computacional.

Preguntas de investigación

- ¿Cómo representar los conceptos fundamentales de los sistemas de toma de decisiones correctamente dentro de una herramienta computacional?

- ¿Qué diseño garantiza la flexibilidad y extensibilidad de la herramienta computacional?

Justificación de la investigación

A diario nos enfrentamos a problemas de toma de decisiones en diversos ámbitos que necesitan solución. En estos momentos nuestro país aboga por la soberanía tecnológica a través del software libre y el código abierto. Con los cambios y avances constantes en las tecnologías de la informática urge un software flexible y que se extienda con facilidad para poder dar solución a nuevos problemas.

Descripción de los capítulos

El primer capítulo se titula “Fundamentación teórica”, tiene como objetivo principal analizar críticamente y presentar los elementos teóricos que sirven de base a la investigación. Se expondrán los conceptos fundamentales que están ligados a la solución de problemas de toma

de decisiones con múltiples criterios. También se realizará una descripción de los sistemas existentes que dan solución a este tipo de problemas. Y por último se explican las herramientas y proceso de desarrollo a utilizar en la construcción de la propuesta de solución.

El segundo capítulo se titula “Modelo del Negocio y Requisitos”, contiene los elementos que caracterizan la propuesta de solución. Se explica el modelo de negocio, se establecen las reglas de negocio y se especifican las características y funcionalidades del sistema a partir de los requisitos funcionales y no funcionales identificados. Además, se describen los casos de uso más significativos del sistema.

El tercer capítulo se titula “Descripción de la propuesta de solución”, tiene como objetivo mostrar el proceso de implementación de la plataforma. El capítulo describe la arquitectura del sistema y la construcción de la solución de software propuesta. Se presentan los diagramas de clases del diseño fundamentales y diagramas de secuencia. Se exponen los principios de diseño usados en la implementación y se explica cómo se llevó a cabo el tratamiento de errores.

El cuarto capítulo y último se titula “Pruebas y análisis de factibilidad”, donde se expone el análisis de costo del sistema y de la descripción de las pruebas de software realizadas.

“Fundamentación teórica”.

1. Fundamentación teórica

Este capítulo tiene como objetivo principal analizar críticamente y presentar los elementos teóricos que sirven de base a la investigación. Se expondrán los conceptos fundamentales que están ligados a la solución de problemas de toma de decisiones con múltiples criterios. También se realizará una descripción de los sistemas existentes que dan solución a este tipo de problemas. Por último se explican las herramientas y artefactos usados para el proceso de desarrollo de la propuesta de solución.

1.1 Objeto de estudio

El objeto de estudio son los patrones de diseño que favorecen el desarrollo de una herramienta para la de toma de decisiones con múltiples criterios que sea flexible y extensible.

1.1.1 Proceso de toma de decisiones

En el mundo la toma de decisiones es un proceso cotidiano y de suma importancia en diferentes ámbitos de la sociedad tales como la gestión de recursos humanos, la educación y el comercio (F. Shipley and Johnson, 2009, A. Krohling and C. Campanharo, 2010a). En la actualidad la toma de decisiones con múltiples criterios y expertos ante situaciones complejas sigue siendo un proceso engorroso y complejo. En la solución de problemas de este tipo se involucran múltiples criterios y los juicios emitidos por los expertos lo que muchas veces supera las capacidades de procesamiento de información del ser humano que debe darle solución. También es conocido que el juicio intuitivo humano y la toma de decisiones derivados de este no son óptimos, siendo afectados por la complejidad e incertidumbre propias del problema. Ofrecer una herramienta de ayuda que erradique las deficiencias anteriormente expuestas constituye una de las líneas principales de la ciencia a lo largo de la historia (Sánchez González et al., 2013).

1.1.2 Métodos de solución a problemas de toma de decisiones

El análisis de la solución a los problemas de toma de decisión ha sido ampliamente reflejado por la literatura y existen un gran número de métodos para obtener la propuesta de solución entre los que se encuentran el Proceso Analítico en Red (ANP por sus siglas en inglés)(Ayub et al., 2009), Proceso de Análisis Jerárquico (AHP por sus siglas en inglés)(Karimi et al., 2011), PROMETHEE(Chen-Tung et al., 2009), DELPHI (Kaufmann and Gil Aluja, 1993), TOPSIS (Jadidi et al., 2008) y ELECTRE (ELimination Et Choix Traduisant la

R´Ealite)(MARTÍNEZ CHÁVEZ, 2013) entre otros. Un problema generalizado en la literatura es que muchos autores proponen nuevos métodos o variaciones de los existentes regularmente (A. Krohling and C. Campanharo, 2010b),(F. Shipley and Johnson, 2009); sin embargo, hoy son pocas las plataformas que permiten definir el problema de decisión, adicionar nuevos métodos y realizar experimentos que permitan comparar los resultados ofrecidos por diferentes métodos de solución. Entre ellas, *1000Minds, Criterium DecisionPlus, D-Sight, DecideIT, Decision Lens, Expert Choice, PriEsT,* MultiDecision PAAT, Apolo, entre otras.

Para este trabajo se analizan los métodos AHP y ELECTRE por su amplio uso en distintas esferas en el marco de toma de decisiones. El método de solución AHP permite al decisor tener una visión más clara del problema a tratar. Gracias a la manera en que estructura el problema en forma de jerarquía permite la modelación de problemas complejos. Por otra parte el método de solución ELECTRE es ampliamente conocido y utilizado también. En muchas ocasiones no se tiene mucha información sobre las alternativas, ahí es cuando el método ELECTRE juega un papel fundamental, siendo así uno de los métodos de solución basado en relaciones de sobreclasificación más usados.

El Proceso de Análisis Jerárquico

El Proceso de Análisis Jerárquico (AHP[1]) es un método inventado por el Dr. Thomas Saaty que se basa en descomponer jerárquicamente el problema a tratar en sus componentes. En el más alto nivel de la jerarquía se ubica entonces el objetivo a alcanzar. El segundo nivel está constituido por los sub-objetivos y criterios que permitirán evaluar el grado de logro de los mismos por las diferentes alternativas, que están en el tercer nivel. De esta manera las alternativas serán ordenadas jerárquicamente a través de los pesos globales calculados mediante el método (Cables Pérez, 2011).

Según (Cables Pérez, 2011) AHP consta del siguiente algoritmo:

Paso 1: Repetir los pasos siguientes para cada criterio:

[1] Del ingles *Analytic Hierarchy Process*

Paso 1.1: Crear la Matriz de Comparación por Pares de alternativas para cada uno de los criterios y establecer el peso de importancia relativa entre cada par de alternativas consideradas. El peso de importancia se establece a partir de una escala del 1 al 9, donde 1 es muy poco importante y 9 es extremadamente importante. Se aplica un peso recíproco cuando la segunda alternativa es más importante que la primera. El valor 1 siempre es asignado a la comparación de una alternativa con sí misma.

Paso 1.2: Crear la Matriz Normalizada, donde se divide cada número de la Matriz de Comparación por Pares por la suma total de su columna.

Paso 1.3: Determinar el Vector de Prioridad para el criterio calculando el promedio de cada fila de la Matriz Normalizada. Este promedio por fila representa el Valor de Prioridad de las alternativas con respecto al criterio seleccionado.

Paso 1.4: Realizar comparaciones por pares entre las opiniones y juicios expresados por los expertos.

Paso 2: Resumir en una Matriz de Prioridad los resultados obtenidos del paso 1.3, listando las alternativas por fila y los criterios por columna.

Paso 3: Confeccionar una Matriz de Comparación por Pares de criterios de manera similar a como se hizo para las alternativas en los pasos del 1.1 al 1.3.

Paso 4: Calcular el Vector de Prioridad Global multiplicando el Vector de Prioridad de los criterios obtenido en el Paso 3 por la Matriz de Prioridad de las alternativas obtenido en el Paso 2.

Paso 8: Seleccionar la alternativa que tenga un mayor valor en el Vector de Prioridad Global.

Algunas aplicaciones de método AHP:

- Establecimiento de un centro de atención de primer nivel que incluya los servicios de medicina general y odontología para cubrir la demanda de la población del norte del

Cauca (Colombia)(Osorio Gómez, 2008).

- Selección de personal en Empresas e Instituciones (Cables Pérez, 2011).
- Algoritmos de cálculo de vectores de prioridad a partir de matrices de comparación por pares imprecisas (RUIZ-TAGLE MOLINA, 2011).

ELECTRE

El método ELECTRE (ELimination Et Choix Traduisant la R´Ealite), fue propuesto por Benayoun, Roy y Sussman en 1966 [97], y se destaca por su utilización práctica desde finales de la década de los años 60. El principio básico que sigue el mismo consiste en dividir las alternativas en dos subconjuntos, las más y las menos favorables, utilizando una relación de sobreclasificación mediante la comparación por pares, calculando para cada par de alternativas el índice de concordancia y de discordancia de forma simultánea (Cables Pérez, 2011).

Este método organiza su proceso de la forma siguiente según (Cables Pérez, 2011):

Paso 1:

Conformar la matriz de decisión *R* y definir el vector de pesos W.

La matriz de decisión tiene la forma siguiente:

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{pmatrix}$$

Figura 1.1: Matriz de decisión

donde:

Los *rij* son los juicios o evaluaciones de cada alternativa *Ai*, *i* = 1, ..,*m* para el criterio *Cj* y *j* = 1, .., *n*. En el caso del vector de pesos *W* = (w_1,w_2,...,w_n), se utiliza para indicar la importancia de cada criterio *Cj*, con *j* = 1, ..., *n*. Además, los valores del vector de pesos deben pertenecer al intervalo unitario y su suma ser igual a 1.

Paso 2:

Normalizar la matriz de decisión inicial (R^*). Para esta operación se puede utilizar cualquier método de normalización. Por ejemplo:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$$

Figura 1.2: Fórmula de normalización

Paso 3:

Ponderar la matriz de decisión normalizada con los pesos *wj* , *j* = 1, ..., *n*, obteniéndose una matriz con la influencia de los pesos definidos. Esta operación consiste en que cada columna de la matriz R^* se multiplica por su correspondiente peso, resultando:

$$R_P^* = R \cdot W = \begin{pmatrix} r_{11}^* \cdot w_1 & r_{12}^* \cdot w_2 & \cdots & r_{1n}^* \cdot w_n \\ r_{21}^* \cdot w_1 & r_{22}^* \cdot w_2 & \cdots & r_{2n}^* \cdot w_n \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1}^* \cdot w_1 & r_{m2}^* \cdot w_2 & \cdots & r_{mn}^* \cdot w_n \end{pmatrix}$$

Figura 1.3: Matriz de decisión ponderada

Paso 4:

Calcular la matriz del índice de concordancia (ic). La misma es una matriz de orden $m \times m$ donde sus elementos expresan la cuantificación de los criterios con que una alternativa es preferida sobre otra, al realizar una comparación por pares. Luego el índice de concordancia en el proceso de comparación se obtiene mediante la fórmula:

$$ic(A_l, A_k) = \sum_{j \in Y} w_j$$

Figura 1.4: Función para el cálculo de la matriz de concordancia

donde el conjunto *Y* está formado por todos los subíndices relativos a los criterios para los cuales la alternativa *Al* es mejor que la alternativa *Ak*. En los casos donde las alternativas tienen la misma preferencia (o están empatadas) el peso correspondiente se divide entre las dos. Esta expresión no es más que la suma de los pesos asociados a los criterios para los cuales la alternativa *Al* es mejor que la alternativa *Ak* o las alternativas tienen la misma preferencia.

Paso 5:

Calcular la matriz del índice de discordancia (id). Al igual que para *ic* es una matriz de orden *m* × *m*, que se obtiene a partir de la matriz R_p^* y los elementos de la misma expresan la cuantificación de los criterios con que una alternativa es dominada sobre otra, al realizar una comparación por pares. Luego el índice de discordancia en el proceso de comparación se obtiene mediante la fórmula:

$$id(A_l, A_k) = \frac{\max\limits_{j \in X} \left| r^*_{P\,l\,j} - r^*_{P\,k\,j} \right|}{\max\limits_{j \in \{1,..,n\}} \left| r^*_{P\,l\,j} - r^*_{P\,k\,j} \right|}$$

Figura 1.5: Función para el cálculo de la matriz de discordancia

donde el conjunto *X* está formado por todos los subíndices de los criterios para los cuales la alternativa *Al* está dominada por la alternativa *Ak*.

Paso 6:

Fijar los umbrales o niveles de exigencia:

- un umbral mínimo *UC*mín para el índice de concordancia (Media de *ic*).
- un umbral máximo *UD*máx para el índice de discordancia (Media de *id*).

Paso 7:

Determinar la matriz de dominancia concordante (*dc)*. Para obtener dicha matriz (*dc*), se parte de la matriz de índice de concordancia *ic* y se compara cada elemento de la misma con

el umbral mínimo definido *UC*mín para efectuar la asignación del valor correspondiente de los elementos de la matriz *dc*, utilizando en este caso la función siguiente:

$$dc_{l\,k} = \begin{cases} 1 & \text{si} \quad ic_{l\,k} > UC_{\text{mín}} \\ 0 & \text{si} \quad ic_{l\,k} \leq UC_{\text{mín}} \end{cases}$$

Figura 1.6: Función para el cálculo de la matriz de dominancia concordante

Paso 8:

Determinar la matriz de dominancia discordante *dd*. Para obtener dicha matriz (*dd*), se parte de la matriz de índice de discordancia *id* y se compara cada elemento de la misma con el umbral máximo definido *UD*máx para efectuar la asignación del valor correspondiente de los elementos de la matriz *dd*, utilizando en este caso la función siguiente:

$$dd_{l\,k} = \begin{cases} 1 & \text{si} \quad id_{l\,k} < UD_{\text{máx}} \\ 0 & \text{si} \quad id_{l\,k} \geq UD_{\text{máx}} \end{cases}$$

Figura 1.7: Función para el cálculo de la matriz de dominancia discordante

Paso 9:

Determinar la matriz de dominancia agregada *da* (concordante-discordante), donde los elementos de la misma se obtienen a partir de los elementos homólogos de las matrices *dc* y *dd*, a través de la función:

$$da_{l\,k} = dc_{l\,k} \cdot dd_{l\,k}$$

Figura 1.8: Cálculo de la matriz de dominancia agregada

Paso 10:

Interpretar los valores de la matriz de dominancia agregada *da*, tal que, si el elemento da *k* = 1, significa que la alternativa *Al* es mejor que la alternativa *Ak*, para un número importante de los criterios utilizados, y de esta forma se conforma el núcleo del conjunto de alternativas. Luego:

Hasta aquí llega el método ELECTRE clásico, de aquí en adelante se propone establecer un orden entre las alternativas, apoyándose en el *conjunto frente paretiano óptimo*, estableciendo una jerarquía H donde las alternativas que vayan quedando en los niveles más altos tendrán mayor prioridades.

Conjunto frente paretiano óptimo (*P*): Dado un conjunto de soluciones o alternativas factibles(S), el conjunto P de aquellas alternativas no dominadas en el espacio de búsqueda o dominio es llamado frente paretiano óptimo (Burke and Kendall, 2005).

Paso 11:

Encontrar P=$\{P_0, P_1, P_2, \ldots, P_m\}$ para la matriz de dominancia agregada (*da*), que expresa las relaciones por pares de sobreclasificación entre un conjunto de alternativas A=$\{A_0, A_1, A_2, \ldots, A_n\}$, obtenida como resultado de la aplicación del método ELECTRE.

Algoritmo para la obtención de P en *da*:

1. Inicializar $P_0 = A_0$, i=1
2. j=0
3. Comparar A_i con P_j
4. Si A_i supera P_j, eliminar P_j de P

   Sino si P_j supera a A_i incrementar i e ir al paso 3.
5. Si j < m, incrementar j e ir al paso 3,

   Si no, ir al paso 6.
6. Insertar A_i en P
7. Si i < n, incrementar i e ir al paso 2,

   Si no, declarar P como Conjunto frente paretiano óptimo**.**

Paso 12:

Ubicar los elementos de P, en el nivel de la jerarquía correspondiente al número de la iteración. Por ejemplo, si es la primera iteración los elementos en P se ubicarían en el primer nivel de la jerarquía, si es la segunda iteración los elementos en P se ubicarían en el segundo nivel de la jerarquía, de esta manera hasta la última iteración.

Paso 13:

Eliminar los elementos en P del conjunto A y se vacía P.

Paso 14:

Si A no está vacío volver al Paso 11**,** Sino, Terminar.

Paso 15:

En caso de que quede más de un elemento en algún nivel de la jerarquía, establecer pre-ordenes por nivel, por ejemplo, si quedan 3 elementos en un mismo nivel de la jerarquía se pueden ordenar de mayor a menor de acuerdo a la cantidad de elementos que supere. En caso de tener la misma cantidad de superaciones significa que los dos elementos tienen igual preferencia.

1.1.3 Agregación de Rankings

En el ámbito de la toma de decisiones no existe, en general ninguna solución que sea la mejor simultáneamente desde todos los puntos de vista, por lo que el resultado que se persigue con estos métodos de solución es obtener un ranking[2] que comprenda las alternativas que se están analizando.

Los algoritmos genéticos han sido bastante utilizados en la agregación de rankings sin duda alguna. Trabajan manteniendo una población de soluciones o individuos las cuales evolucionan de acuerdo a los principios de selección natural. Estos comienzan con una población de individuos generada aleatoriamente por lo general, luego un proceso de optimización estocástico es llevado a cabo, donde los individuos mejores de acuerdo a una función objetivo son preferidos y la población descendiente es obtenida aplicando tres principales operadores genéticos: selección, cruce y mutación (Aledo et al., 2013).

Biblioteca JGAP

JGAP son las siglas de Java Genetic Algorithms Package (paquete de algoritmos genéticos para Java). Es un componente de programación de algoritmos genéticos que se utiliza como un framework. Fue diseñado para que fuera muy fácil de usar, es altamente modular y configurable, con esta biblioteca es fácil crear nuevos y personalizados operadores genéticos (MICHAY, 2011). Se trata de una solución genérica, sin relación alguna con un problema particular. Por esa razón se deben crear nuevas clases que heredan o implementan las clases e interfaces, así se adapta JGAP al problema específico que se quiere solucionar.

[2] Relación entre un conjunto de elementos tales que, para uno o varios criterios, el primero de ellos presenta un valor superior al segundo, este a su vez mayor que el tercero y así sucesivamente.

Se utiliza para la agregación de rankings un algoritmo genético propuesto por (Aledo et al., 2013), el cual se expone a continuación.

Dado un conjunto de N rankings de n elementos, para identificar el ranking que mejor representa a este conjunto es importante poder medirlos y compararlos entre sí, el cálculo de una distancia es una herramienta muy utilizada para esto. Aunque en la literatura existen varias funciones para el cálculo de distancias, la distancia de Kendall es muy usada en el ámbito de agregación de rankings.

Distancia de Kendall: La distancia $d(r_0, r_1)$ entre dos rankings r_0 y r_1 es definida por el total de pares (i,j) en que difieran los órdenes relativos a los respectivos rankings (Aledo et al., 2013).

Ejemplo:

Para los individuos r_0y r_1, dos rankings hipotéticos:

r_0

1	2	3	4	5	6

r_1

2	1	3	5	4	6

Luego: $d(r_0, r_1) = 4$

Utilizando la distancia de Kendall, la función objetivo para medir cuan bueno es un individuo de la población, la cual nos permite hacer comparaciones entre diferentes individuos de la población.

Función objetivo: $min: f(r) \frac{1}{N} \sum_{i=1}^{N} d(r_i, r)$

Operador de cruce basado en la posición (POS): Selecciona aleatoriamente un conjunto de posiciones los valores en estas posiciones son mantenidos en ambos individuos, las posiciones restantes son llenadas usando el orden relativo del otro individuo.

