Universidad de las Ciencias Informáticas

Facultad 2

Sistema Inteligente para el apoyo a la Toma de Decisiones combinando el Razonamiento Basado en Casos y el algoritmo de agrupamiento LC-Conceptual

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Liyanis Peláez Baños

Erisel Almarales Vázquez

Tutores:

MSc Yunia Reyes González

Dra Natalia Martínez Sánchez

La Habana, junio 2015

"Año 57 de la Revolución"

"El único autógrafo digno de un hombre es el que deja escrito con sus obras."

José Martí

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la UCI los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Firma del autor	Firma del autor
Liyanis Peláez Baños	Erisel Almarales Vázquez
Firma del tutor	Firma del tutor
Dra Natalia Martínez Sánchez	MSc Yunia Reyes González

A la persona que me inspiró siempre con su carácter, su fuerza, su valor y su destreza para superar los problemas, esa por la que me esfuerzo diariamente para ganarme su orgullo, a ti MAMI te agradezco TODO.

Al viejátilo por creer en mí en todo momento y por cubrir todas las expectativas de un padre para mi.

A mi segunda madre Mary por estar ahí en los momentos más difíciles, por apoyarme y quererme. A Ernesto por cuidar de Mary, por su amistad sincera y por el cariño que me ha dado.

A Jose María y Clara que son los miembros más recientes de mi familia, gracias por su afecto y por cuidar de mi viejita.

A mi novio Raidel que tanto me ha apoyado en esta etapa, que soporta mis dolores y mi mal humor y aun así permanece a mi lado. Te quiero pipo.

A mi hermana Mileidis que me alienta cuando me siento triste, que me apoya y me aconseja. A mi hermana Liselis que aunque viva peleando con ella quiero que sepa que la quiero mucho, y que las cosas que le digo son para que crezca como persona. A mi hermano Osvaldito que me enseñó que podía lograr todo lo que me propusiera, hoy se encuentra lejos, pero siempre llevo conmigo su cariño y sus buenos consejos.

A mi suegra Mally por acogerme con tanto cariño.

A mis amigos que llamo con mucho orgullo porque se que son dignos de ese nombre: Luis Daniel, Gaby, Frank, Laimber y Ayeban.

A mi compañero de tesis que por todo el trabajo y esmero dedicado, y por escuchar mis regaños en todo el proceso.

A mis queridas tutoras por guiarme y confiar en mí.

A todos ustedes…Gracias.

Liyanis Peláez Baños.

Al Guako por ser el padre con quien puedes contar. Por darme animo cuando creía que no podía. Por entenderme y apoyarme.

A mi hermano del alma Kalelito que hoy no pudo estar aquí pero que siempre esta en mi corazón.

A Tito, mi hermanito que está más grande que yo. Crece mi hermano pa´ que me defiendas.

A mi tio Maikel por estar en todo momento y ayudarme a desconectar del estudio, y hacerme saber que no estaba solo.

A mis compañeros de aula por arrastrarme hasta quito año. A Ronal, el Blade, el Musche, Luigi, por aguantar el chucho y por las discusiones a las 4 de la madrugada sobre cualquier tema.

Al Pelú y Daniel por aguantarme de compañero de cuarto y prestarme la máquina para que programara toda la noche o poner la guantanamera a las 3 de la madrugada. Supongo que para desconectar.

A la Rubia que ya no es rubia. Mi compañera de tesis por ser una jefa intransigente y no dejarme ir para San Antonio pese a mis numerosos intentos de soborno. Sin ti no se que estaríamos presentando hoy.

A todas las personas que de una manera u otra han contribuido para que hoy pueda decir soy INGENIERO. A todos gracias.

Erisel Almarales Vázquez

A mi madre Rosa.

Liyanis Peláez Baños

Dedico esta tesis a mi viejita linda. Mi madre. Quien ha sacrificado mucho por que yo este aquí. Bichito eres lo más grande que tengo en la vida. Eres la razón por la que supero cualquier obstáculo. Espero que te sientas orgullosa de mí como yo lo estoy de ti.

Erisel Almarales Vázquez

Resumen

A partir del estudio de los Sistemas Basados en Casos y de los algoritmos de agrupamiento conceptuales en el marco del enfoque lógico combinatorio, se desarrolló una herramienta que permite implementar el ciclo de funcionamiento del Razonamiento Basado en Casos integrado al algoritmo de agrupamiento LC-Conceptual, en el que la estructura de la base de casos garantiza un acceso y recuperación eficientes para el apoyo a la toma de decisiones en dominios de aplicación de datos mezclados e incompletos.

Palabras clave: LC Conceptual, Razonamiento Basado en Casos.

Summary

From the study of the Systems Based on Cases and the conceptual algorithms of grouping in the frame of the logical combinatorial approach, there developed a tool that allows to implement the cycle of functioning of the Reasoning Based on Cases integrated to the algorithm of LC-Conceptual grouping, in which the structure of the base of cases guarantees an efficient access and recovery for the support to the capture of decisions in domains of application of mixed and incomplete information.