Ejemplo, Dado los dos siguientes individuos:

r_0

1	2	3	4	5	6

r_1

6	1	3	5	4	2

Asumiendo que las posiciones 2,3,5 son seleccionadas queda de esta manera:

r_0

*	2	3	*	5	*

r_1

*	1	3	*	4	*

Mientras que los ítems que no son usados tienen el siguiente orden en el otro individuo:

(1,4,6) → (6,1,4) para r_0, y

(6,5,2) → (2,5,6) para r_1,

Luego se llenan las posiciones vacías con el nuevo orden:

r_0

6	2	3	1	5	4

r_1

2	1	3	5	4	6

Operador de mutación por inserción (ISM): Aleatoriamente selecciona un elemento, el cual es removido de su posición e insertado en otra posición escogida aleatoriamente también.

r_0

6	2	3	1	5	4

r_0

6	2	1	5	3	4

1.2 Tendencias actuales

1.2.1 Análisis crítico de sistemas homólogos

A continuación y a partir de la revisión documental se realiza un breve análisis de algunos sistemas utilizados en la actualidad para la ayuda a la toma de decisiones. Es necesario destacar que se analizan varios sistemas sin embargo no fue posible encontrar abundante documentación en todos los casos, esto obedece a que son sistemas propietarios. Este análisis se centra fundamentalmente en los indicadores:

- métodos que utiliza
- el tipo de licencia que ofrece
- costo
- extensibilidad.

***1000Minds*:**

1000Minds es un software en línea de apoyo a la toma de decisión desarrollado por *100Minds Ltd* en el año 2002. Este sistema está disponible en el idioma inglés solamente y es una

herramienta con licencia privativa que implementa el método PAPRIKA[3] (McGinley, 2012). Se puede obtener gratis para usos académicos.

***Criterium DecisionPlus*:**

Criterium DeicisionPlus es un DSS basado en la toma de decisiones con múltiples criterios creado por *InfoHarvest Inc*. Es un software que implementa el método AHP y el método SMART. Su licencia es privativa (Haerer, 2000), con un costo de $895.00.

***D-Sight*:**

D-Sight es un software en línea de apoyo a la toma de decisión desarrollado por *D-Sight Inc* en Febrero del 2010. Este sistema está disponible en el idioma inglés solamente y es una herramienta con licencia privativa que implementa el método PROMETHEE y el método GAIA. Este DSS es muy usado en la administración de energía y medio ambiente (McGinley, 2012). Sus costos son de 249€ en su licencia académica y de 1990€ la comercial.

***DecideIT*:**

DecideIT es un software en línea de apoyo a la toma de decisión desarrollado por *Preference AB*. Este sistema implementa el método Delta MCDM y es usado en procesos de toma de decisión con múltiples criterios (McGinley, 2012). Gratis para usos académicos y para el uso commercial 1900€ + 900€/año

***Decision Lens*:**

Decision Lens es un software de apoyo a la toma de decisión desarrollado por *Decision Lens Inc*, una compañía fundada en el año 2002 por los hermanos John y Dan Saaty. Este DSS implementa los métodos AHP y ANP y es usado en procesos de toma de decisión con múltiples expertos.

***Expert Choice*:**

Expert Choice es un sistema de apoyo a la toma de decisión desarrollado por Thomas Saaty

[3] Del ingles *Potentially all pairwise rankings of all possible alternatives*

y Ernest Forman en 1983. Este DSS implementa el método AHP, y es usado en la administración del medio ambiente y la agricultura. Su licencia es privativa (McGinley, 2012).

***PriEsT*:**

PriEsT, acrónimo de *Priority Estimation Tool*, es un sistema de apoyo a la toma de decisión multiplataforma desarrollado para software libre. Este DSS implementa el método AHP, y es utilizado en numerosos campos como la salud, transportación, telecomunicaciones y otros (Siraj et al., 2013).

Además de los sistemas anteriores en la UCI se han desarrollado los siguientes sistemas:

MultiDecision PAAT:

Este sistema, desarrollado en la Universidad de las Ciencias Informáticas en el año 2013, es una herramienta web que implementa los métodos PROMETHEE, AHP, ANP y TOPSIS. No está diseñado para problemas que presenten incertidumbre, y no permite la incorporación de nuevos métodos de solución de problemas de toma de decisión (Peña Táramo, 2013),(Torres Estol and Padrón Del Pico, 2013) ,(Sánchez González and González Landeiro, 2013).

Apolo:

Este sistema de soporte a la decisión, desarrollado en la Universidad de las Ciencias Informáticas en el año 2012, es una herramienta web con un fin específico: la Selección de Personal en el campo de los Recursos Humanos específicamente la ubicación de estudiantes en un rol del proyecto (Arza Pérez and Borrego León, 2012).

A continuación se exponen comparaciones de acuerdo a distintos criterios de estos sistemas:

Sistema	Licencia	Métodos	Precio	Extensibilidad
1000Minds	Privada	PAPRIKA	Gratis para usos académicos	No encontrado
Criterium DecisionPlus	Privada	AHP y SMART	$895.00	No encontrado
D-Sight	Privada	PROMETHEE y GAIA	Académica 249€, Comercial 1990€	No encontrado
DecideIT	Privada	Delta MCDM	Gratis para usos académicos, Commercial 1900€ + 900€/año	No encontrado

- Fragmento de tabla "Soporte al proceso de toma de decisiones" extraído de (Mustajoki and Marttunen, 2013):

Sistema	Software de propósito general	Soporte del proceso	Guía al usuario	Nivel de experiencia requerida
1000Minds	Si	Si	Si	2
Criterium DecisionPlus	Si	No	No	3
D-Sight	Si	Si	No	2
DecideIT	Si	No	No	3

Tabla 1.1: Soporte al proceso de toma de decisiones

- Fragmento de tabla "Construcción del modelo" extraído de (Mustajoki and Marttunen, 2013):

Sistema	Modelo Jerárquico	Modelo tabular	Puntaje visual	Modelo Excel
1000Minds	No	Si	No	No
Criterium DecisionPlus	Si	Si	Si	Si
D-Sight	Si	Si	No	Si
DecideIT	Si	No	Si	No

Tabla 1.2: Construcción del modelo

- Fragmento de tabla "Análisis de los resultados" extraído de (Mustajoki and Marttunen, 2013):

Sistema	Gráficas	Valores en general	Análisis de sensibilidad	Reportes
1000Minds	Si	Si	Si	Si
Criterium DecisionPlus	Si	Si	Si	No
D-Sight	Si	Si	Si	No
DecideIT	Si	Si	Si	No

Tabla 1.3: Análisis de los resultados

1.2.2 Patrones de diseño

Los patrones de diseño juegan un papel fundamental en el desarrollo de software en la actualidad, deben estar presentes en todo código en cualquier lenguaje de programación. Estos agilizan el proceso de desarrollo, enriquecen y esclarecen el código escrito, además de facilitar su posterior mantención y reutilización. Con su uso el software resultante es capaz de adaptarse a nuevos requisitos y extenderse con nuevas funcionalidades que resuelvan nuevos problemas.

Un patrón de diseño puede ser caracterizado como "una regla de tres partes que expresa una relación entre un determinado contexto, un problema y una solución" (MICHAY, 2011, Pressman, 2010, Sommerville, 2011). Estos se dividen en tres tipos fundamentales:

- **Patrones creacionales**: Relacionados con la instanciación de objetos y brindan una manera de desacoplar al cliente del objeto que necesita instanciar.
- **Patrones de comportamiento**: Relacionados con la manera en que las clases y objetos interactúan y distribuyen responsabilidades.
- **Patrones estructurales**: Permiten componer clases u objetos en estructuras más grandes.

1.2.2.1 Patrón Strategy

El patrón Strategy define una familia de algoritmos, los encapsula y los hace intercambiables. Permite que el algoritmo varíe independientemente del cliente que lo utiliza(Fowler et al., 2002, Buschman et al., 2007).

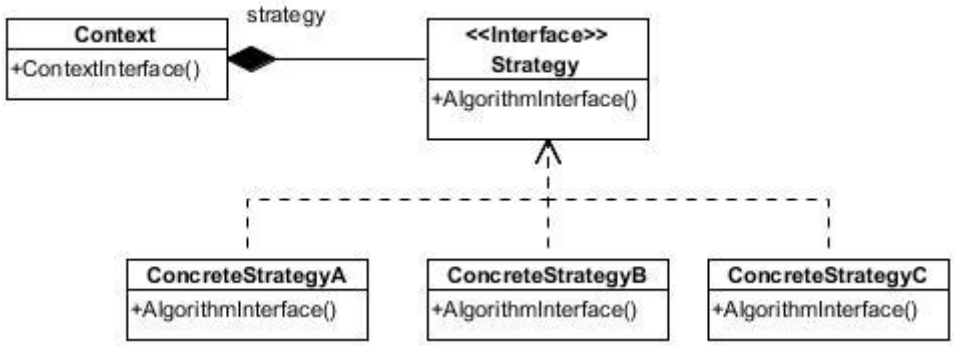

Figura 1.9: Patrón Strategy

1.2.2.2 Patrón Command

El patrón Command encapsula una llamada como un objeto(Fowler et al., 2002, Buschman et al., 2007).

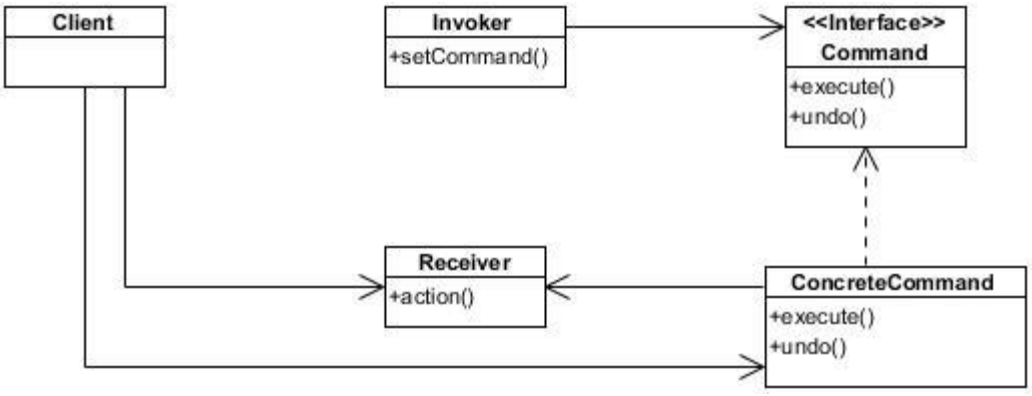

Figura 1.10: Patrón Command

1.2.2.3 Patrón PlugIn

El patrón PlugIn consiste en cargar y ensamblar a una aplicación implementaciones de una interfaz común en tiempo de ejecución (Tech's, 2013).

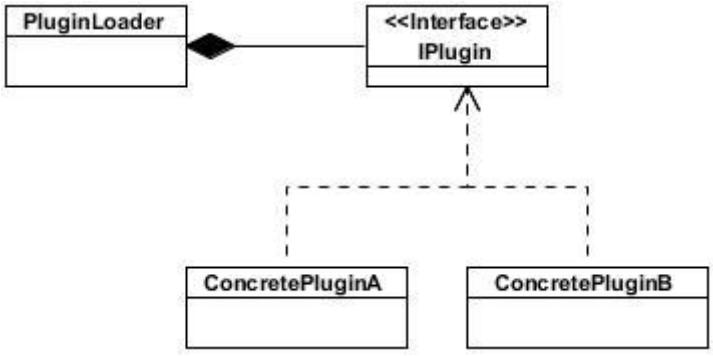

Figura 1.11: Patrón PlugIn

1.3 Fundamentación del Entorno de Desarrollo, Lenguaje de Programación y Tecnologías utilizadas

Para el correcto desarrollo de un sistema informático es muy importante definir las herramientas, lenguajes y metodologías que se utilizarán, por lo que se debe hacer un análisis de las que se emplearán en la solución propuesta.

1.3.1 Herramienta CASE

Las herramientas CASE4 constituyen un conjunto de programas que brindan a los analistas, ingenieros de software y desarrolladores una ayuda durante el ciclo de desarrollo de un software.

La herramienta CASE seleccionada es Visual Paradigm en su versión 8.0. Esta herramienta permite la integración con el entorno de desarrollo que será posteriormente utilizado en la fase de implementación y la generación de código en el lenguaje Java a partir de los diagramas confeccionados. Se tomó en cuenta también que se puede adquirir mediante licencia gratuita y comercial, además de ser muy fácil de instalar y es la herramienta CASE por defecto que se utiliza en la Universidad.

[4] Del ingles *Computer Aided Software Engineering*

1.3.2 Lenguaje de Modelado

El Lenguaje Unificado de Modelado (UML5) es un lenguaje utilizado para especificar, visualizar, construir y documentar los artefactos de los sistemas de software, así como para el modelado del negocio. UML se centra en la representación gráfica del sistema, lo que facilita la comprensión del mismo al proporcionar un vocabulario estándar para modelar sistemas orientados a objetos. Está compuesto por tres clases de bloques de construcción: los elementos que representan abstracciones de cosas reales o ficticias como son objetos o acciones; las relaciones que vinculan los elementos entre sí como son la asociación y la generalización; y finalmente están los diagramas que son las colecciones de elementos y sus relaciones (Booch et al., 2007). Se utilizará UML como notación al ser un estándar que contiene los elementos necesarios para modelar sistemas informáticos.

1.3.3 Lenguaje de programación

Java es un lenguaje de programación de alto nivel orientado a objetos desarrollado por Sun Microsystems a principios de los años 90, y desde sus inicios se diseñó para ser un lenguaje independiente de la plataforma y un entorno de ejecución ligero y gratuito para las plataformas más populares de forma que los códigos binarios de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma. Java contiene muchas bibliotecas de clases que brindan al programador funcionalidades muy útiles. Uno de los elementos más importantes de Java es su característica de ser ejecutable en los sistemas que contienen una implementación de su máquina virtual, lo que lo hace multiplataforma. En la actualidad existen versiones de la Máquina Virtual de Java en plataformas como Windows y Linux, lo que facilita la ejecución de los programas escritos en Java en estos sistemas operativos. Por estas razones se seleccionó este lenguaje de programación para desarrollar el sistema.

1.3.4 Entorno de desarrollo integrado

Un entorno de desarrollo integrado o IDE6 es un entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Un IDE provee un marco de trabajo amigable para la mayoría de los lenguajes de programación.

[5] Del ingles *Unified Modeling Language*
[6] Del ingles *Integrated Development Environment*

Para el desarrollo del sistema se utilizó NetBeans en su versión 7.2.1 como IDE. NetBeans es un IDE escrito en Java, de código abierto, gratuito para desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones profesionales, empresariales, web y móviles con los lenguajes Java, JavaFX, C/C++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. NetBeans es fácil de instalar y se puede ejecutar en los sistemas operativos Windows, Linux, Mac OS X y Solaris. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio de gran tamaño.

1.4 Conclusiones parciales

Después de analizar los aspectos teóricos que intervienen en un problema de toma de decisiones con múltiples criterios se logró un mejor entendimiento de los elementos necesarios para modelar el problema en cuestión llegando a las siguientes conclusiones parciales:

1. Los principales métodos encontrados en la literatura son AHP, ANP, Delphi, TOPSIS y la familia PROMETHEE. La aplicación manual de la mayoría de estos métodos se hace compleja cuando se manejan varios criterios y varios expertos.
2. El análisis de algunos Sistemas de Soporte a la Decisión arroja como principales desventajas que la mayoría de estos son herramientas con licencias privativas, tienen elevados costos en el mercado y carecen de mecanismos para la incorporación de nuevos métodos sin tener que realizar cambios significativos en la arquitectura del software.
3. Con el uso de algunos patrones de diseño se podría obtener una plataforma extensible y flexible.

“Modelo del negocio y requisitos”.

2. Modelo del Negocio y Requisitos

Este capítulo contiene los principales elementos que caracterizan la propuesta de solución. Se explica el modelo de negocio, se establecen las reglas de negocio y se especifican las características y funcionalidades del sistema a partir de los requisitos funcionales y no funcionales identificados. Además se describen los casos de uso del sistema más significativos.

2.1 Modelo del negocio actual

Para modelar un problema de toma de decisiones con múltiples criterios se deben identificar los siguientes elementos:

- Los dominios (D) de acotación para los criterios.
- Los criterios (C) que van a caracterizar a las alternativas.
- Los pesos (W) asociados a los criterios.
- Las alternativas (A) entre las que se quiere elegir.
- El/Los expertos (E) que van a evaluar las alternativas, donde cada experto genera una matriz de evaluación.

El cual se debería modelar de la siguiente manera:

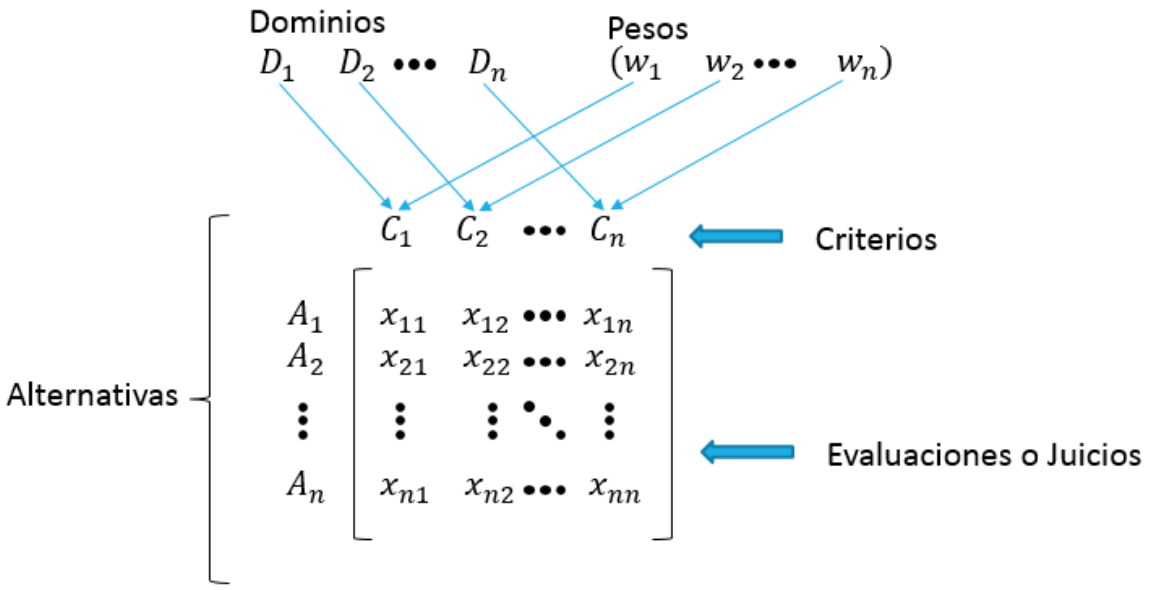

Figura 2.1: Matriz de evaluación

2.2 Reglas del negocio a considerar

- Todas las alternativas tienen que ser evaluadas por todos los criterios definidos.
- Las decisiones de un experto no pueden tener prioridad sobre otras decisiones de otros expertos.
- El valor con que se evalúe la alternativa en el criterio tiene que estar en el dominio previamente definido para el criterio.
- El valor con que se evalúe la alternativa en el criterio no puede ser vacío o nulo.
- Un criterio solo puede tener un único dominio definido.
- No pueden existir criterios duplicados.
- No pueden existir alternativas duplicadas.

2.3 Actores del negocio

- **Decisor**: persona responsable de seleccionar la/las mejor/es alternativa/s.
- **Experto**: persona responsable de evaluar las alternativas de acuerdo a los criterios que las describen.

2.4 Diagrama de casos de uso del negocio

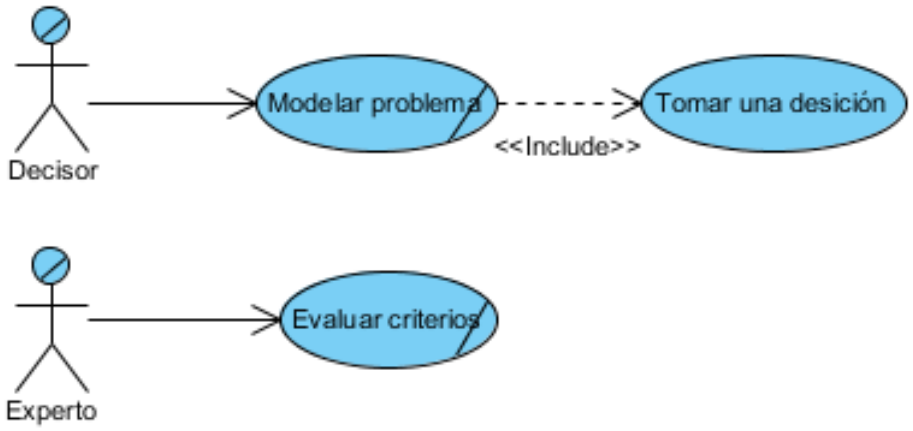

Figura 2.2: Diagrama de casos de uso del negocio

2.5 Casos de uso del negocio

- **Modelar problema**: es donde se definen los elementos fundamentales de problema de toma de decisiones.
- **Evaluar alternativas por criterios**: el experto expresa sus opiniones acerca de las alternativas respecto a los criterios.
- **Tomar una decisión**: una vez evaluada las alternativas se aplica un método y se lleva a cabo la decisión.

2.6 Actores del negocio a automatizar

- **Decisor**: persona responsable de seleccionar la/las mejor/es alternativa/s.

2.7 Requisitos funcionales

RF1	Gestionar proyectos
Descripción	La herramienta tiene permitir crear un proyecto nuevo, salvarlo, abrir y editar un proyecto creado.
Precedencia	CUN_ Modelar problema.
Seguimiento	CUS_ Gestionar proyecto.
RF2	**Gestionar dominios**
Descripción	El sistema tiene que permitir al usuario definir nuevos dominios, modificarlos, eliminarlos y exportar el conjunto de dominios definidos.
Precedencia	CUN_ Modelar problema.
Seguimiento	CUS_ Gestionar dominios.
RF3	**Gestionar criterios**
Descripción	El sistema tiene que permitir al usuario definir nuevos criterios, modificar criterios existentes o eliminarlos.
Precedencia	CUN_ Modelar problema.
Seguimiento	CUS_ Gestionar criterios.
RF4	**Gestionar alternativas**

Descripción	El sistema tiene que permitir al usuario definir nuevas alternativas, modificar alternativas existentes o eliminarlas.
Precedencia	CUN_ Modelar problema.
Seguimiento	CUS_ Gestionar alternativas.
RF5	**Gestionar expertos**
Descripción	El sistema tiene que permitir al usuario definir nuevos expertos o eliminarlos.
Precedencia	-
Seguimiento	CUS_ Gestionar expertos.
RF6	**Gestionar plantilla de experto**
Descripción	El sistema tiene que permitir al usuario definir, editar y exportar a una hoja de cálculo una plantilla de experto la cual será presentada en forma tabular para cada experto que se defina.
Precedencia	-
Seguimiento	CUS_ Gestionar Plantilla de Experto.
RF7	**Importar respuestas experto**
Descripción	El sistema tiene que brindar a posibilidad de que el usuario pueda importar las respuestas de los expertos desde una hoja de cálculo.
Precedencia	CUN_ Evaluar alternativas por criterios.
Seguimiento	CUS_ Importar evaluaciones.
RF8	**Solucionar problema**
Descripción	El sistema tiene que brindar al usuario una solución al problema de toma de decisiones una vez definido todos los aspectos necesarios en la modelación del problema.
Precedencia	CUN_Tomar decisión.
Seguimiento	CUS_ Solucionar problema.
RF9	**Gestión de Pacientes**

Descripción	El sistema debe ofrecer interfaz para los doctores, donde se pueda agregar, eliminar, actualizar y consultar a los pacientes.
Precedencia	CUN_Realizar estudio angiográfico
Seguimiento	CUS_ Gestionar paciente

Tabla 2.1: Requisitos funcionales

2.8 Requisitos no funcionales

2.8.1 Usabilidad

RNF1	Exploración
Descripción	Se incluirán textos emergentes, iconos y etiquetas en los botones que faciliten la comprensión de las acciones y favorezcan la rápida familiarización con el sistema.

Tabla 2.2: Requisito no funcional 1 "Exploración"

2.8.2 Seguridad

RNF2	Ficheros
Descripción	Los ficheros de almacenamiento de los datos de los proyectos, serán ficheros binarios, los cuales no permiten leer su contenido con editores de texto convencionales.

Tabla 2.3: Requisito no funcional 2 "Ficheros"

2.8.3 Rendimiento

RNF3	Tiempo de respuesta
Descripción	El tiempo de respuesta no debe exceder los cinco segundos ante las solicitudes del usuario.

Tabla 2.4: Requisito no funcional 3 "Tiempo de respuesta"

2.8.4 Estandarización

RNF4	Hojas de cálculo
Descripción	Las tablas que se exporten de la herramienta tendrán un formato de hoja de cálculo.

RNF5	Gráficas
Descripción	Los resultados obtenidos en gráficas se podrán exportar a imágenes en formato jpg.

Tabla 2.5: Requisitos no funcionales 4 "Hojas de cálculo" y 5 "Gráficas"

2.8.5 Extensibilidad

RNF6	PlugIns
Descripción	El sistema debe permitir su extensibilidad a través de plugins, garantizando en su diseño que pueda asimilar y absorber los cambios en los requisitos sin ningún contratiempo.

Tabla 2.6: Requisito no funcional 6 "PlugIns"

2.8.6 Software

RNF7	Plataforma
Descripción	En la computadora donde se vaya a ejecutar el sistema tiene que estar instalada la máquina virtual de Java correspondiente para el sistema operativo en uso.

Tabla 2.7: Requisito no funcional 7 "Plataforma"

2.8.7 Ayuda y documentación

RNF8	Documentación y Ayuda
Descripción	Se debe incluir un vínculo a un manual de usuario en el sistema.

Tabla 2.8: Requisito no funcional 7 "Documentación y Ayuda"

2.9 Paquetes y sus relaciones

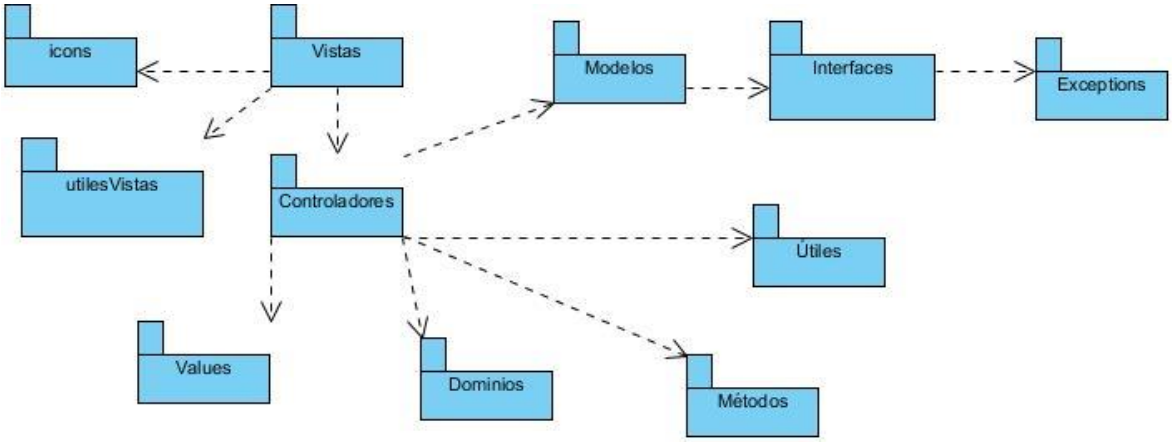

Figura 2.3: Diagrama de paquetes

2.10 Diagrama de casos de uso del sistema (CUS)

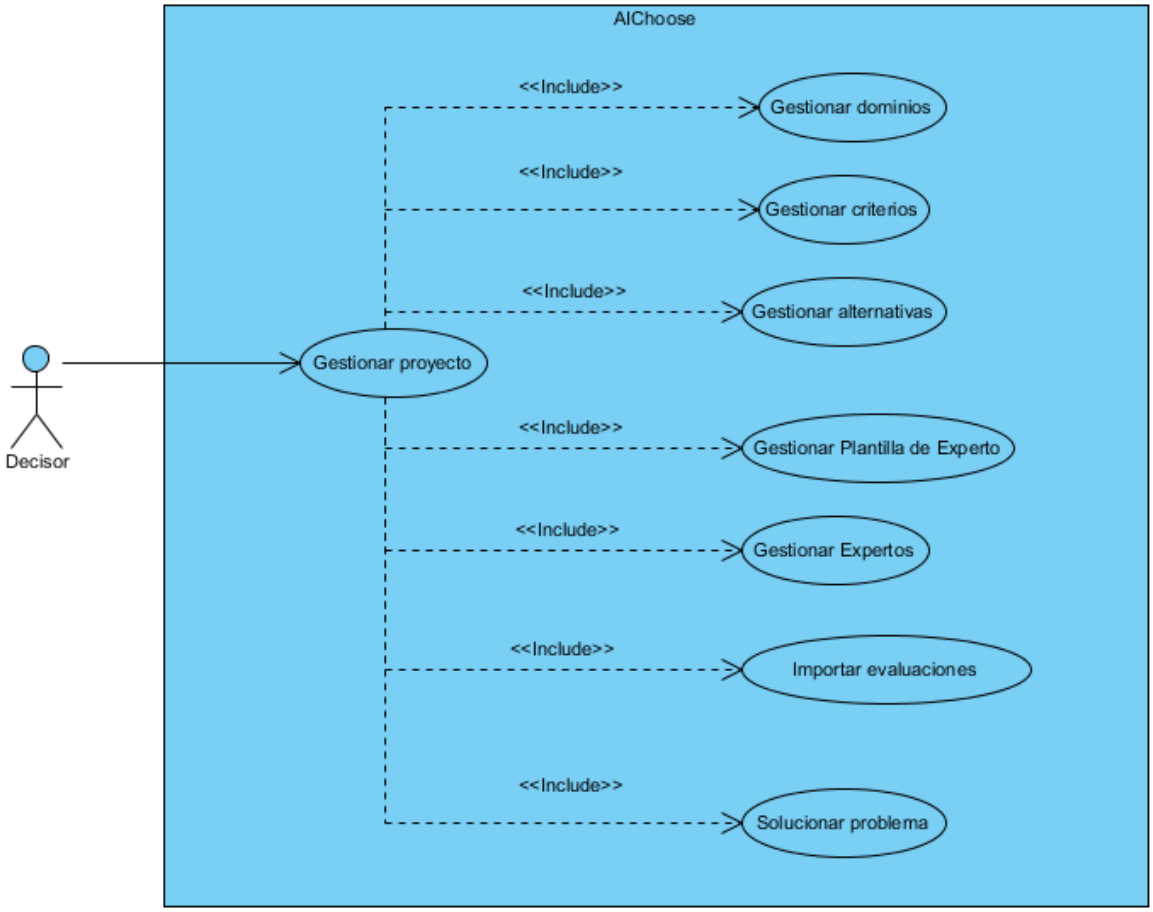

Figura 2.4: Diagrama de casos de uso del sistema

Descripción de los casos de uso del sistema significativos.

2.10.1 Caso de uso del sistema "Gestionar Proyecto"

Caso de uso del sistema	Gestionar Proyecto
Actores	Decisor
Propósito	Definir el problema a resolver.
Resumen	El Decisor define el problema a resolver identificándolo por un título e indicando la URL donde se guardara el archivo del proyecto.
Responsabilidades	Listar, cerrar, abrir, editar o crear proyectos.
Casos de uso asociados	CUS Gestionar dominios, CUS Gestionar criterios, CUS Gestionar alternativas, CUS Gestionar Plantilla de experto, CUS Expertos, CUS Importar evaluaciones, CUS solucionar problema.
Requisitos especiales	-
Precondiciones	-
Descripción	

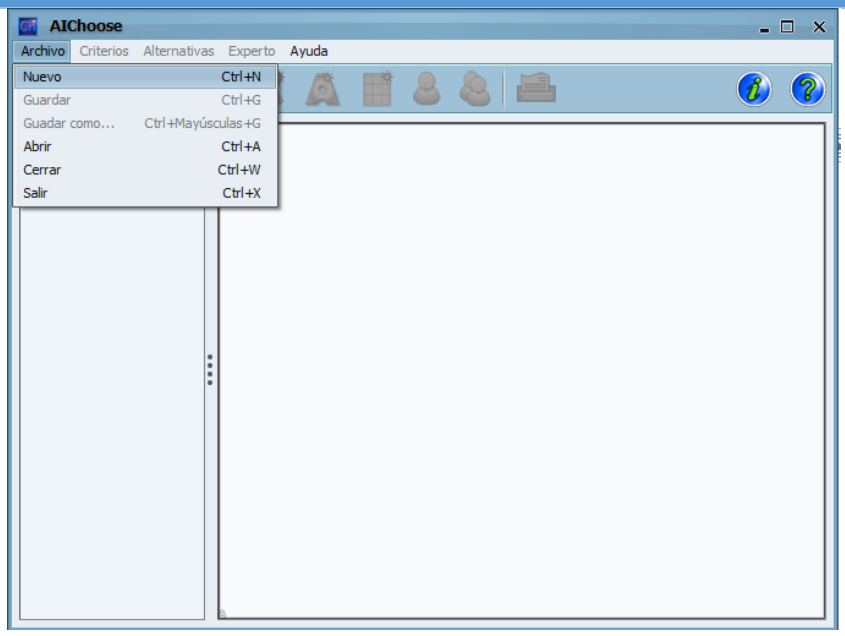

Figura 2.5: Pantalla1 "Nuevo proyecto"

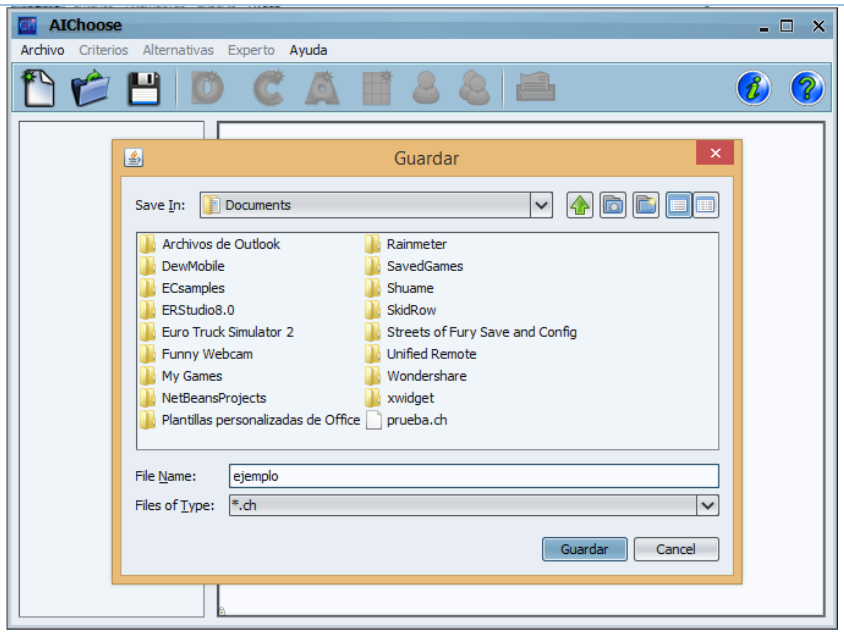

Figura 2.6: Pantalla2 "Guardar proyecto"

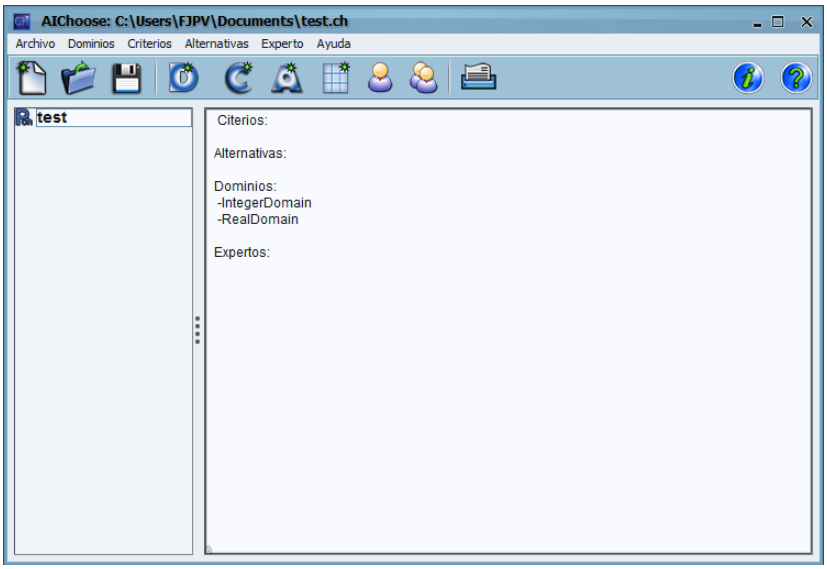

Figura 2.7: Pantalla3 "Proyecto creado"

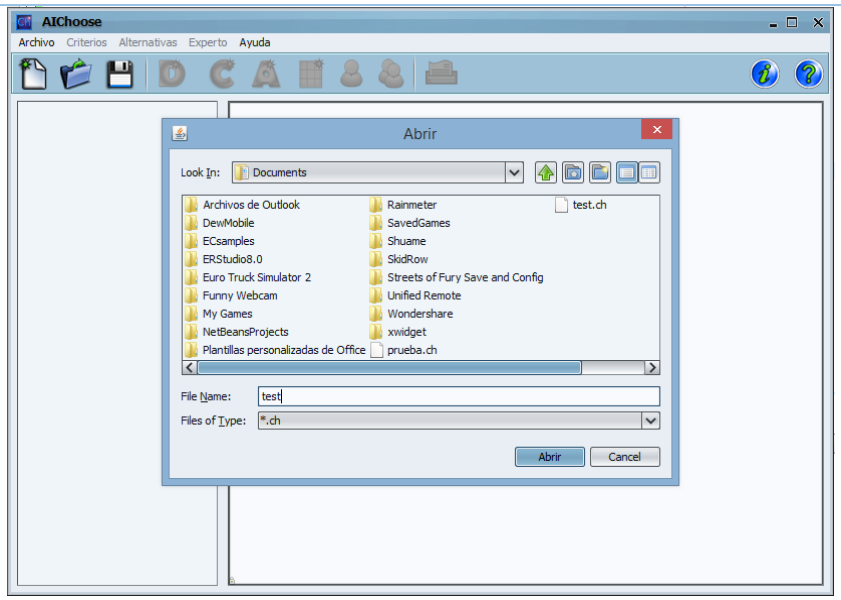

Figura 2.8: Pantalla4 "Abrir proyecto"

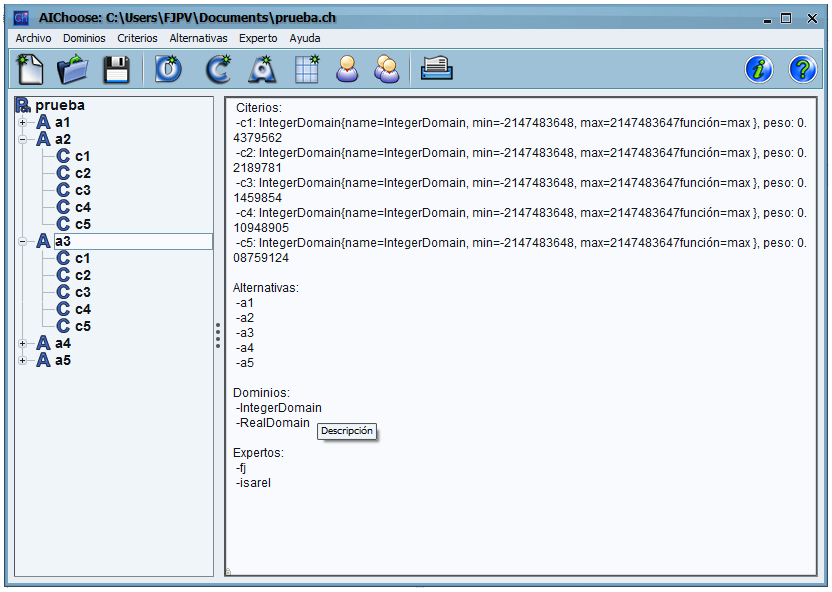

Figura 2.9: Pantalla5 "Proyecto cargado"

Acción del actor	Respuesta del sistema
1. El Decisor elige el menú "Archivo/Nuevo".	
	2. Se muestra la pantalla 2.
3. El Decisor busca la URL donde quiere salvar el	

proyecto y da click en el botón “Guardar”.	4. Se muestra la pantalla 3.
Otras secciones	
Abrir proyecto	
1. El Decisor elige el menú “Archivo/Abrir”. 3. El Decisor busca el proyecto que desea abrir y da click en el botón “Abrir”.	2. Se muestra la pantalla 4. 4. Se muestra la pantalla 5.
Flujos alternos	
Cuando el Decisor elige una URL para salvar el proyecto en la cual no hay espacio suficiente o no tiene permiso de creación.	El sistema muestra una alerta indicando el error adecuado, “Espacio insuficiente” o “Acceso denegado”
Post condiciones	Queda guardado en memoria un fichero que contiene los datos del proyecto.