Keywords: LC-Conceptual, Case Based Reasoning.

Índice

Introducción

En la actualidad el análisis y la toma de decisiones se han convertido en una tarea fundamental en la dirección de diferentes procesos. El desarrollo de la computación ha tenido un impacto significativo en esta cuestión por las posibilidades que brinda para almacenar y manipular grandes cantidades de información.

La toma de decisiones es el proceso de identificación y selección de la acción adecuada para la solución de un problema específico. No hay un método de toma de decisiones que garantice que siempre se tomará la decisión correcta. Sin embargo la aplicación de un método racional, inteligente y sistemático tendrá mayores probabilidades que otros de llegar a soluciones de alta calidad.

Los Sistemas de Ayuda a la Decisión son sistemas informáticos interactivos que ayudan a los que toman las decisiones en lugar de sustituirlos. Utilizan datos, modelos, resuelven problemas con varios grados de estructura, y se centran en la efectividad de los procesos de decisión más que en la eficiencia.

Modelos como el Razonamiento Basado en Casos (Kolodner, 1992) se utilizan en la toma de decisiones a partir de ejemplos de situaciones y decisiones pasadas. En (Pomerol, 1995) se plantea que el RBC[1] constituye uno de los modelos emergentes de la Inteligencia Artificial más interesantes para la toma de decisiones.

El RBC ha sido utilizado para resolver problemas en las distintas esferas económicas, científicas, tecnológicas y sociales, por citar algunas, a través de la creación de numerosas aplicaciones en un espectro bastante extenso de dominios de aplicación, incluyendo las siguientes: El diagnóstico médico, el análisis financiero, la clasificación, previsión y planeación, entre otras.

El RBC representa un método para resolver problemas no estructurados, en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos. En este paradigma la base del comportamiento inteligente de un sistema radica en recordar situaciones similares existentes en el pasado. Debe destacarse que es una técnica, en la cual la memoria se sitúa como fundamento de la IA[2] y más concretamente de los sistemas basados en el conocimiento. (Gálvez Lio, 2008) El RBC constituye el método de solución de problemas en los Sistemas Basados en Casos.

Los SBC[3] apoyan sus predicciones en ejemplos (casos). Una función de distancia o de semejanza determina los casos más semejantes al nuevo problema y las soluciones de los casos recuperados se adaptan para obtener una solución. (Bello, 2002) Durante este proceso se

[1] **RBC:** Razonamiento Basado en Casos.
[2] **IA:** Inteligencia Artificial.
[3] **SBC:** Sistemas Basados en Casos.

presentan dos problemas fundamentales: lograr una representación de la memoria de casos que permita una recuperación eficiente y considerar la relevancia del conocimiento para mejorar la eficiencia en el proceso de adaptación (Kolodner, 1992).

Para lograr que la velocidad de recuperación no sea afectada por el volumen de la memoria permanente, la organización tiene que permitir acceder eficientemente a los elementos de memoria relevantes al problema. Por eso la organización de la memoria está estrechamente vinculada con el método de búsqueda. (Gálvez Lio, 2008)

Entre las problemáticas actuales de los SBC se encuentran: la selección de rasgos relevantes, cuestión central tanto en la definición del modelo de la base de casos como en el modelo de recuperación de casos y decidir cómo la memoria de casos debe ser organizada para un almacenamiento, recuperación y reúso efectivos y eficientes.

Una variante para tener en cuenta a la hora de organizar la base de casos e implementar el ciclo de funcionamiento del RBC puede ser a partir de los algoritmos de agrupamiento conceptual en el marco del enfoque lógico-combinatorio. (Reyes González, 2014)

El agrupamiento plantea que dado un conjunto de objetos definidos en términos de un conjunto de rasgos, los algoritmos de agrupamiento intentan construir particiones de este conjunto, donde la semejanza intra-grupo sea máxima y la semejanza inter-grupos sea mínima. (Pons Porrata, 2004)

Los algoritmos de agrupamiento conceptual surgen como una alternativa en la clasificación no supervisada para la solución de algunos problemas donde es necesario obtener los conceptos de los grupos resultantes.

El agrupamiento conceptual está compuesto por dos tareas fundamentales: el agrupamiento de entidades en el que se determinan grupos a partir de una colección de objetos (estructuración), y la caracterización, en la cual se determina el concepto de cada grupo de la estructuración, las propiedades que caracterizan el agrupamiento. (Reyes González, y otros, 2014)

Los algoritmos de agrupamiento conceptual se pueden dividir en dos grandes grupos: los algoritmos incrementales y los no incrementales. Los algoritmos incrementales basan su funcionamiento en la adaptación de los grupos (o conceptos) con los nuevos objetos que se le van presentando, es decir, cada vez que se analiza un nuevo objeto éste se clasifica, mediante cierta estrategia, en uno de los grupos ya existentes o se crean nuevos grupos. Por lo general, estos algoritmos se han desarrollado para los casos en que el conjunto de objetos a estructurar no está completamente dado, es decir, es dinámico. Por otro lado, los algoritmos no incrementales estructuran una muestra de objetos utilizando el conjunto de datos completo. (Guerra Gandón, y otros, 2012)

Se debe tener en cuenta que en los problemas de la vida real, los datos que se puedan procesar pueden ser numéricos, no numéricos o simplemente no tener la información. Procesar datos mezclados e incompletos es un problema que ha sido abordado por varios investigadores.

Ante esta situación se pueden adoptar cuatro posiciones diferentes: analizar los rasgos numéricos por separado de los rasgos no numéricos y posteriormente tomar una decisión sobre la base de los resultados parciales obtenidos, el único problema de esto es que las variables no numéricas nunca pueden ser analizadas junto a las variables numéricas; discretizar los rasgos numéricos de dos formas sustancialmente diferentes: considerando los nuevos valores como nombres o como valores de una lógica dada (bivalente o k-valente), en este caso a pesar de su popularidad, es un hecho que hay rasgos que no pueden ser discretizados o perderían su esencia; codificar los rasgos no numéricos considerando los códigos como números y procediendo con ellos como si lo fueran, este procedimiento puede considerarse más inadecuado que todos los anteriores ya que los códigos no son números y se conoce que, en general, es imposible aplicar operadores matemáticos a códigos; finalmente trabajar con las descripciones mezcladas e incompletas de manera simultánea, este es justamente el enfoque que sigue el reconocimiento lógico-combinatorio de patrones.

El algoritmo LC-Conceptual está basado en los conceptos de la clasificación no supervisada en el enfoque lógico combinatorio y retoma algunas ideas propuestas por Michalski para generar conceptos, en términos del conjunto de rasgos original. (Guerra Gandón, y otros, 2012) Este algoritmo puede manipular combinaciones de atributos tanto cualitativos como cuantitativos, además de aquellos atributos con ausencia de información. Los agrupamientos se crean sobre la base de ciertas propiedades de la semejanza entre objetos y los conceptos construidos son propiedades lógicas basadas en los rasgos, en término de los cuales se describen a los objetos en análisis.

Se han desarrollado sistemas para el apoyo a la toma de decisiones, sin embargo en la literatura científica consultada no se han encontrado herramientas computacionales que soporten los Sistemas Basados en Casos integrados con los algoritmos de agrupamiento conceptuales para organizar la base de casos permitiendo la recuperación de los casos relevantes al problema, y a su vez procesen simultáneamente datos mezclados e incompletos.

La problemática previamente descrita origina la necesidad de desarrollar una investigación para dar respuesta al siguiente **problema científico:** ¿Cómo contribuir a la toma de decisiones en un sistema basado en casos en dominios de aplicación de datos mezclados e incompletos?

Teniendo en cuenta el problema antes mencionado se define como **objeto de estudio:** El Razonamiento Basado en Casos y el enfoque lógico combinatorio para el agrupamiento conceptual.

Se propone como **objetivo general:** Desarrollar un Sistema Inteligente mediante el uso del RBC

y el algoritmo de agrupamiento LC-Conceptual, en dominios de aplicación de datos mezclados e incompletos para el apoyo a la toma de decisiones.

Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos:**

1. Definir los referentes teóricos de los SBC y los algoritmos de agrupamiento conceptual.
2. Implementar el ciclo del RBC utilizando el algoritmo de agrupamiento LC-Conceptual.
3. Validar la herramienta computacional desarrollada a través de las pruebas de software.

El **campo de acción** se enmarca en: El Razonamiento Basado en Casos utilizando el algoritmo de agrupamiento LC-Conceptual.

Se propone como **Idea a Defender** que: El desarrollo de un sistema inteligente integrando el RBC y el algoritmo LC-Conceptual permite apoyar la toma de decisiones en los SBC cuando se presentan datos mezclados e incompletos.

Para el cumplimiento del objetivo general se plantean las siguientes **tareas de Investigación:**

1. Análisis de los referentes teóricos de los SBC.
2. Caracterización de los métodos de adaptación en los SBC.
3. Análisis de los conceptos generales del Aprendizaje no supervisado con el enfoque lógico-combinatorio.
4. Análisis del algoritmo de agrupamiento LC-Conceptual.
5. Selección de la metodología de desarrollo de software a utilizar en el diseño del sistema.
6. Selección del entorno de desarrollo y lenguaje de programación a utilizar en la implementación del sistema.
7. Implementación del algoritmo para la organización de la Base de casos.
8. Implementación del algoritmo para la selección de rasgos.
9. Implementación de algoritmos para el acceso y recuperación de los casos semejantes.
10. Implementación de un método para la adaptación de las soluciones utilizando agrupamiento conceptual.
11. Implementación de un método para el almacenamiento de los casos en la Base.
12. Realización de pruebas que garanticen un correcto funcionamiento del software.