Tabla 2.9: Especificación CUS "Gestionar Proyecto"

2.10.2 Caso de uso del sistema “Gestionar dominios”

Caso de uso del sistema	Gestionar dominios
Actores	Decisor
Propósito	Definir los dominios a utilizar en el proyecto.

Resumen	El Decisor define los dominios a utilizar en el proyecto por los criterios que se definan posteriormente. Estos dominios podrán ser exportados como un paquete o importados hacia el proyecto.
Responsabilidades	Listar, crear, eliminar, importar o exportar dominios.
Casos de uso asociados	-
Requisitos especiales	-
Precondiciones	El sistema debe contar con un conjunto de PlugIns de dominio cargados en tiempo de ejecución que permitirán que el Decisor pueda crear nuevos dominios, localizados en una carpeta llamada "domains" que tiene que estar en la misma carpeta del sistema.

Descripción

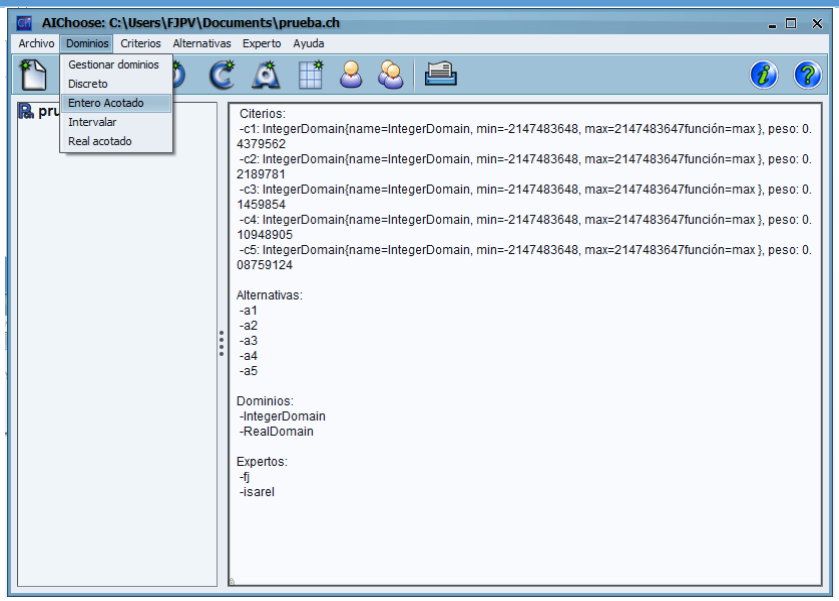

Figura 2.10: Pantalla1 "Vista principal"

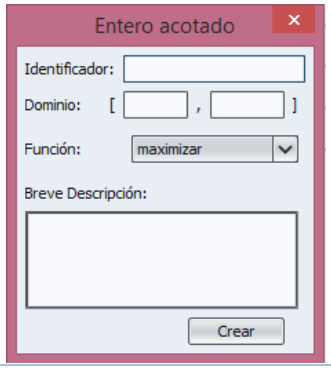

Figura 2.11: Pantalla 2 "Nuevo dominio Entero acotado"

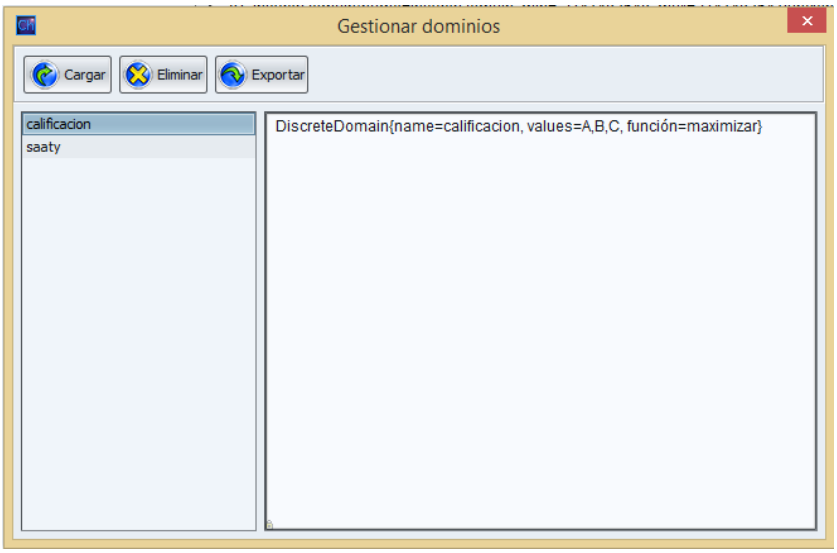

Figura 2.12: Pantalla 3 "Gestionar dominios"

Acción del actor	Respuesta del sistema
1. En la pantalla 1, de la lista de PlugIns de dominio disponible el Decisor elige uno, en este caso "Entero acotado".	
	2. Se muestra la pantalla 2, la cual se va a encargar de recoger los datos del dominio (cada dominio recoge diferentes datos).
3. El Decisor hace click en el botón "Crear".	
	4. El sistema crea el dominio.
Otras secciones	
Gestionar dominios	
1. En la pantalla 1, el Decisor elige la opción del menú "Dominios/Gestionar dominios"	
	2. Se muestra la pantalla 3, que visualiza una lista de los dominios creados.

3. El Decisor hace click en el botón “Cargar”.	4. Se muestra una ventana para buscar la localización de los dominios y cargarlos.
Flujos alternos	
3. El Decisor hace click en el botón “Eliminar”. 3. El Decisor hace click en el botón “Exportar”.	4. Eliminar el dominio seleccionado. 4. Se muestra una ventana para buscar la donde se quiere salvar los dominios.
Post condiciones	Quedan definidos los dominios que se utilizaran posteriormente en la definición de los criterios.

Tabla 2.10: Especificación CUS "Gestionar dominios"

2.10.3 Caso de uso del sistema “Gestionar criterios”

Caso de uso del sistema	Gestionar criterios
Actores	Decisor
Propósito	Definir los criterios a utilizar en el proyecto.
Resumen	El Decisor define los criterios a utilizar en el proyecto que posteriormente caracterizarán a las alternativas.
Responsabilidades	Listar, crear, eliminar o editar criterios.
Casos de uso asociados	-
Requisitos especiales	-
Precondiciones	-
Descripción	

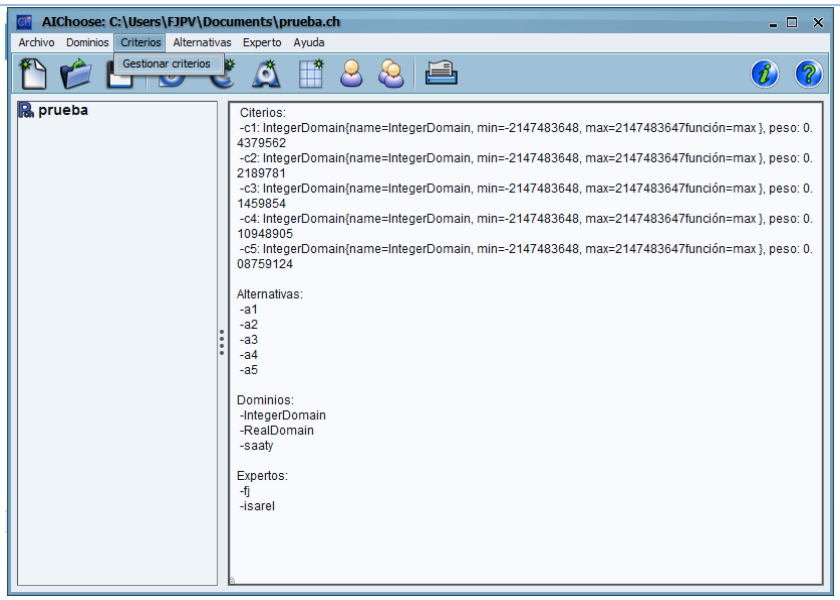

Figura 2.13: Pantalla1 "Vista principal"

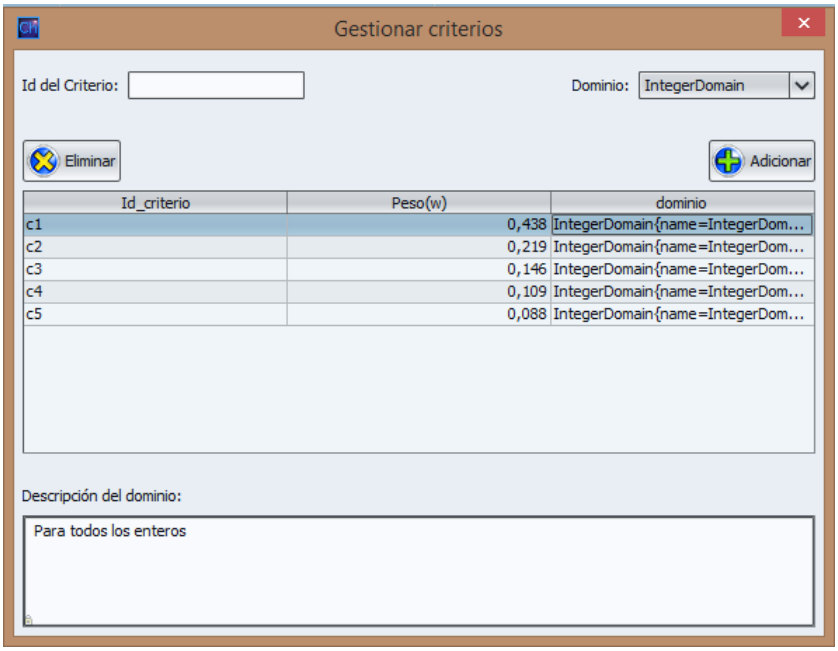

Figura 2.14: Pantalla2 "Gestionar criterios"

Acción del actor	Respuesta del sistema

1. En la pantalla 1, el Decisor elige la opción de menú “Criterios/Gestionar criterios”.	2. Se muestra la pantalla 2, en la cual se realizarán todas las operaciones con los criterios.
3. El Decisor introduce el identificador del criterio, elige el dominio al que pertenece y hace click en el botón “Adicionar”.	4. Se crea el criterio y le asigna un peso en dependencia del orden en que se inserte.
Otras secciones	
Eliminar criterios	
1. En la pantalla 2, el Decisor elige uno o varios criterios y hace click en el botón “Eliminar”.	2. Se eliminan los criterios seleccionados y se restablecen los pesos para los criterios restantes.
Editar criterios	
1. En la pantalla 2, el Decisor elige un criterio y hace doble click en la celda que desea editar. 2. El decisor presiona la tecla “Enter” en el teclado.	3. Se guarda el criterio modificado.
Flujos alternos	
3. El Decisor deja el identificador del criterio	

vacío y hace click en el botón "Adicionar". 3. El Decisor introduce un identificador de criterio existente y hace click en el botón "Adicionar". 3. El Decisor modifica el valor del peso de un criterio existente y presiona "Enter".	4. Se muestra una alerta indicando que el identificador no puede estar vacío. 4. Se muestra una alerta indicando que el identificador no puede estar duplicado. 4. Se verifica que la suma de todos los pesos sume 1. 5. De no ser 1 la suma, se muestra una alerta notificando al Decisor de esto.
Post condiciones	Quedan definidos los criterios que caracterizarán a las alternativas.

Tabla 2.11: Especificación de CUS "Gestionar criterios"

2.10.4 Caso de uso del sistema "Gestionar alternativas"

Caso de uso del sistema	Gestionar alternativas
Actores	Decisor
Propósito	Definir las alternativas de decisión.
Resumen	El Decisor define las alternativas que se quieren comparar.
Responsabilidades	Listar, crear, eliminar o editar alternativas.
Casos de uso asociados	-
Requisitos especiales	-
Precondiciones	Tiene que haber al menos un criterio definido.
Descripción	

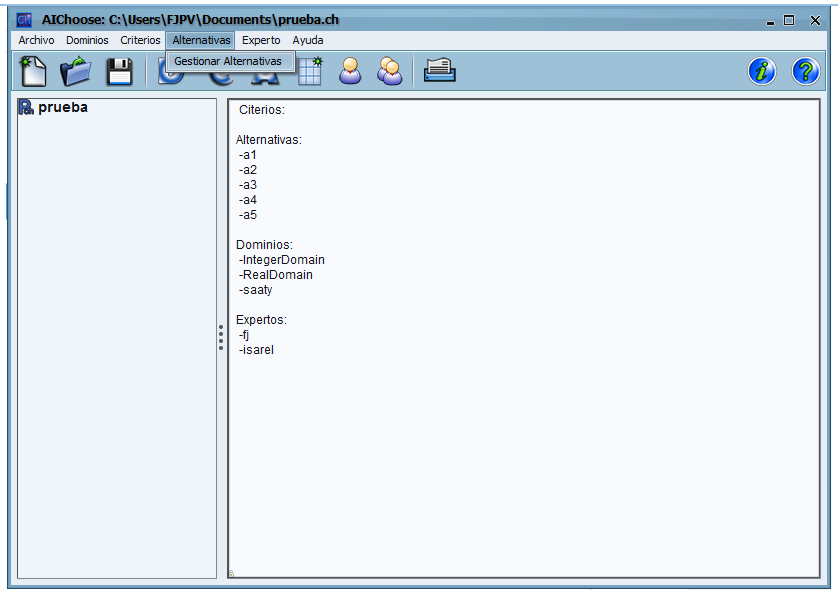

Figura 2.15: Pantalla1 "Vista principal"

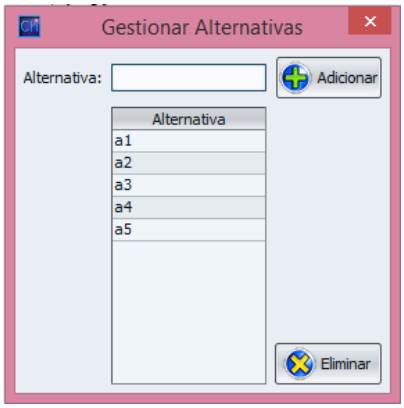

Figura 2.16: Pantalla2 "Gestionar alternativas"

Acción del actor	Respuesta del sistema

1. En la pantalla 1, el Decisor elige la opción de menú "Alternativas/Gestionar alternativas".	2. Se muestra la pantalla 2, en la cual se realizarán todas las operaciones con las alternativas.
3. El Decisor introduce el identificador de la alternativa y hace click en el botón "Adicionar".	4. Se crea la alternativa.
Otras secciones	
Eliminar alternativas	
1. En la pantalla 2, el Decisor elige una o varias alternativas y hace click en el botón "Eliminar".	2. Se eliminan las alternativas seleccionadas.
Editar alternativas	
1. En la pantalla 2, el Decisor elige una alternativa y hace doble click en la celda que desea editar. 2. El decisor presiona la tecla "Enter" en el teclado.	3. Se guarda la alternativa modificada.
Flujos alternos	
3. El Decisor deja el identificador de la alternativa	

vacío y hace click en el botón “Adicionar”. 3. El Decisor introduce un identificador de alternativa existente y hace click en el botón “Adicionar”.	4. Se muestra una alerta indicando que el identificador no puede estar vacío. 4. Se muestra una alerta indicando que el identificador no puede estar duplicado.
Post condiciones	Quedan definidas las alternativas que se quieren comparar.

Tabla 2.12: Especificación CUS "Gestionar alternativas"

2.10.5 Caso de uso del sistema “Resolver problema”

Caso de uso del sistema	Resolver problema
Actores	Decisor
Propósito	Aplicar métodos de solución al problema definido para obtener soluciones factibles.
Resumen	El Decisor elige uno o varios de los PlugIns de métodos de solución y lo aplica al problema definido.
Responsabilidades	Mostrar resultados.
Casos de uso asociados	-
Requisitos especiales	-
Precondiciones	1. El sistema debe contar con un conjunto de PlugIns de métodos de solución cargados en tiempo de ejecución que permitirán que el Decisor pueda elegir cual método aplicar al problema. Los cuales tienen que estar en una carpeta llamada “methods” donde se encuentra localizado el sistema. 2. Tiene que estar bien definido el problema a resolver y sus componentes, dígase dominios, criterios, alternativas, expertos y evaluaciones.
Descripción	

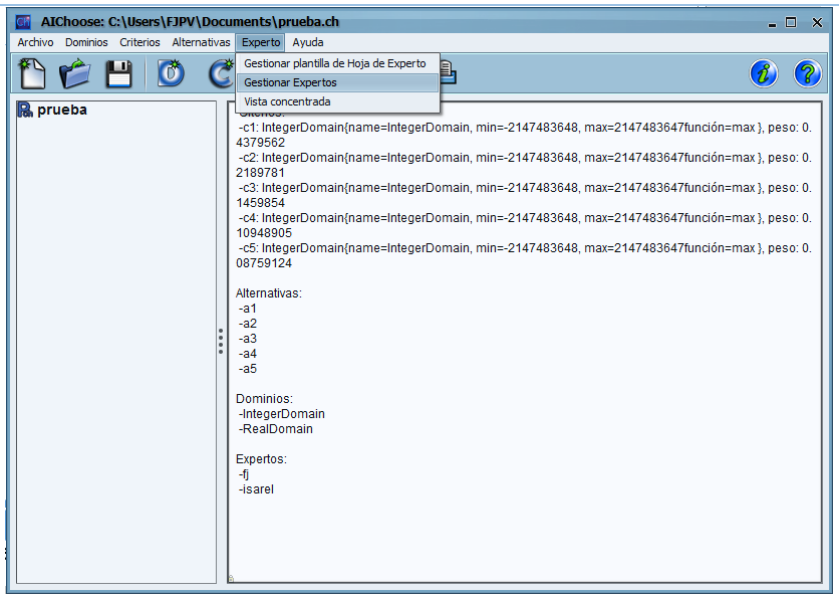

Figura 2.17: Pantalla1 "Vista principal"

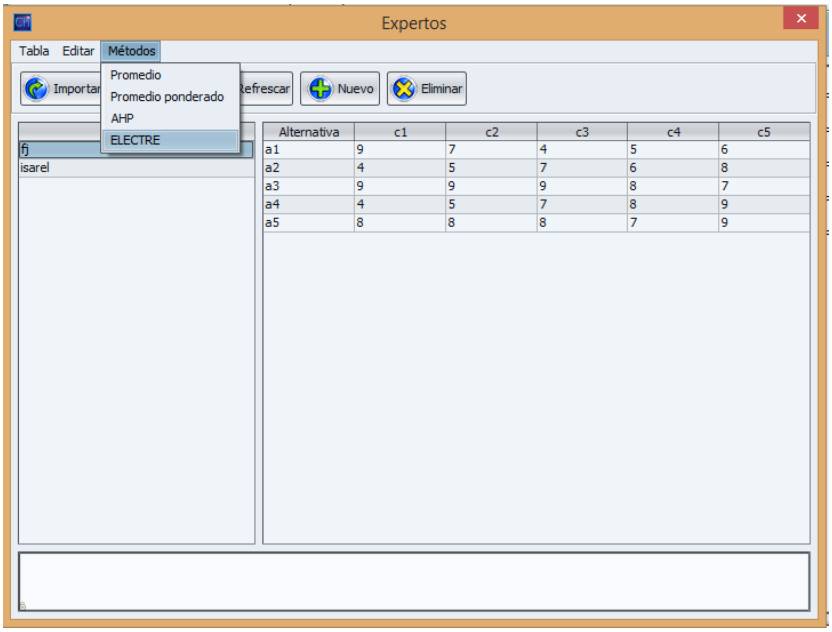

Figura 2.18: Pantalla2 "Métodos de solución"