Métodos teóricos:

Análisis síntesis: permite realizar el estudio teórico de la investigación y la extracción de los elementos más importantes acerca del funcionamiento y proceso de desarrollo de los Sistemas Basados en Casos, utilizando además el agrupamiento conceptual.

Histórico-lógico: este método se utiliza para estudiar el comportamiento de los Sistemas Inteligentes, específicamente de los sistemas Basados en Casos a través de los años y analizar su funcionamiento con el fin de dar cumplimiento al objetivo de la investigación.

Modelación: facilita la confección de los diferentes modelos y artefactos generados además se utiliza durante la implementación del Sistema Inteligente Basado en Casos para el apoyo a la Toma de Decisiones utilizando el algoritmo LC-Conceptual.

El presente trabajo de diploma está estructurado en un total de 3 capítulos:

Capítulo 1. *Referentes Teóricos de los Sistemas Basados en Casos y el Agrupamiento Conceptual:* se realiza una descripción de los módulos del Razonamiento Basado en Casos y los algoritmos de agrupamiento conceptual en el marco del enfoque lógico combinatorio, además se fundamenta la selección de la metodología y tecnologías de desarrollo de software.

Capítulo 2. *Descripción de la propuesta de solución:* se presenta la propuesta de solución para la herramienta y se realizan las fases de exploración, planificación, diseño e implementación de la metodología de software, así como los artefactos generados por cada una de estas fases.

Capítulo 3. *Pruebas y experimentación:* se evidencian los resultados de las pruebas unitarias para las funcionalidades y las pruebas de aceptación para las historias de usuario, además se muestran los resultados de la experimentación obtenidos con la base de datos seleccionada.

CAPÍTULO 1: Referentes Teóricos de los Sistemas Basados en Casos y el Agrupamiento Conceptual

En el desarrollo de este capítulo se abordan los principales conceptos relacionados con los Sistemas Basados en Casos, realizando una descripción de los módulos que componen el ciclo de vida del Razonamiento Basado en Casos. Se definen los elementos del enfoque lógico combinatorio en el marco de la clasificación no supervisada, particularizando en el algoritmo LC-Conceptual que se corresponde con este enfoque. Además se fundamenta la selección de la metodología de desarrollo de software, la tecnología y el lenguaje de programación a utilizar en la construcción de la herramienta informática

1.1 Razonamiento Basado en Casos

El RBC es el método de solución de problemas de los SBC. Representa un nuevo método para resolver problemas no estructurados, en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos. (Gálvez Lio, 2008)

El RBC significa razonar sobre una base de experiencias o casos previamente analizados, de esta manera el solucionador de problemas recuerda situaciones previas similares a la actual y las utiliza para ayudar a resolver el nuevo problema. Cada caso almacenado en la base de casos tiene características denominadas rasgos, y estos a su vez tienen un dominio de definición el cual no es más que aquellos valores que puede tomar.

1.1.1 Tipos de Sistemas Basados en Casos

Hay dos tipos de sistemas con RBC: los sistemas interpretativos y los resolvedores de problemas. En el estilo de solucionadores de problemas se derivan las soluciones de los nuevos problemas usando las soluciones dadas a los problemas semejantes almacenados. En los interpretativos las situaciones nuevas se evalúan en el contexto de las situaciones viejas.

a) Sistemas interpretativos (Gálvez Lio, 2008)

Son muy útiles cuando el problema no está bien comprendido, son utilizados en sistemas de clasificación, evaluación, cuando es necesaria la justificación, la interpretación o la predicción de los efectos de una decisión en un problema. En estos sistemas se toma la situación planteada y se soluciona con un argumento fundamentando la solución.

b) Sistemas resolvedores de problemas (Gálvez Lio, 2008)

Un sistema de este tipo se aplica cuando aun teniéndose la solución de los casos previos, no es suficiente para dar solución al nuevo problema; se necesita modificar el caso para adecuarlo al problema actual, se caracteriza por tratar de encontrar una solución mejor aún cuando ya esté disponible una. Es muy útil para resolver tareas de diseño y planificación.

1.1.2 Arquitectura del Razonamiento Basado en Casos

Un SBC tiene dos componentes principales: una base de casos y un resolvedor de problemas. La BC[4] contiene las descripciones de los casos analizados. Cada caso puede describir un episodio o un conjunto de episodios relacionados. Cada caso de la BC contiene la descripción de este que se resume al problema y la solución del mismo. El resolvedor tiene a su vez dos componentes: un recuperador de casos y un razonador. Estos componentes se pueden ver en la figura 1.1.

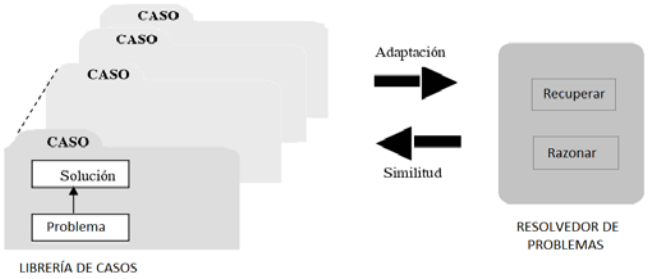


Figura 1.1 Arquitectura de un Sistema Basado en Casos. (Ochoa, y otros, 2006)

Resolver un problema depende del tipo de SBC. En el tipo solucionador de problemas el proceso que se ejecuta es recordar un caso y adaptar su solución, por tanto se deberá resolver aplicando un algoritmo de adaptación. En el interpretativo, es recordar un caso y evaluar el problema nuevo basado en su solución, entonces se realizará un procedimiento de justificación.

1.1.3 Descripción de los Módulos de un Sistema Basado en Casos

Dada la descripción de un nuevo problema, el recuperador de casos identifica un caso semejante a este y la solución del caso recuperado se propone como solución potencial del nuevo problema. Seguidamente se realiza un proceso de adaptación o reutilización en el cual se adecua la vieja solución a la nueva situación. Posteriormente se realiza un análisis crítico evaluativo en el que se revisa la solución resultante de la adaptación. Y por último se realiza el proceso de recordar, este consiste en almacenar el caso en la BC. En la figura 1.2 se puede apreciar el ciclo de un RBC descrito anteriormente.

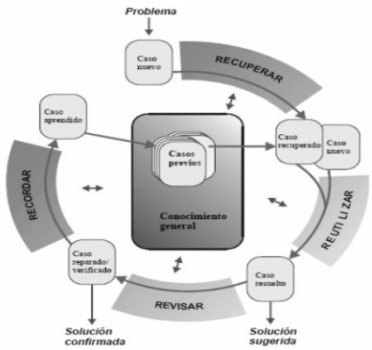


Figura 1.2 Ciclo de un RBC. (Bregón , y otros, 2005)

[4] **BC:** Base de Casos

1.1.3.1 Módulo recuperador de casos

Básicamente lo que se hace en esta fase es buscar en la BC aquellas situaciones que son similares al problema.

Para lograr que la efectividad y velocidad de recuperación no sea afectada por el volumen de la BC, la organización tiene que permitir acceder eficientemente a los elementos de memoria relevantes al problema. Por eso la organización de la memoria está estrechamente vinculada con el método de búsqueda. La medida en que se alcance una combinación eficiente de ambas componentes determinará el éxito o el fracaso del sistema con esta clase de razonamiento.

Para la recuperación o selección de los casos almacenados en la BC que son semejantes al problema actual se pueden realizar dos métodos (Gálvez Lio, 2008):

1. Recuperación por semejanza parcial.

En el este proceso se emplea una función de semejanza, que determina una medida numérica del grado de similaridad que tiene un caso con respecto al problema a resolver.

2. Recuperación por analogía.

La recuperación por analogía busca un caso de la base, cuya descripción se pueda hacer igual a la del problema actual aplicando un reemplazamiento de los valores de los rasgos que tienen valores diferentes, a partir de la correspondencia entre valores definida en una red semántica.

1.1.3.2 Módulo de adaptación o reutilización

Una vez seleccionados los casos de la base, semejantes al problema actual, y teniéndose la descripción de dicho problema, se desea construir la solución partir de las soluciones de los casos semejantes a este. Para ello se necesita la selección de un método para encontrar aquellos valores para los rasgos desconocidos.

Principales métodos de adaptación (Méndiz Noguera, 2007):

1. Adaptación “nula”

Estrictamente hablando, no es ni estructural ni derivacional. Es la primera técnica, y ciertamente la más simple, es no hacer nada y simplemente aplicar cualquiera que sea la solución recuperada, a la nueva situación. Esta es útil en tareas donde el razonamiento necesario para una aplicación puede ser muy complejo, y la solución en sí misma es muy simple.

2. Soluciones parametrizadas

La idea es que cuando se recupera un caso para una entrada concreta, las descripciones del nuevo problema y del viejo se comparan con respecto a unos parámetros especificados. Las diferencias se usan para modificar los parámetros de la solución en las direcciones apropiadas. Cada parámetro del problema se asocia con uno o más parámetros de la solución. Una limitación de esta técnica es que las soluciones parametrizadas son válidas para modificar una solución pero no para crear una nueva solución.

3. Abstracción y reespecialización

La idea es la siguiente: si un fragmento de la solución recuperada no es aplicable para el caso actual, se buscan abstracciones de ese fragmento de solución que no tengan la misma dificultad. A continuación, se reespecializa, es decir, se intentan aplicar a la situación actual otras especializaciones de la abstracción. Se obtienen de esta forma hermanos o parientes colaterales del concepto original.