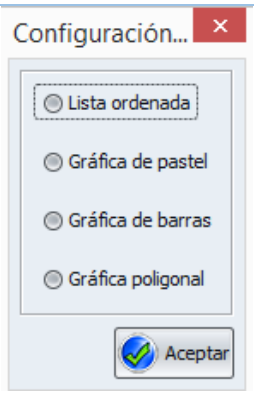

Figura 2.19: Pantalla3 "Configuración de resultados"

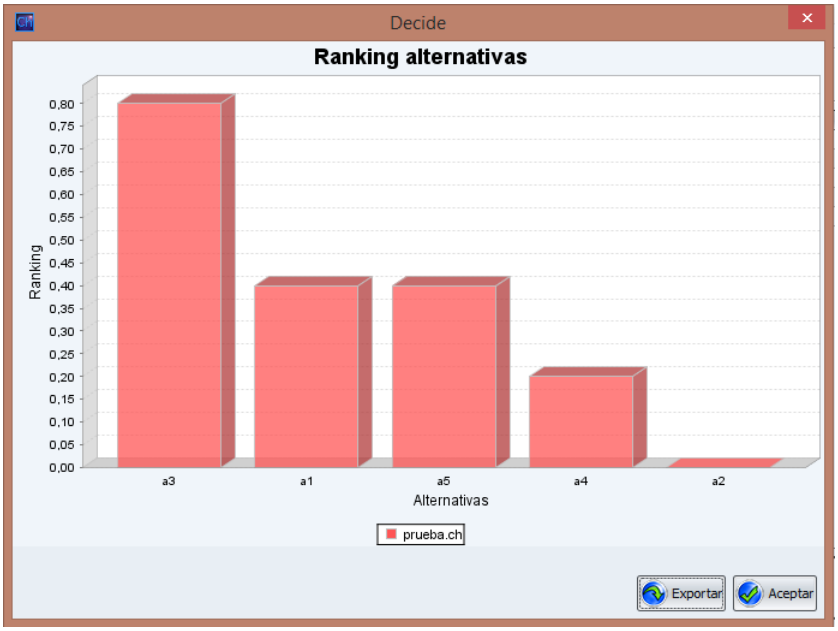

Figura 2.20: Pantalla4 "Vista de resultados"

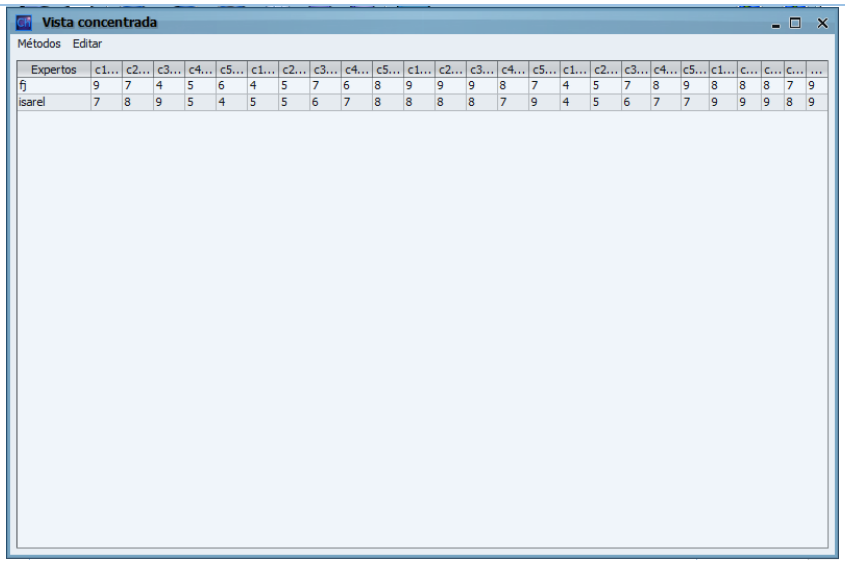

Figura 2.21: Pantalla5 "Vista concentrada de expertos"

Acción del actor	Respuesta del sistema
1. En la pantalla 1, el Decisor elige la opción de menú "Expertos/Gestionar expertos".	2. Se muestra la pantalla 2.
3. El Decisor selecciona un experto de la lista. 4. El Decisor selecciona un método de solución de los que se encuentran disponibles en el menú "Métodos".	4. Se muestra la pantalla 3.
5. El decisor elige la forma en que quiere visualizar los resultados.	

	6. Se muestra la pantalla 4, en dependencia de la opción que seleccione.
Otras secciones	
Varios expertos	
1. En la pantalla 1, el Decisor elige la opción de menú "Expertos/Vista concentrada". 3. Continuar de manera similar desde el paso 3 en sección principal.	2. Se muestra la pantalla 5.
Post condiciones	-

Tabla 2.13: Especificación CUS "Resolver problema"

2.11 Conclusiones parciales

En este capítulo se realizó un análisis del negocio actual, las reglas del negocio, los requisitos fundamentales y los principales casos de uso del sistema para el desarrollo de la herramienta AIChoose, llegando a las siguientes conclusiones parciales:

1. Se utilizó la herramienta CASE Visual Paradigm lo que facilitó la organización en el proceso de desarrollo del sistema, generando todos los diagramas y esquemas necesarios.

2. Se utilizó el Lenguaje de Modelado Unificado (UML) para construir y visualizar los artefactos del sistema y para el modelado del negocio ayudando a la estandarización de los diagramas.
3. Se describieron los casos de uso del sistema más significativos con lo cual quedaron resumidas las principales funcionalidades del sistema.

“Descripción de la propuesta de solución”.

3. Descripción de la propuesta de solución

Este capítulo tiene como objetivo mostrar el proceso de implementación de la plataforma. El capítulo describe la arquitectura del sistema y la construcción de la solución de software propuesta. Se presentan los diagramas de clases del diseño fundamentales y diagramas de secuencia. Se exponen los principios de diseño usados en la implementación, además se explica cómo se llevó a cabo el tratamiento de errores.

3.1 Arquitectura del sistema

- Asimilar los conceptos fundamentales de los sistemas de toma de decisiones para lograr su representación correcta dentro de la herramienta computacional.
- Crear un marco de trabajo donde el usuario pueda modelar cualquier tipo de problema de toma de decisiones que tenga en cuenta uno o varios criterios y que el sistema sea capaz de darle solución de manera automatizada.
- Integrar bajo una misma plataforma varios métodos de solución a problemas de toma de decisiones, desarrollando una herramienta flexible y extensible siguiendo principalmente los patrones de arquitectura Modelo Vista Controlador (MVC) y Arquitectura basada en componentes (CBA).
- Obtener una aplicación de escritorio desarrollada con el lenguaje de programación Java en su edición estándar, permitiendo que la aplicación sea multiplataforma.
- La herramienta está dividida en paquetes que separan las clases por funcionalidad.
- Para lograr una herramienta flexible y extensible, se utiliza la tecnología de plugins.

3.2 Diagramas de clases del diseño

Durante el desarrollo de un software el diseño de sus clases, cómo se relacionan entre sí y cuáles son las mejores formas de representarlas es de suma importancia ya que esto influye en la robustez, flexibilidad, extensibilidad y calidad del propio software. Debido a esto se presentarán una serie de fragmentos fundamentales para el desarrollo de la herramienta del diagrama de clases original, divididos para su mejor comprensión.

3.2.1 Modelo del problema

En este fragmento del diagrama de clases se observa cómo quedará representado en la herramienta computacional un problema de toma de decisiones.

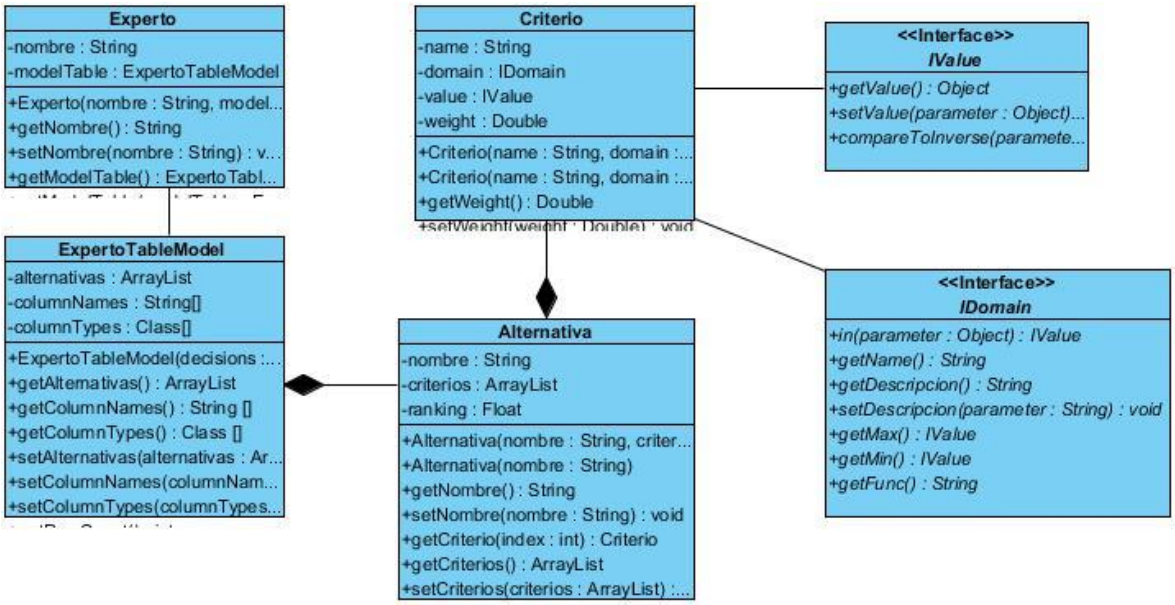

Figura 3.1: Clases del diseño Modelo del problema

3.2.2 Caso de uso del sistema "Gestionar proyecto"

En esta parte del diagrama de clases se puede apreciar en la clase "PersistenceController" el uso del patrón Bridge.

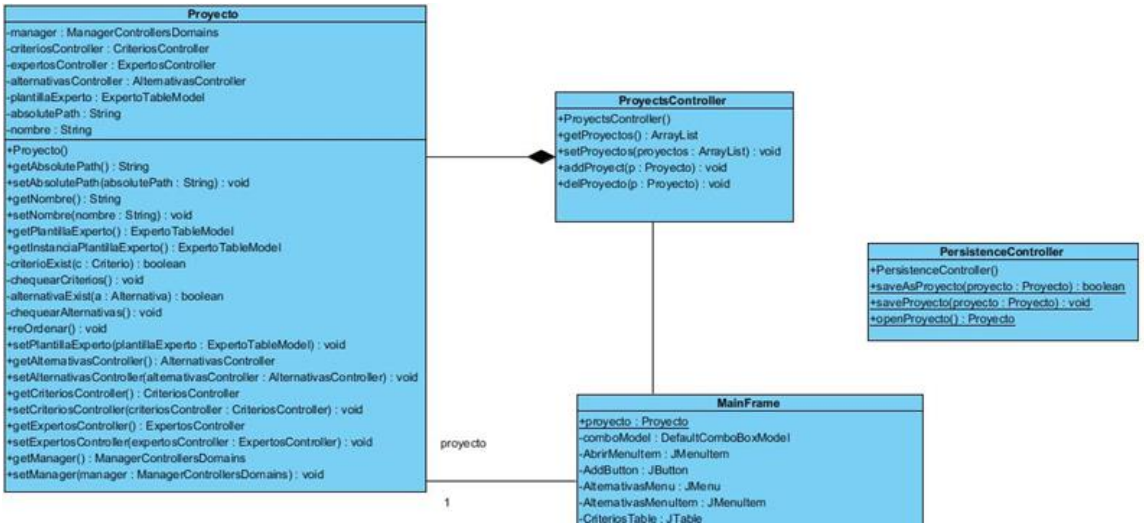

Figura 3.2: Clases del diseño CUS "Gestionar proyecto"

3.2.3 Caso de uso del sistema “Gestionar dominios”

En esta parte del diagrama de clases se puede apreciar cómo quedaría diseñado para el controlador de Dominio para los enteros acotados con las clases “IEDController” y “IEDomain”, para agregar a la plataforma nuevos plugins de dominios solo habría que incorporar nuevas implementaciones de la interfaz “IControllerDomain”.

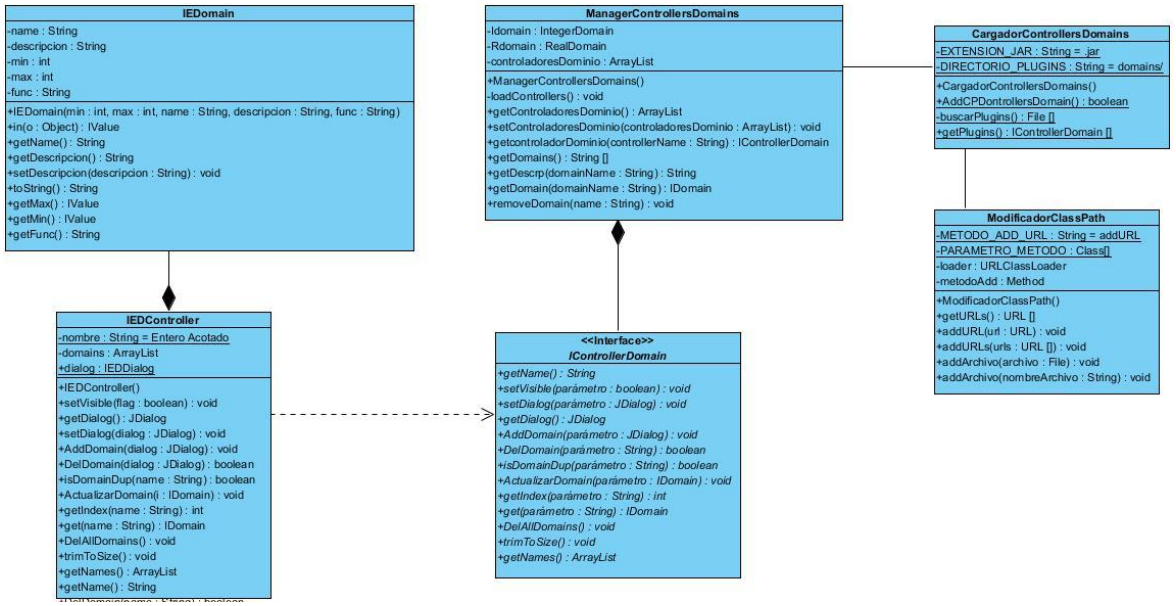

Figura 3.3: Clases del diseño CUS "Gestionar dominios"

3.2.4 Caso de uso del sistema “Gestionar criterios”

En esta parte del diagrama se puede apreciar el uso del patrón Strategy a partir de la interfaz “IValue” que tiene varias implementaciones.

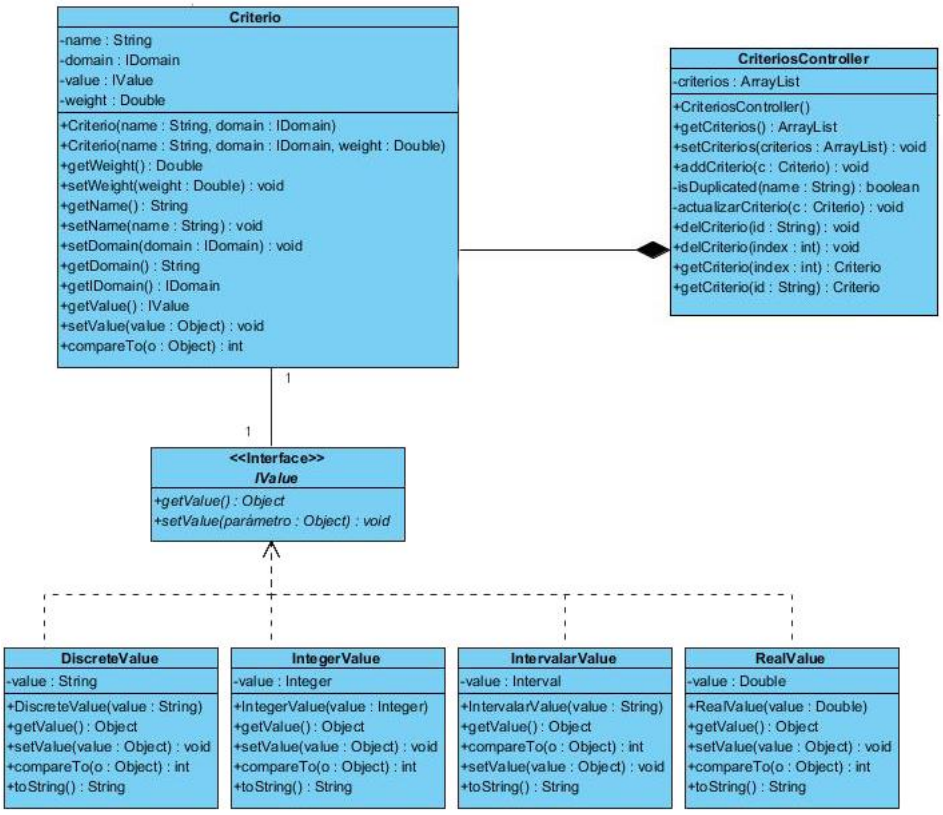

Figura 3.4: Clases del diseño CUS "Gestionar criterios"

3.2.5 Caso de uso del sistema “Gestionar alternativas”

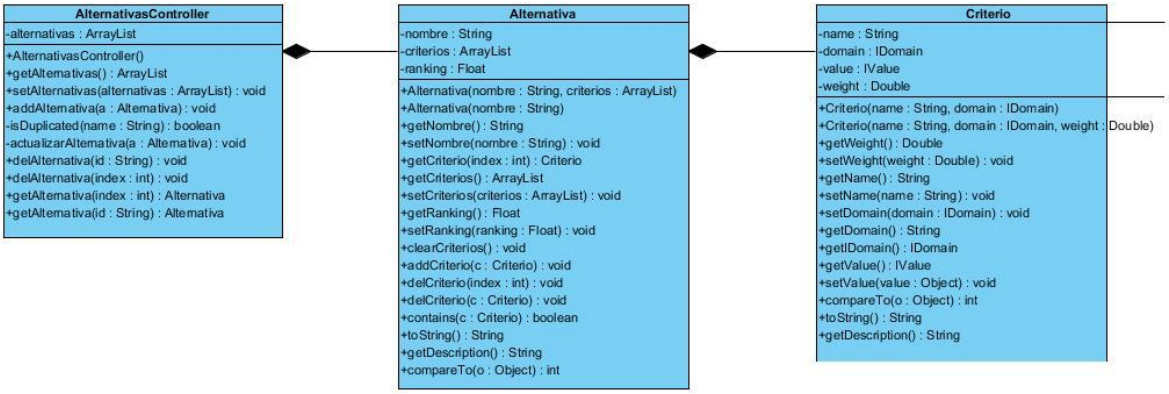

Figura 3.5: Clases del diseño CUS "Gestionar alternativas"

3.2.6 Caso de uso del sistema “Resolver problema”

En esta parte del diagrama de clases se puede apreciar cómo quedaría diseñado para el método de solución AHP, para añadir otro método de solución habría que diseñar una nueva implementación de la interfaz “IMethod”.