4. Adaptación basada en crítica

Una crítica busca alguna combinación de características que puede causar un problema en un plan. Existen estrategias de reparación asociadas con diferentes situaciones problemáticas. Un plan que debe alcanzar varios objetivos simultáneamente se deriva poniendo juntos los planes que podrían alcanzar cada objetivo independientemente. Las críticas pueden entonces comprobar si algún plan interfiere con otro o si hay planes redundantes.

Las técnicas descritas hasta el momento están comprendidas dentro de los métodos de adaptación estructural. Operan directamente sobre las viejas soluciones para producir las nuevas.

5. Reparticularización

La reparticularización (‘Reinstantiation”) es un método derivacional. No opera sobre la solución original sino sobre el método que fue usado para generar dicha solución. Reparticularización significa reemplazar un paso en el plan que generó la solución y reejecutarlo en el contexto de la situación actual. La potencia de la reparticularización está limitada por la potencia de planificación del sistema, puesto que reparticularizar un plan es planificar.

A partir del análisis del componente de adaptación en los SBC es posible afirmar que la toma de decisiones en estos sistemas está en correspondencia con el mecanismo de adaptación que implemente el razonador.

1.1.3.3 Módulo de revisión o evaluación de soluciones

En este módulo es donde se realiza la revisión de la solución encontrada en el proceso de adaptación, es necesario que esta sea la correcta, aquella que cumple con todas las demandas del problema, aunque no sea la mejor solución.

La calidad de las soluciones del RBC depende de las experiencias que se tienen, de la habilidad para entender nuevas situaciones en términos de viejas experiencias, y de la destreza en la adaptación y evaluación de soluciones. (Gálvez Lio, 2008) Este módulo es de vital importancia ya que un razonador requiere de retroalimentación, el necesita conocer qué soluciones dadas por él han sido correctas y cuales no, para que no repita sus faltas e incremente sus capacidades.

1.1.3.4 Módulo de almacenamiento o de recordar

En este módulo se guarda en la base el caso recién analizado, para ello es necesario que se almacene de forma organizada, esto permite al recuperador una mayor facilidad de búsqueda cuando necesite extraer casos para solucionar un nuevo problema.

Lo esencial para el trabajo de un SBC es que todos los casos relevantes (similares, semejantes) al nuevo problema puedan recuperarse eficientemente desde la BC. Esto depende directamente de la forma en que están organizados los casos en la base, a esa forma se le denomina modelo de la BC.

Existen diversos enfoques para organizar y modelar esta, dentro de ellos los más utilizados son los siguientes (Gálvez Lio, 2008):

a) Organización de los casos en una estructura plana.

En este modelo los casos son unidades de información que se organizan en forma secuencial en una lista, arreglo o fichero. Tiene la ventaja de ser poco costoso al añadir casos a la memoria, pero todos los casos son comparados durante la recuperación, por tanto tienen la desventaja de que este proceso si sea costoso. Por eso, este enfoque se utiliza solamente en BC pequeñas.

b) Organización de los casos en una estructura jerárquica compartida.

En este modelo los casos están ubicados en nodos de un árbol. La estructura subdivide los casos de acuerdo a los atributos que comparten. Esto se realiza situando atributos comunes en los nodos internos del árbol. Todos los casos que compartan valores se sitúan debajo de dicho nodo.

c) Organización de los casos en redes de discriminación.

Cada nodo es una pregunta que subdivide el conjunto de casos y cada subnodo es una respuesta distinta a la pregunta, esta puede traer otra pegunta para más casos asociados a el. Las preguntas más importantes se colocan en la parte superior. Tiene la ventaja de mejorar la eficiencia en el acceso, pero añadir casos requiere modificar la red, lo que es muy costoso computacionalmente, además algunos casos pueden no ser considerados dado el orden de las preguntas.

1.2 Enfoque Lógico-Combinatorio

Este enfoque se basa en la idea de que la modelación del problema debe ser lo más cercana posible a la realidad del mismo, sin hacer suposiciones que no estén fundamentadas. Uno de los aspectos esenciales del enfoque es que las características utilizadas para describir a los objetos de estudio deben ser tratadas cuidadosamente. (Carrasco Ochoa, 2009). En el Enfoque Lógico Combinatorio se supone que los objetos pueden ser descritos por rasgos numéricos y no numéricos y pueden ser procesados por funciones numéricas. (Ruíz Shulcloper, y otros, 1995)

En este enfoque es necesario identificar el problema, que puede ser de clasificación supervisada, no supervisada o de selección de variables. La primera consiste en clasificar nuevos objetos basándose en la información de una muestra ya clasificada, y la segunda: dada una muestra no

clasificada encontrar la clasificación de la misma. En el caso de la selección de variables se realiza para la clasificación y para la representación de estas. (Carrasco Ochoa, 2009)

1.3.1 Teoría de testores para la selección de rasgos

Esta teoría permite disminuir el número de variables del problema, de esta forma facilita el trabajo debido a que el número de estas puede ser bastante grande, se aplica a través del conjunto de testores que componen dicho problema.

Un testor es una colección de características que discriminan las descripciones de objetos pertenecientes a diferentes clases, y es mínimo en el orden parcial determinado por la inclusión de conjuntos. (Ruíz Shulcloper, y otros, 1999) En esencia, se buscan todos los subconjuntos de variables discriminantes minimales, con estos se evalúa la relevancia de cada variable y se seleccionan aquellas con mayor relevancia, o sea un testor es un subconjunto de características, que permite la diferenciación completa de objetos de diferentes clases.

1.3.1.1 Conceptos Básicos

Definición 1.1 Se llama Rasgo a aquella característica de un objeto ($X_s(O)$). Este a su vez contiene un dominio de definición o conjunto de valores admisibles (M_i), que no es más que los valores que este puede tomar.

Definición 1.2 Una descripción de un objeto es un n-uplo $I(O) = (X_j(O), \dots, X_n(O))$ donde $X_i(O)$ es el valor del rasgo X_i en el objeto O y pertenece al dominio de definición de este (M_i). Entonces $I(O)$ es la descripción completa del objeto O. (Ruíz Shulcloper, y otros, 1999)

Definición 1.3 Se llama criterio de comparación de valores del rasgo X_i a una función C_j (también representada como δ_j), donde C_j es un criterio de disimilaridad o similaridad entre los rasgos de dos elementos. (Pons Porrata, 2004)

Ejemplos de Criterios de comparación por rasgos (Pons Porrata, 2004)*:*

1. $$C_s\big(X_s(O_i), X_s(O_j)\big) = \begin{cases} 1 \ si \ X_s(O_i) = X_s(O_j) \vee X_s(O_i) = * \ \vee X_s(O_j) = * \\ \\ 0 \ en \ otro \ caso \end{cases}$$

Este es un criterio Booleano, conocido como criterio de comparación de igualdad estricta, donde el valor 1 significa que los valores son coincidentes y 0 que son diferentes; mientras que "?" denota la ausencia de información que también se representa con "*".

2. $$C_s\big(X_s(O_i), X_s(O_j)\big) = \begin{cases} 1 \ si \ X_s(O_i), X_s\big(O_j\big) \in \big[a_p, a_{p+1}\big] \vee X_s(O_i) = * \ \vee X_s\big(O_j\big) = * \\ \\ 0 \ en \ otro \ caso \end{cases}$$

Este es un criterio Booleano, donde el valor 1 significa que ambos valores pertenecen al mismo intervalo, mientras que toma valor 0 en cualquier otro caso que se presente.

3. $C_s\big(X_s(O_i), X_s(O_j)\big) = \begin{cases} 1 \; si \; \left|X_s(O_i) - X_s(O_j)\right| \leq \varepsilon \vee X_s(O_i) = * \vee X_s(O_j) = * \\ \\ 0 \; en \; otro \; caso \end{cases}$

Este es un criterio Booleano, conocido como criterio de comparación de error admisible, donde **ε** es un umbral asociado a cada rasgo Xi y **ε** ≥ 0. Por supuesto que para poder usar este criterio de comparación es necesario que los valores de los rasgos que estemos considerando admitan las operaciones señaladas, es decir, que sean números.

4. $C_s\big(X_s(O_i), X_s(O_j)\big) = \begin{cases} 1 \; si \; X_s(O_i), X_s\big(O_j\big) \in A_P \, , \vee X_s(O_i) = * \vee X_s\big(O_j\big) = * \\ \\ 0 \; \; en \; otro \; caso \end{cases}$

Este es un criterio Booleano, donde la variable X_s toma valores iguales a la unión finita de los conjuntos A_p, esto significa que los valores pertenecen al intervalo señalado, mientras que toma valor 0 cuando no pertenecen a este.

5. $C_s\big(X_s(O_i), X_s(O_j)\big) = 1 - \dfrac{|X_s(O_i) - X_s(O_j)|}{\Delta X_s}$

Este criterio exige que la variable sea cuantitativa y que no presente ausencia de información. Los cuatro ejemplos anteriores fueron ejemplos de criterios de comparación Booleanos de valores de una variable y este último es un criterio de comparación real de valores de una variable, siendo $\Delta X = Max(M_i) - Min(M_i)$, donde M_i es el conjunto de valores admisibles de X_i o dominio de definición de este. También se denota $\Delta X = Max(X_i) - Min(X_i)$, es decir, el diámetro del conjunto de valores admisibles de la variable X_i.

No siempre es posible definir un tipo particular de criterio de comparación. El uso de estos tipos está en dependencia del problema específico que se esté modelando. Además, su selección obviamente condicionará todo el trabajo posterior.

Existen varios tipos de criterios de comparación, los siguientes son unos de los más utilizados:
Criterio booleano: Es aquel criterio que responde a una respuesta única, es cierto o falso y se aplica tanto a los rasgos literales como a los numéricos.