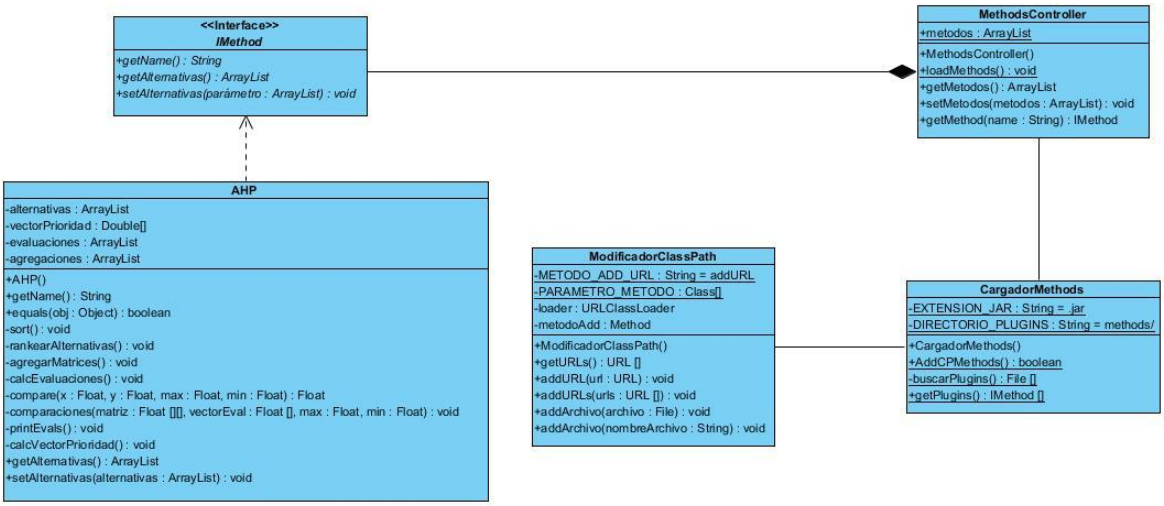

Figura 3.6: Clases del diseño CUS "Resolver problema"

3.3 Diagramas de secuencias

3.3.1 Caso de uso del sistema “Gestionar proyecto”

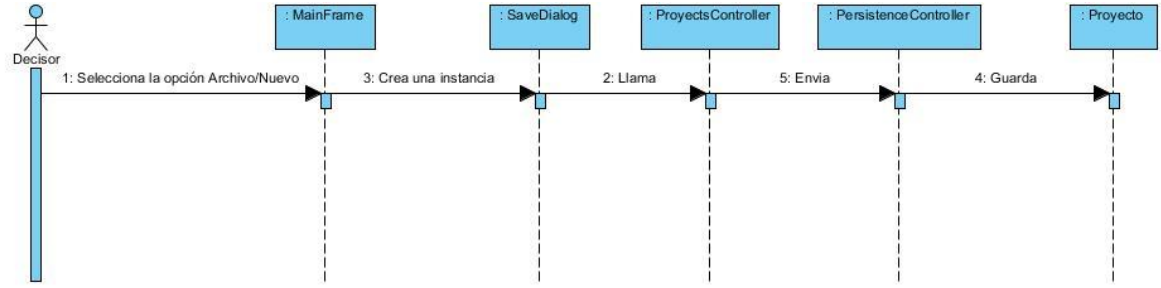

Figura 3.7: Diagrama de secuencia CUS "Gestionar proyecto"

3.3.2 Caso de uso del sistema “Gestionar dominios”

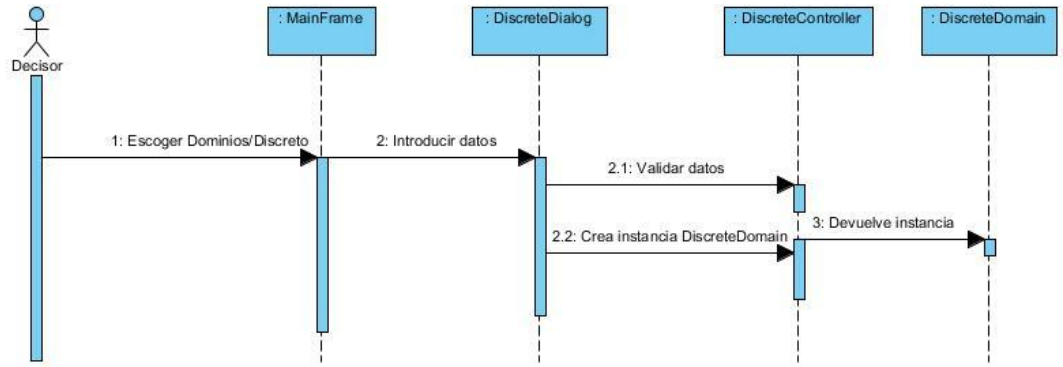

Figura 3.8: Diagrama de secuencia CUS "Gestionar dominios"

3.3.3 Caso de uso del sistema “Gestionar criterios”

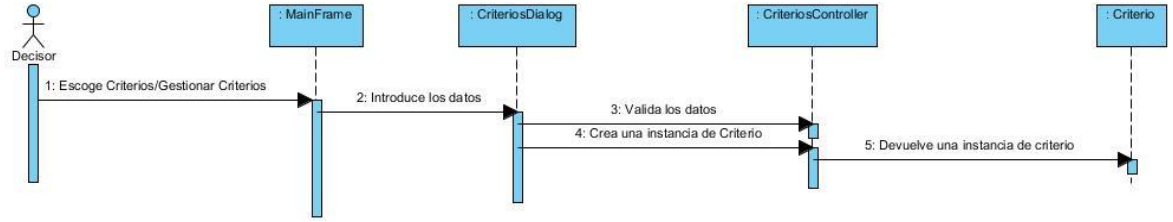

Figura 3.9: Diagrama de secuencia CUS "Gestionar criterios"

3.3.4 Caso de uso del sistema “Gestionar alternativas”

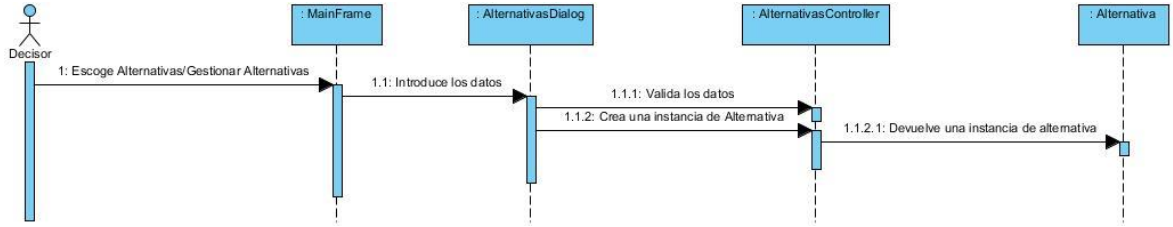

Figura 3.10: Diagrama de secuencia CUS "Gestionar alternativas"

3.3.5 Caso de uso del sistema “Resolver problema”

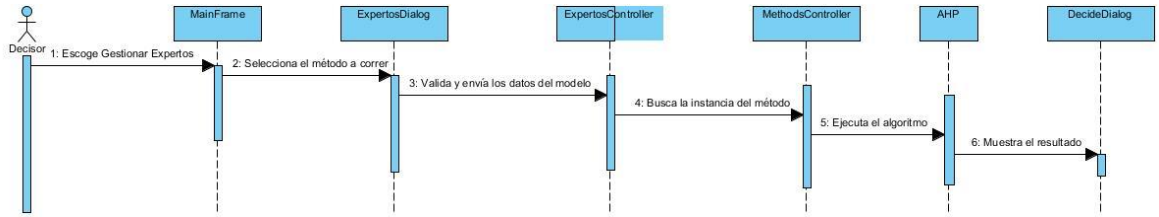

Figura 3.11: Diagrama de secuencia CUS "Resolver problema"

3.4 Principios de diseño

El uso de patrones de diseño asegura la calidad del software, permite que se adapte a cambios en los requisitos del negocio, facilita la mantención y corrección del mismo. De ahí que se utilizaran varios patrones de diseño como:

- **Observador:**

```
gestionarDominios.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        DominiosDialog d = new DominiosDialog(null,true);
        d.setLocationRelativeTo(null);
        d.setVisible(true);
        proyecto.modificado = true;
        descripcionTextArea.setText(proyecto.getDescription());
    }
});
```

Figura 3.12: Fragmento de código "Patrón Observador"

- **Comando:**

```
JMenu m=jMenuBar1.getMenu(2);
for (final IMethod ic : MainFrame.proyecto.getMethodsController().getMetodos()) {
    JMenuItem i = new JMenuItem(ic.getName());
    i.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent evt) {
            String metodo_name=((JMenuItem)evt.getSource()).getText();
            IMethod metodo = MainFrame.proyecto.getMethodsController().getMethod(metodo_name);
            int r = ListaExpertoTable.getSelectedRow();
            if (r > -1) {
                String nombre = ListaExpertoTable.getValueAt(r, 0).toString();
                Experto exp = MainFrame.proyecto.getExpertosController().getExperto(nombre);
                if(exp.validateModel()){
                    metodo.setAlternativas(exp.getModelTable().getAlternativas());
                    SalidaDialog salida = new SalidaDialog(null, true, metodo.getAlternativas());
                    salida.setLocationRelativeTo(null);
                    salida.setVisible(true);
                }else JOptionPane.showMessageDialog(null, "El modelo no puede ser validado, puede que
            }else JOptionPane.showMessageDialog(null, "Debe seleccionar al menos un elemento");
        }
    });
    m.add(i);
}
```

Figura 3.13: Fragmento de código "Patrón Comando"

- **Puente:** Se puede apreciar en el diagrama de clases del diseño del caso de uso "Gestionar Proyecto" sección 3.2.2, en la clase PersistenceController.
- **Plugins:** Se puede apreciar en los diagramas de clases del diseño de los casos de uso "Gestionar dominios" y "Resolver problema" en las secciones 3.2.3 y 3.2.6 respectivamente.

- **Estratégico:** Se puede apreciar en varios diagramas de clases del diseño en la sección 3.2.

3.5 Tratamiento de errores

El tratamiento de errores se manejó a través de la captura de excepciones predefinidas del lenguaje Java en conjunto con otras que se implementaron específicamente para la herramienta, donde se le muestra al usuario una ventana emergente con el mensaje de error para que lo pueda leer y luego seguir trabajando sin ningún tipo de interrupción.

Entre las excepciones implementadas específicamente para la herramienta están:

- ElementoDuplicado
- FormatDomainException
- FormatDomainExecption
- OutOfDomainException
- PlantillaException
- ProyectoVacioException

Entre las excepciones predefinidas de Java que se utilizaron están:

- NullPointerException
- Exception
- BrowserLaunchingInitializingException
- UnsupportedOperatingSystemException
- MalformedURLException
- ServiceConfigurationError
- HeadlessException
- IOException
- WriteException
- BiffException

3.6 Diagrama de componentes

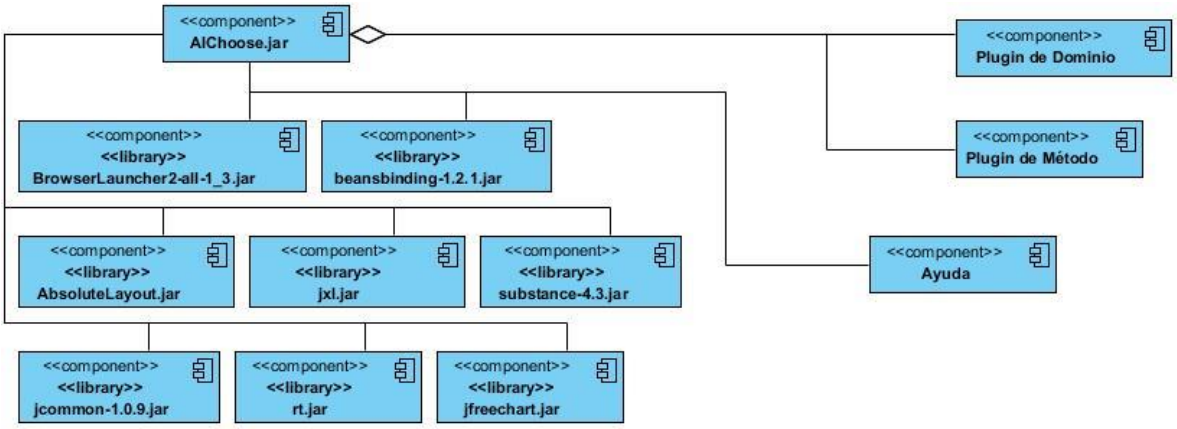

Figura 3.14: Diagrama de componentes

3.7 Plugins en la herramienta computacional

Una vez concluido el desarrollo de la herramienta se extenderá con la tecnología de plugins, en este caso, plugins de dominios y métodos de solución.

Para el desarrollo e incorporación de nuevos plugins de métodos de solución a la plataforma hay que seguir los siguientes pasos:

1. Una vez creado el proyecto de Java agregar la plataforma (solo el archivo *.jar) como biblioteca del proyecto.

2. Crear la siguiente estructura de paquetes en el proyecto:

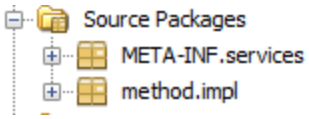

Figura 3.15: Estructura de paquetes para un plugin de método

3. Dentro del paquete "**META-INF.services**" crear un archivo con el nombre "**interfaces.IMethod**" y su contenido será el nombre de la clase del método a implementar, por ejemplo "**method.impl.ELECTRE**", quedando de esta manera:

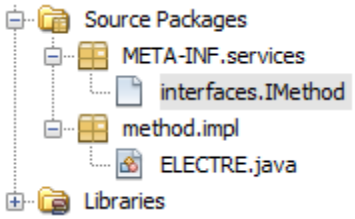

Figura 3.16: Estructura de archivos para un plugin de método

4. Una vez creada la clase que contendrá el método de solución, esta tiene que implementar la interfaz "IMethod", quedando para este ejemplo de la siguiente manera:

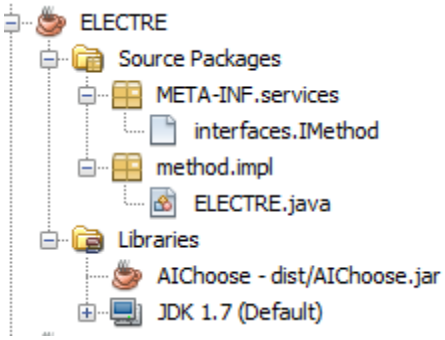

Figura 3.17: Estructura final del proyecto de plugin de método

5. Después que se compila el proyecto copiar el archivo *.jar para la carpeta "**methods**" de la plataforma.

Para el desarrollo e incorporación de nuevos plugins de dominio a la plataforma hay que seguir los siguientes pasos:

1. Una vez creado el proyecto de Java agregar la plataforma (solo el archivo *.jar) como biblioteca del proyecto.

2. Crear la siguiente estructura de paquetes en el proyecto:

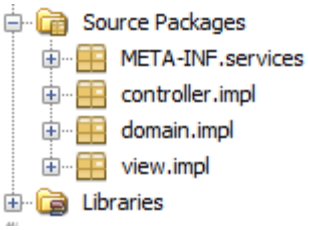

Figura 3.18: Estructura de paquetes para un plugin de dominio

3. Dentro del paquete "**META-INF.services**" crear un archivo con el nombre "**interfaces.IControllerDomain**" y su contenido será el nombre de la clase del controlador de dominio, por ejemplo "**controller.impl.DiscreteController**", quedando de esta manera:

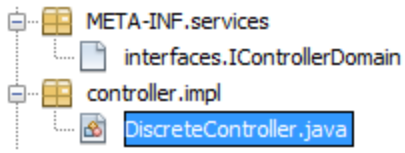

Figura 3.19: Estructura de archivos para un plugin de dominio

4. Una vez creada la clase que contendrá el dominio, esta tiene que implementar la interfaz "**IControllerDomain**".

5. Implementar el modelo del dominio, para este ejemplo una clase llamada "**DiscreteDomain**" que tiene que implementar la interfaz "**IDomain**".

6. Implementar la vista del dominio que se quiere añadir a la plataforma, quedando finalmente para este ejemplo de la siguiente manera:

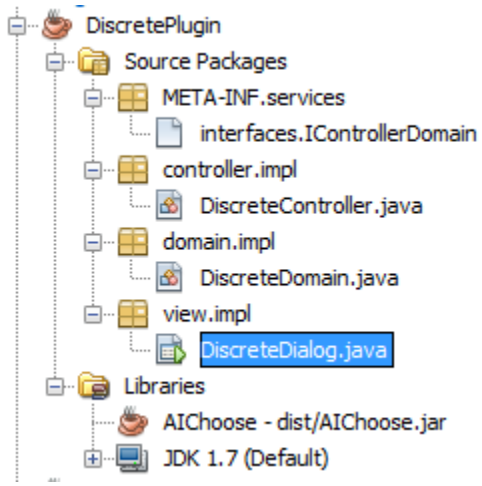

Figura 3.20: Estructura final del proyecto de plugin de dominio

7. Después que se compila el proyecto copiar el archivo *.jar para la carpeta "**domains**" de la plataforma.

3.8 Conclusiones parciales

En el presente capítulo se analizó la arquitectura principal de la herramienta computacional, se describieron los elementos necesarios para completar la fase de implementación y los principios de diseño utilizados, lo que posibilitó llegar a las siguientes consideraciones parciales:

1. A partir de los diagramas generados y teniendo en cuenta los principios de diseño se obtuvo una herramienta computacional que sirva de plataforma a los métodos de solución a problemas de toma de decisiones con múltiples criterios que sea flexible y extensible.
2. Se definió los pasos a seguir para la implementación e incorporación de nuevos plugins a la herramienta.
3. Se implementaron varios plugins de dominios para la herramienta.
4. Se implementaron los métodos de solución AHP y ELECTRE.

“Prueba y análisis de factibilidad”.

4. Pruebas y análisis de factibilidad

En este capítulo es donde se expone el análisis de costo del sistema, además de la descripción de las pruebas de software realizadas.

4.1 Estimación y costo

4.1.1 Calcular los Puntos de Casos de Uso (PCU)

El primer paso para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin ajustar. Este valor, se calcula a partir de la siguiente ecuación:

PCU = FPA + FPCU

Donde:

PCU: Puntos de Casos de Uso sin ajustar.

FPA: Factor de Peso de los Actores sin ajustar.

FPCU: Factor de Peso de los Casos de Uso sin ajustar.

Factor de Peso de los Actores sin ajustar (FPA)

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece según se indica en la siguiente tabla:

Tipo de actor	Descripción	Factor de peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de aplicación.	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema a desarrollar mediante una interfaz gráfica.	3

Tabla 4.1: Clasificación por tipo de actor

Se ha identificado 1 actores concretos, el Decisor que entra en la categoría de actores complejos puesto que es una persona que interactúa con el sistema. Por lo que se puede concluir que:

FPA = 3

Factor de Peso de los CU sin ajustar (FPCU)

Este valor se calcula mediante un análisis de la cantidad de CU presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los CU se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo según muestra la siguiente tabla:

Tipo de CU	Descripción	Factor de peso
Simple	El CU contiene de 1 a 3 transacciones.	5
Medio	El CU contiene de 4 a 7 transacciones.	10
Complejo	El CU contiene 8 o más transacciones.	15

Tabla 4.2: Clasificación por tipo de CU

En la siguiente tabla se muestra la cantidad de transacciones por caso de uso y sus pesos relativos:

Caso de uso	Cantidad de transacciones	Peso
Gestionar proyecto	5	10
Gestionar dominios	5	10
Gestionar criterios	4	10
Gestionar alternativas	4	10
Gestionar plantilla de Experto	3	5
Gestionar expertos	8	15
Importar evaluaciones	4	10
Solucionar problemas	10	15

Tabla 4.3: Clasificación de los CUS de la herramienta

Finalmente el FPCU se obtiene mediante la suma de los pesos de todos los CU como se muestra a continuación:

FPCU = 10+10+10+10+5+15+10+15= 85

Con los datos de FPA y FPCU obtenidos se puede determinar que:

PCU = 85+3 = 88

4.1.2 Calcular los Puntos de Casos de Usos Ajustados (PCUA)

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar este valor mediante la siguiente ecuación:

PCUA = PCU * FCT * FA

Donde:

PCUA: Puntos de Casos de Uso ajustados

PCU: Puntos de Casos de Uso sin ajustar

FCT: Factor de complejidad técnica

FA: Factor de ambiente

Calcular el Factor de Complejidad Técnica (FCT)

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la siguiente tabla se muestran los factores tenidos en cuenta y el peso de cada uno de estos:

Factor	Descripción	Peso
1	Sistema distribuido.	2
2	Objetivos de performance o tiempo de respuesta.	1
3	Eficiencia del usuario final.	1
4	Procesamiento interno complejo.	1
5	El código debe ser reutilizable.	1
6	Facilidad de instalación.	0.5
7	Facilidad de uso.	0.5
8	Portabilidad.	2

9	Facilidad de cambio.	1
10	Concurrencia.	1
11	Incluye objetivos especiales de seguridad.	1
12	Provee acceso directo a terceras partes.	1
13	Se requieren facilidades especiales de entrenamiento a usuarios.	1

Tabla 4.4: Factor de complejidad ténica

El FCT se calcula mediante la siguiente ecuación:

FCT = 0.6 + 0.01 * $\sum_{i=1}^{n} Peso_i * Valor\ asignado_i$

Haciendo un análisis de lo que plantea cada factor a medir y las características del sistema que se pretende realizar se asignaron los siguientes valores a cada factor:

Factor	Valor asignado	Valor asignado*Peso
1	4(El sistema es distribuido)	4 * 2 = 8
2	5(El sistema debe responder rápidamente a los pedidos)	5 * 1 = 5
3	3 (Los clientes no tienen por qué ser eficientes)	3 * 1 = 3
4	5 (Si existe un procesamiento complejo)	5 * 1 = 5
5	5 (Se desea que el código sea lo más reutilizable posible por las magnitudes que puede alcanzar el software)	5 * 1 = 5
6	3 (Se desea que el proceso de instalación no sea tan complejo puesto que una vez terminado el software pueden aparecer nuevos compradores)	3 * 0.5 = 1.5
7	5 (El software debe ser muy fácil de usar por cuanto los clientes no siempre tienen dominio sobre el trabajo con sistemas informáticos)	5 * 0.5 = 2.5
8	5 (A los usuarios del sistema si les interesa cambiar de SO)	5 * 2 = 10
9	5 (El sistema debe ser muy fácil de cambiar ya que en este negocio aparecen cosas nuevas cada día)	5 * 1 = 5
10	0	0 * 1 = 0
11	0	0 * 1 = 0
12	0	0 * 1 = 0
13	0	0 * 1 = 0

Tabla 4.5: Descripción de los factores para la herramienta.

Una vez obtenidos estos valores, se puede obtener el valor de FCT que viene dado por:

FCT = 0.6 + 0.01 * (8+5+3+5+5+1.5+2.5+10+5+0+0+0+0)

= 0.6 + 0.45 = 1.05

Calcular el Factor de Ambiente (FA)

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del FA. El cálculo del mismo es similar al cálculo del FCT, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5.

En la siguiente tabla se muestra el significado y el peso de cada uno de estos factores:

Factor	Descripción	Peso
1	Familiaridad con el modelo de proyecto utilizado.	1.5
2	Experiencia en la aplicación.	0.5
3	Experiencia en la orientación a objetos.	1
4	Capacidad del analista líder.	0.5
5	Motivación.	1
6	Estabilidad de los requerimientos.	2
7	Personal a tiempo parcial.	-1
8	Dificultad del lenguaje de programación.	-1

Tabla 4.6: Factor ambiente

Deben tenerse en cuenta las siguientes consideraciones:

- Para los factores E1 al E4, un valor asignado de 0 significa sin experiencia, 3 experiencia media y 5 amplia experiencia (experto).
- Para el factor E5, 0 significa sin motivación para el proyecto, 3 motivación media y 5 alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 0 significa que no hay personal a tiempo parcial, 3 significa mitad y mitad, y 5 significa que todo el personal es a tiempo parcial.

- Para el factor E8, 0 significa que el lenguaje de programación es fácil de usar, 3 medio y 5 que el lenguaje es extremadamente difícil.

El Factor de ambiente se calcula mediante la siguiente ecuación:

FA =1.4 - 0.03 * $\sum_{i=1}^{n} Peso_i * Valor\ asignado_i$

Haciendo un análisis de lo que plantea cada factor a medir, las características del equipo de desarrollo y de la solución a desarrollar se asignaron los siguientes valores a cada factor:

Factor	Valor asignado	Valor asignado *Peso
1	4	4 * 1.5 = 6
2	2	2 * 0.5= 1
3	4	4 * 1 = 4
4	4	4 * 0.5 = 2
5	5	5 * 1 = 5
6	3	3 * 2 = 6
7	0	0 * -1 = 0
8	3	3 * -1 = -3

Tabla 4.7: Descripción del Factor ambiente para la herramienta.

Una vez obtenidos estos valores, se puede obtener el valor de FCT que viene dado por:

FA = 1.4 + 0.03 * (6 + 1 + 4 + 2 + 5 + 6 + 0 - 3)

= 1.4 + 0.63 = 2.03

En estos momentos se cuenta con los valores de PCU, FCT y FA y por tanto se puede calcular el valor de PCUA el cual viene dado en este caso por:

PCUA = 88 * 1.05 * 2.03 = 187.57

4.1.3 Calcular el Esfuerzo de desarrollo (E)

El esfuerzo en horas-hombre viene dado por:

E = PCUA x FC

Donde:

E: esfuerzo estimado en horas-hombre.

PCUA: Puntos de Casos de Uso ajustados.

FC: Factor de conversión. Para el cálculo del FC se siguen los siguientes pasos:

1. Se contabilizan cuántos factores de los que afectan al FA están por debajo del valor medio (3), para los factores 1 al 6.
2. Se contabilizan cuántos factores de los que afectan al FA están por encima del valor medio (3), para los factores 7 y 8.
3. Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/PCU, es decir, un PCU toma 20 horas-hombre.
4. Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/PCU, es decir, un PCU toma 28 horas-hombre.
5. Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

Se debe tener en cuenta que este método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso.

Finalmente, para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software.

Para ello se puede tener en cuenta el siguiente criterio, que estadísticamente se considera aceptable, el cual plantea la distribución del esfuerzo entre las diferentes actividades de un proyecto de la siguiente manera:

Actividad	Porcentaje
Análisis	10%
Diseño	20%
Implementación	40%
Pruebas	15%

Sobrecarga (otras actividades)	15%

Tabla 4.8: Estimación por flujo de trabajo.

Para este caso, y siguiendo los criterios que existen para el FC se tomó 20 como valor del mismo y por tanto:

E = 187.57 * 20 = 3751.4 Horas-Hombre

Este esfuerzo es el que se requiere para la implementación. Si se tiene en cuenta que este representa un 40 % del esfuerzo total para desarrollar el software entonces tenemos que el esfuerzo total es el siguiente:

E (Total) = E / 0.4 = 9378.5 Horas-Hombre

La siguiente tabla muestra el esfuerzo necesario para cada actividad del proyecto siguiendo los porcentajes especificados en la tabla anterior:

Actividad	Horas-Hombre
Análisis	937.85
Diseño	1875.7
Implementación	3751.4
Pruebas	1406.78
Sobrecarga (otras actividades)	1406.78
Total	9378.5

Tabla 4.9: Estimación en Horas-Hombre por flujo de trabajo.

4.1.4 Estimación del tiempo de desarrollo del proyecto

El tiempo de desarrollo aproximado del proyecto (TDes) se calcula de la siguiente manera:

TDes = E (Total)/CH

Donde:

E (Total): Esfuerzo total

CH: Es la cantidad de hombres que desarrollan el proyecto.

Por tanto:

TDes = 9378.5 HH / 1 Hombres = 9378.5 Horas

4.1.5 Estimación del costo de desarrollo del proyecto

Una vez estimado el tiempo de desarrollo del proyecto y conociendo la cantidad de desarrolladores y el pago que recibe cada uno de estos se puede llevar a cabo una estimación del costo total del proyecto referidos a los recursos humanos; existen otros costos como por ejemplo del equipamiento que se suman al anterior. El costo por concepto de desarrolladores viene dado por:

C = E (Total) * CHH

Donde:

CHH: Costo por hombre hora

CHH = K * THP

*Donde**:***

K: Coeficiente que tiene en cuenta los costos indirectos (1,5 y 2,0).

THP: Tarifa Horaria Promedio. El salario promedio de las personas que trabajan en el proyecto dividido entre 160 horas.

Entonces:

C = E (Total) * K * THP

El salario promedio de 1 desarrollador es de $1000 y por tanto la **THP** = 1000 / 160 = 6.25

C = 9378.5 * 2 * 6.25 = **$117231.25**

4.2 Pruebas de software

Dentro del proceso de desarrollo de software la fase de pruebas es una de las más importantes. El objetivo de esta fase es validar que los requerimientos de software han sido cumplidos, además de garantizar la calidad del sistema. Para esto se realizarán varias pruebas de caja negra a la herramienta utilizando distintos casos de prueba.

4.2.1 Caso de prueba 1

En este caso de prueba se expondrá un ejemplo hipotético donde se pueda apreciar a simple vista que una de las alternativas es la más dominante. Luego se verificará que la herramienta obtiene el resultado esperado. A continuación se expone la matriz de decisión de un experto ficticio para este ejemplo:

	Criterio1 (**C1**)	Criterio2 (**C2**)	Criterio3 (**C3**)	Criterio4 (**C4**)	Criterio5 (**C5**)
Alternativa1 (**A1**)	8	8	5	9	5
Alternativa2 (**A2**)	5	9	3	5	7
Alternativa3 (**A3**)	7	7	3	6	3
Alternativa4 (**A4**)	9	8	9	8	7
Alternativa5 (**A5**)	9	7	5	2	4
Alternativa6 (**A6**)	9	5	8	7	6

Tabla 4.10: Modelo caso de prueba 1

Luego, los juicios asociados a los criterios van a estar en el intervalo [1,9] maximizando sus valores; además los criterios tienen el orden de preferencia siguiente: C1 > C2 > C3 > C4 > C5, con su respectivo vector de pesos W= (0.438, 0.219, 0.146, 0.109, 0.088). Donde se puede apreciar a simple vista que la alternativa más dominante es la A4, por tanto da como resultado:

Para el método ELECTRE.

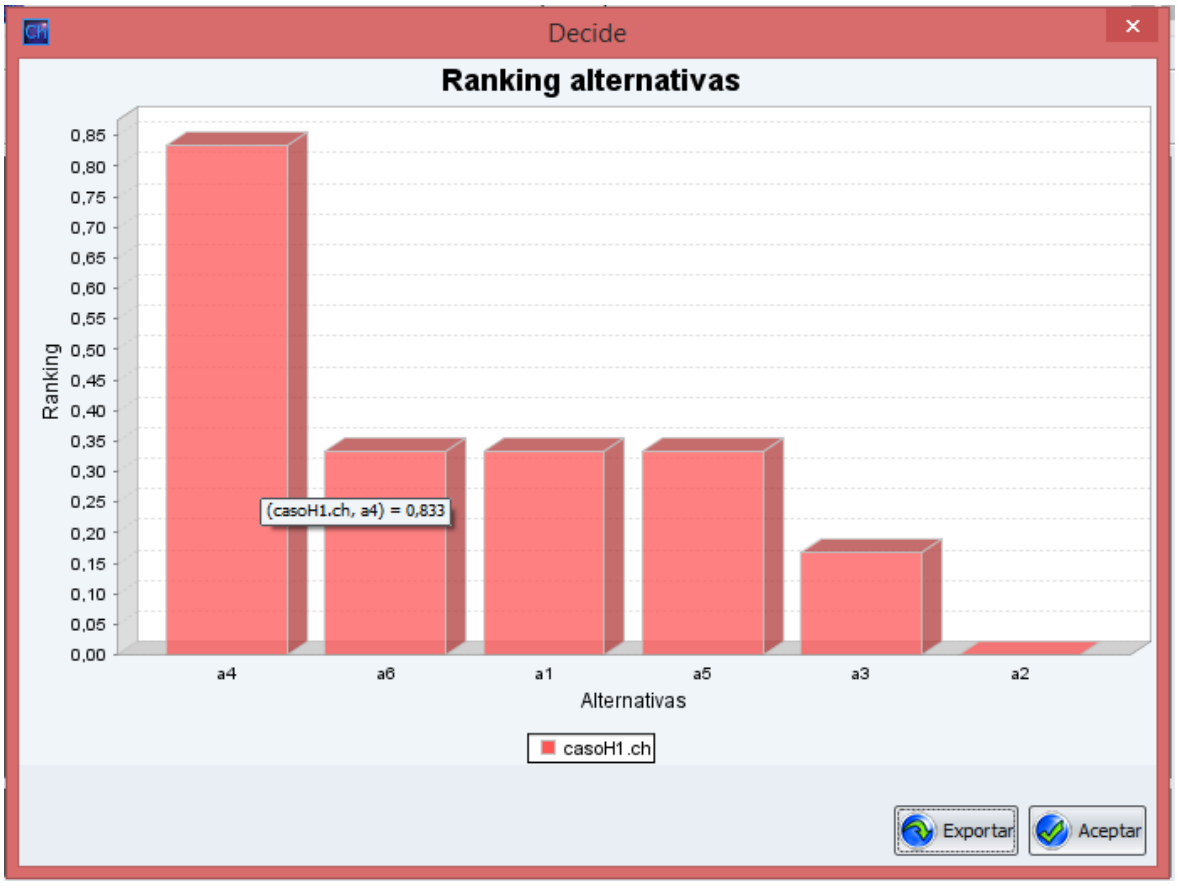

Figura 4.1: Resultado ELECTRE Caso de prueba 1

Para el método AHP:

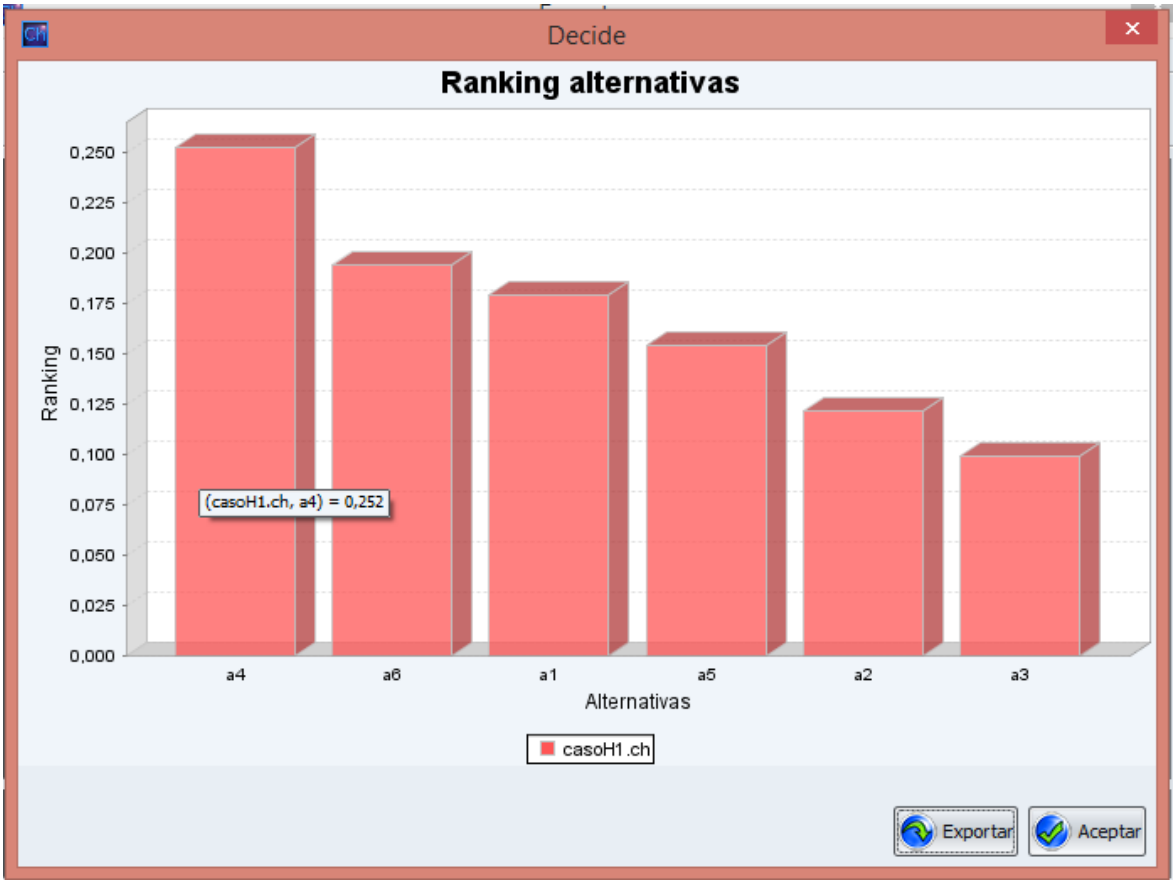

Figura 4.2: Resultado AHP Caso de prueba 1

4.2.2 Caso de prueba 2

En este caso de prueba se expondrá un ejemplo hipotético donde se pueda apreciar a simple vista que una de las alternativas es la menos dominante. Luego se verificará que la herramienta obtiene el resultado esperado. A continuación se expone la matriz de decisión de un experto ficticio para este ejemplo:

	Criterio1 (**C1**)	Criterio2 (**C2**)	Criterio3 (**C3**)	Criterio4 (**C4**)	Criterio5 (**C5**)
Alternativa1 (**A1**)	Regular	Bueno	Regular	Regular	Malo
Alternativa2 (**A2**)	Malo	Bueno	Bueno	Bueno	Malo
Alternativa3 (**A3**)	Bueno	Regular	Bueno	Bueno	Regular
Alternativa4 (**A4**)	Regular	Regular	Malo	Bueno	Regular
Alternativa5 (**A5**)	Malo	Malo	Regular	Regular	Regular

Alternativa6 (**A6**)	Bueno	Bueno	Malo	Malo	Regular

Tabla 4.11: Modelo caso de prueba 1

Luego, los juicios asociados a los criterios van a estar en el dominio discreto [bueno, regular, malo] donde bueno es el resultado preferible; además los criterios tienen el orden de preferencia siguiente: C1 > C2 > C3 > C4 > C5, con su respectivo vector de pesos W= (0.438, 0.219, 0.146, 0.109, 0.088). Donde se puede apreciar a simple vista que la alternativa menos dominante es la A5, por tanto da como resultado:

Para el método ELECTRE:

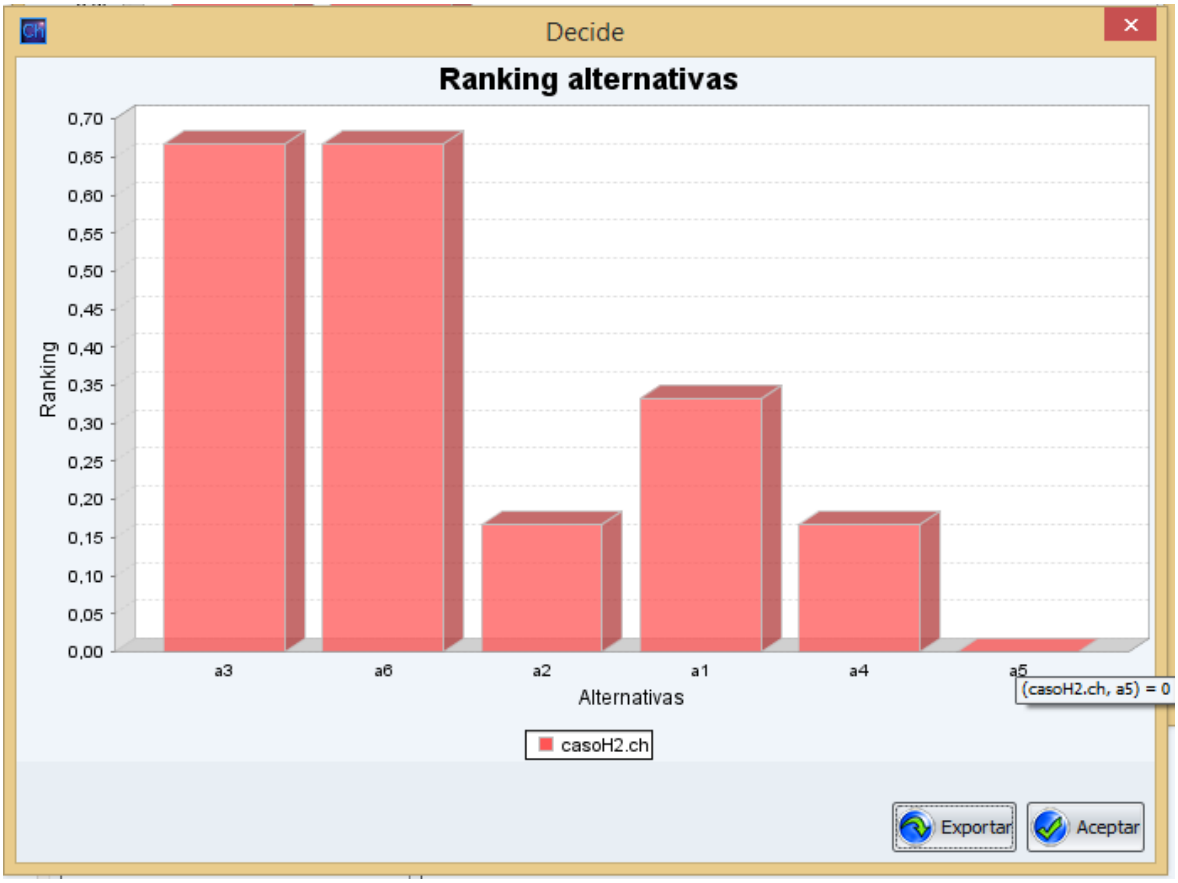

Figura 4.3: Resultado ELECTRE Caso de prueba 2

Para el método AHP:

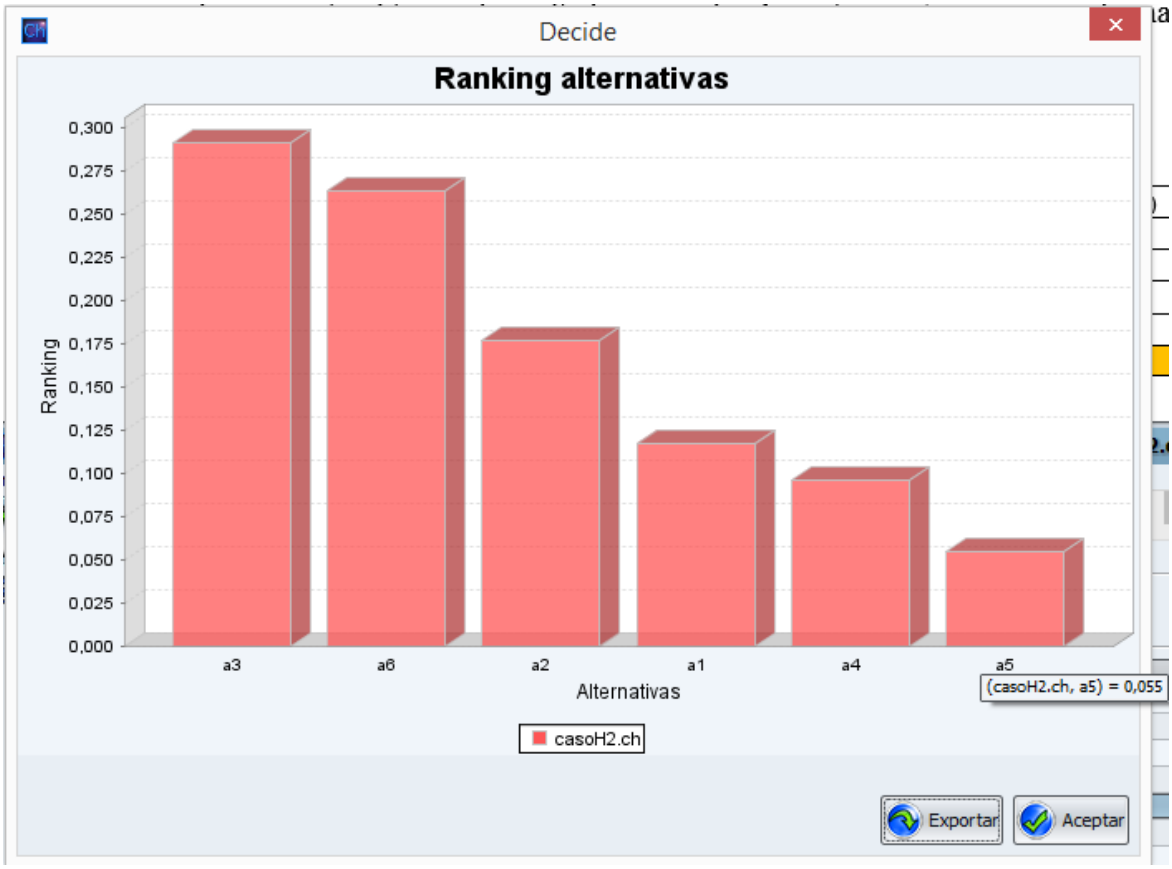

Figura 4.4: Resultado AHP Caso de prueba 2

4.2.3 Caso de prueba 3

En la empresa Transporte Agropecuario de Villa Clara se reciben a finales de mes las solicitudes de transporte por parte de los clientes para el mes próximo. Se lleva a cabo una comisión de contratación para analizar las demandas y escoger las transportaciones más provechosas para la organización teniendo en cuenta los siguientes criterios:

1. La existencia de un contrato actualizado (contrato).
2. La existencia de deudas (deudas).
3. La solvencia económica del cliente (solvencia_economica).
4. Que la transportación tenga un buen aprovechamiento del recorrido, o sea que cuando llegue a su destino el transporte no regrese vacío (aprov_recorrido).

5. Que la transportación aproveche la capacidad de carga del camión (aprov_capacidad_carga).
6. Ciclo de viaje, o sea cuánto se demora en realizar la transportación (ciclo_viaje).
7. Rotación, que es el tiempo que demora en poder realizar otra transportación luego de atendido el pedido (rotacion).

Luego, teniendo en cuenta que los criterios ya están ordenados de acuerdo a su preferencia se les asigna un peso usando la siguiente función de ponderación preferencial:

$$W_j = \frac{1/r_j}{\sum_{j=0}^{n} 1/r_j}$$

Donde:

W_j , es el peso obtenido para el criterio j

r_j, es la posición que tiene el criterio en la lista ordenada

Obteniendo el siguiente vector de pesos W = (0.3857, 0.1928, 0.1286, 0.0964, 0.0771, 0.0643, 0.0550)

Todos los criterios están comprendidos en el dominio Z ϵ [1;9] donde, a mayor valor obtenido significa que la alternativa es más preferible en este criterio.

Luego, las alternativas analizadas en este caso de prueba son:

- Empresa Apícola de Villa Clara, que quiere transportar miel de abeja.
- Empresa de Suministros de Villa Clara, que quiere transportar sacos.
- Agropecuaria Osvaldo Herrera, que quiere transportar ganado.
- Agropecuaria Emilio Córdova, que quiere transportar arroz.

Obteniendo las siguientes evaluaciones en conjunto con el jefe comercial de esta entidad:

	contrato	deudas	solvencia_economica	aprov_recorrido	aprov_capacidad_carga	ciclo_viaje	rotacion
Empresa Apícola de Villa Clara	9	9	9	8	9	9	9
Empresa de Suministros de Villa Clara	9	9	8	9	5	4	4

Agropecuaria Osvaldo Herrera	**9**	**5**	**4**	**4**	**5**	**9**	**9**
Agropecuaria Emilio Córdova	**9**	**8**	**6**	**7**	**7**	**7**	**6**

Tabla 4.12: Evaluaciones Caso de prueba 3

Luego, teniendo en cuenta estas evaluaciones se obtienen los siguientes resultados en la herramienta computacional:

Para el método ELECTRE:

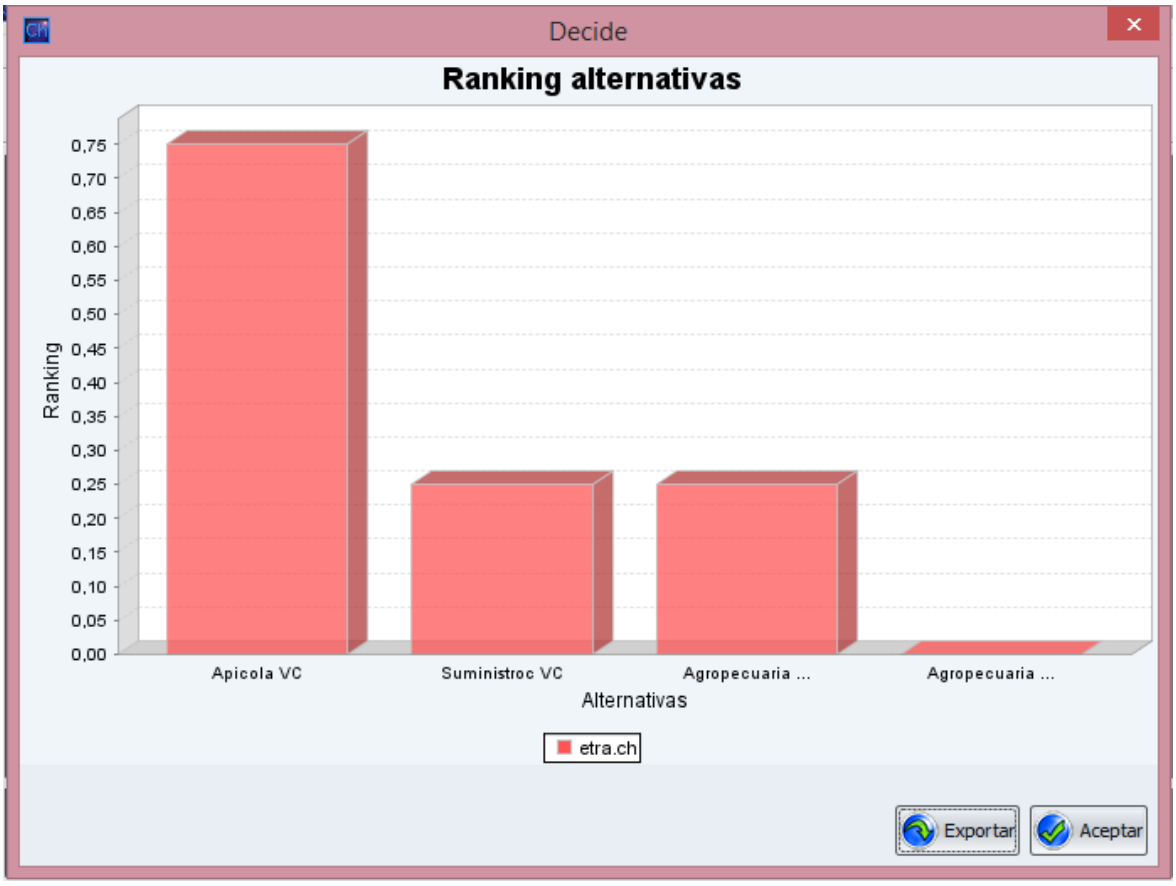

Figura 4.5: Resultado ELECTRE Caso de prueba 3

Para el método AHP:

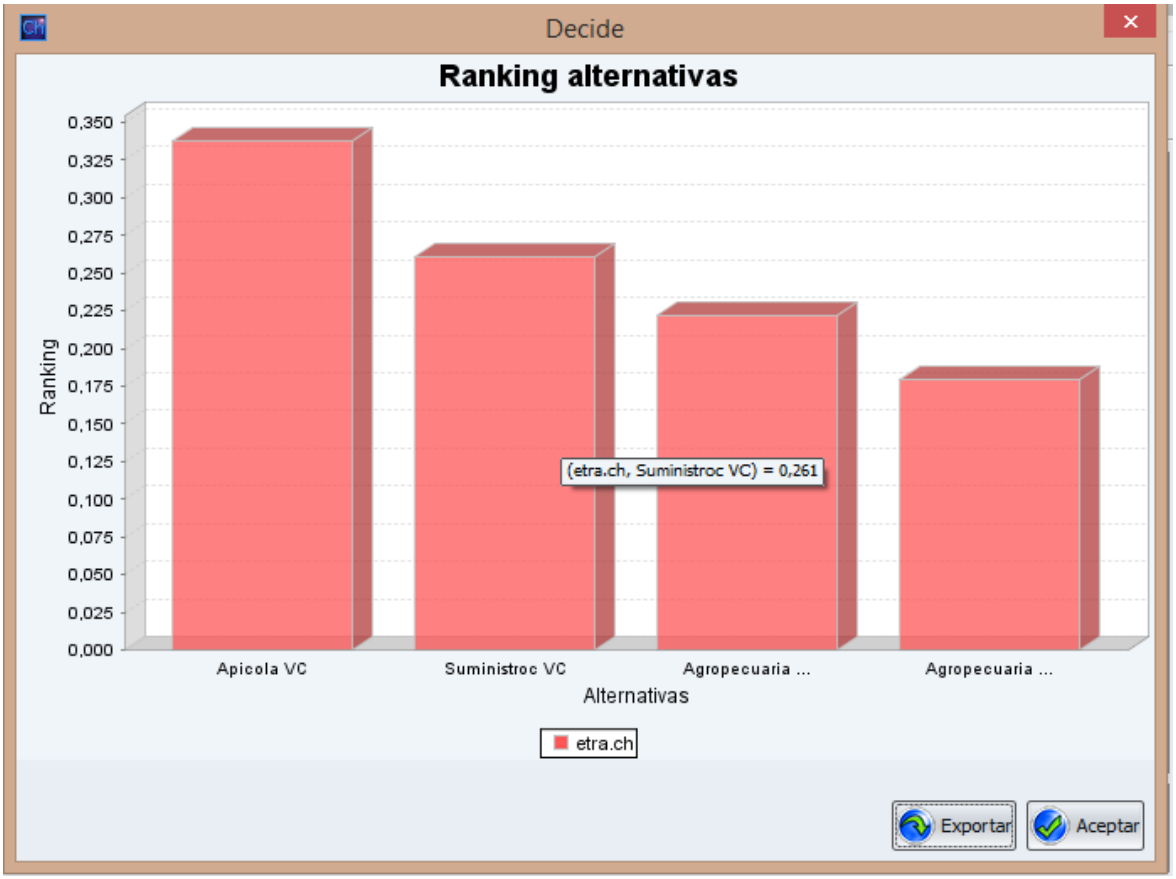

Figura 4.6: Resultado AHP Caso de prueba 3

4.3 Conclusiones parciales

En este capítulo se realizaron varias pruebas de caja negra a la herramienta computacional y se hizo un análisis de esfuerzo por lo que se puede concluir:

1. La herramienta tiene un elevado nivel de complejidad.

2. Las pruebas de software arrojaron resultados favorables.

Conclusiones

Con el desarrollo de la presente investigación se dio cumplimiento a los objetivos planteados, pudiéndose arribar a las siguientes conclusiones:

1. Del estudio de problemas de toma de decisiones con múltiples criterios se logra una correcta representación dentro de la herramienta computacional.
2. A partir del análisis de las principales desventajas de diversos sistemas de ayuda a la toma de decisiones encontradas en la literatura, se concibe un diseño de la herramienta flexible y extensible con la ayuda de los patrones de diseño.
3. Se implementaron e integraron varios plugins de métodos y dominios a la plataforma.
4. Se realizaron satisfactoriamente las pruebas de software.

Recomendaciones

A partir de la implementación de la herramienta computacional se recomienda:

- Agregar a la herramienta nuevos plugins de métodos de solución.
- Integrar a la herramienta nuevos plugins de dominio.
- Integrar el uso de la lógica difusa en la herramienta.

Referencias bibliográficas

A. KROHLING, R. & C. CAMPANHARO, V. 2010a. Fuzzy TOPSIS for group decision making: A case study for accidents with oil spill in the sea. *Elsevier*.

A. KROHLING, R. & C. CAMPANHARO, V. 2010b. Fuzzy TOPSIS for group decision making: A case study for accidents with oil spill in the sea. *Elsevier.*

ALEDO, J. A., GÁMEZ, J. A. & MOLINA, D. 2013. Tackling the rank aggregation problem with evolutionary algorithms. *www.elseiver.com/locate/amc*.

ARZA PÉREZ, L. & BORREGO LEÓN, D. 2012. Sistema de soporte a la decisión para la ubicación del estudiante en un rol del proceso de desarrollo de software.

AYUB, M., KABIR, M. J. & RABIUL ALAM, M. G. 2009. *Personnel Selection Method Using Analytic Network Process (ANP) and Fuzzy Concept*.

BOOCH, G., JACOBSON, I. & RUMBAUGH, J. 2007. *El Lenguaje Unificado de Modelado*.

BURKE, E. K. & KENDALL, G. 2005. *Search methodologies*, Springer Science-i-Business Media, LLC

BUSCHMAN, F., HENNEY, K. & C. SCHMIDT, D. 2007. *Pattern-Oriented software architecture*.

CABLES PÉREZ, E. H. 2011. *Selección de personal con técnicas de Soft Computing. Propuesta de Desarrollo y de Software.* Tesis Doctoral, UNIVERSIDAD DE GRANADA.

CHEN-TUNG, C., YUAN-CHU, H. & WEI-ZHAN, H. 2009. Applying Multiple Linguistic PROMETHEE Method for Personnel Evaluation and Selection. *Proceedings of the 2009 IEEE IEEM.*

F. SHIPLEY, M. & JOHNSON, M. 2009. *A fuzzy approach for selecting project membership to achieve cognitive style goals*.

FOWLER, M., RICE, D., FOEMMEL, M., EDWARD, H., MEE, R. & STAFFORD, R. 2002. *Patterns of Enterprise Application Architecture*, Addison Wesley.

HAERER, W. 2000. Software review: Criterium Decision Plus 3.0. *ORMS Today.*

JADIDI, O., HONG, T. S., FIROUZI, F., YUSUFF, R. M. & ZULKIFLI, N. 2008. TOPSIS and fuzzy multi-objective model integration for supplier selection problem. *Journal of Achievements in Materials and Manufacturing Engineering.* Malasia: Journal of Achievements in Materials and Manufacturing Engineering.

KARIMI, A. R., MEHRDADI, N., HASHEMIAN, S. J., NABI-BIDHENDI, G. R. & TAVAKKOLI-MOGHADDAM, R. 2011. *Using of the Fuzzy TOPSIS and Fuzzy AHP Methods for wastewater treatment process selection*.

KAUFMANN, A. & GIL ALUJA, J. 1993. *Introducción de la Teoría de los subconjuntos borrosos a la Gestión de las empresas*.

MARTÍNEZ CHÁVEZ, O. F. 2013. *PROPUESTA DE ANÁLISIS PARA LA INTEGRACIÓN DE INFORMACIÓN PARA APOYAR LA TOMA DE DECISIONES EN UNA ORGANIZACIÓN PÚBLICA.* INSTITUTO POLITÉCNICO NACIONAL.

MCGINLEY, P. 2012. Decision analysis software survey. *ORMS Today.*

MICHAY, P. 2011. *S. M. JGAP,* Loja – Ecuador.

MUSTAJOKI, J. & MARTTUNEN, M. 2013. Comparison of Multi-Criteria Decision Analytical Software

Searching for ideas for developing a new EIA-specific multi-criteria software.

OSORIO GÓMEZ, J. C. O. C., JUAN PABLO. 2008. EL PROCESO DE ANÁLISIS JERÁRQUICO (AHP) Y LA TOMA DE DECISIONES

MULTICRITERIO. EJEMPLO DE APLICACIÓN.

PEÑA TÁRAMO, Y. 2013. Implementación del método TOPSIS en un sistema de ayuda en el proceso de Toma de Decisiones Multicriterio.

PRESSMAN, R. S. 2010. *Software Engineering A Practitioner's Approach*, McGraw-Hill Companies, Inc.

RUIZ-TAGLE MOLINA, J. M. 2011. *ALGORITMOS DE CÁLCULO DE VECTORES DE PRIORIDAD A*

PARTIR DE MATRICES DE COMPARACIÓN POR PARES

IMPRECISAS. TESIS DOCTORAL, UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA.

SÁNCHEZ GONZÁLEZ, I. J. & GONZÁLEZ LANDEIRO, A. R. 2013. Implementación de los métodos Proceso de Análisis Jerárquico y Proceso Analítico en Red en un sistema de soporte a la toma de decisiones multicriterio.

SÁNCHEZ GONZÁLEZ, I. J., VELÁZQUEZ DOMÍNGUEZ, H. M., LECUONA RODRÍGUEZ, A. L. & PEÑA TÁRAMO, Y. 2013. *Sistema de Soporte a la toma de decisiones multicriterio Multidecision PAAT*.

SIRAJ, S., MIKHAILOV, L. & KEANE, J. A. 2013. *PriEsT: an interactive decision support tool to estimate priorities from pairwise comparison judgments*.

SOMMERVILLE, I. 2011. *SOFTWARE ENGINEERING*, Pearson Education, Inc.

TECH'S, L. 2013. The PlugIn Design Pattern. *Software Architecture & Design Patterns* [Online]. Available from: http://www.lm-tech.it.

TORRES ESTOL, R. & PADRÓN DEL PICO, J. E. 2013. IMPLEMENTACIÓN DEL MÉTODO PROMETHEE EN UN SISTEMA DE AYUDA A LA TOMA DE DECISIÓN MULTICRITERIO.