Criterio Ordinal: Este criterio se aplica tanto a los rasgos numéricos como a los literales, la particularidad radica en que deben tener un orden lógico, los números evidentemente se pueden ordenar, pero en el caso de los literales deben ser ordenables.

Criterio Nominal: Es un criterio que se sujeta a los rasgos literales solamente, donde los elementos de cada rasgo no poseen un orden lógico.

Criterio K-Valente: Este criterio se adecúa a los literales, y tienen la particularidad de que el objeto puede tener uno o más elementos del dominio del rasgo.

Criterio Aritmético: Este criterio solo se utiliza para los rasgos numéricos únicamente.

Definición 1.4 El aspecto de la selección de los casos semejantes de la memoria se focaliza en el uso de una función de semejanza, esta da como resultado un valor entre 0 y 1 que expresa cuan semejante es el problema a cada uno de los casos contenidos en la BC. Para su trabajo utiliza una función de comparación por rasgos C_j (también conocido por δ_j), esta compara el valor de un rasgo del problema con el valor de ese rasgo en un caso, obtiene un valor (generalmente entre 0 y 1) que expresa el grado de similaridad o disimilaridad entre ambos valores. Una de las funciones de semejanza más utilizada es la siguiente (Gálvez Lio, 2008):

$$\beta(P,R) = \sum_{i=1}^{n} W_i * \delta_j(P_i, R_i)$$

Donde P representa la información del problema y R la información del caso de la BC. La sumatoria recorre los n rasgos que caracterizan el problema e incluye el producto del peso de cada rasgo W_i con el valor de la función de comparación por cada rasgo (δ_j o C_j).

Debe garantizarse que: $\sum_{i=1}^{n} W_i = 1$

Los objetos y sus rasgos pueden ser representados en una matriz cuyas filas son las descripciones de los objetos y las columnas son los rasgos.

Definición 1.5 Se llama MI[5] a aquella que contiene las descripciones de los objetos. (Ruíz Shulcloper, y otros, 1999)

Definición 1.6 Una MA[6] es aquella que contiene las descripciones de los objetos así como sus respectivas clases de pertenencia por cada objeto. (Pons Porrata, 2004)

Definición 1.7 Dada una MA y dados los criterios de comparación para cada rasgo, se llama MD[7] a la matriz booleana que se obtiene como resultado de comparar todos los pares de objetos de clases distintas. (Ruíz Shulcloper, y otros, 1999)

Definición 1.8 Sean i_p, i_q dos filas de una MD, se dice que la fila i_p es superfila de i_q si esta última es subfila de i_p. La fila i_q es subfila de i_p si ∀1 en la fila i_p existe un 1 y al menos un 0 en la fila i_q, pero ∀0 en i_p solo puede haber un 0 en i_q. (Ruíz Shulcloper, y otros, 1999)

Definición 1.9 Se llama MB[8] a aquella que está formada por las filas básicas de una MD. Una fila es básica siempre que no tenga una subfila asociada a ella. (Ruíz Shulcloper, y otros, 1999)

[5] **MI:** Matriz Inicial.
[6] **MA:** Matriz de Aprendizaje.
[7] **MD:** Matriz de Diferencia.
[8] **MB:** Matriz Básica.

Definición 1.10 El subconjunto $\tau = \{x_{i_l}, \dots, x_{i_s}\}$ de rasgos de una MA es un testor si al eliminar de su MB todas las columnas, excepto las correspondientes a los elementos de τ, no existe fila alguna completa de ceros. (Pons Porrata, 2004)

Un testor se obtiene a partir de una MB, se conforma tomando aquellos rasgos por fila que tengan valor uno.

Definición 1.11 El peso de un rasgo puede ser determinado a partir de la fórmula (Ruíz Shulcloper, y otros, 1999): $\varepsilon_i(x_i) = \alpha\, F(x_i) + \beta\, L(x_i)$, debe cumplirse que $\alpha > 0$, $\beta > 0$ y $\alpha + \beta = 1$, estos dos parámetros ponderan la participación o influencia de F_i (frecuencia de aparición) y L_i (longitud de los testores).

$$F_{x_i} = \frac{|TT(x_i)|}{TT} \qquad L(x_i) = \frac{\sum_{t \in |TT(x_i)|} \frac{1}{|t|}}{|TT(x_i)|}$$

$|TT(x_i)|$: es el número de testores donde aparece el rasgo i.

TT: es el número de testores.

$|t_i|$: es el número de rasgos que forman el testor t_i.

Definición 1.12 La importancia de un testor es la suma de las importancias o pesos ε_i de los rasgos que componen al testor t_i (Ruíz Shulcloper, y otros, 1999): $\psi_i(t_i) = \sum_{i=1}^{|t_i|} \varepsilon(x_i)$.

Definición 1.13 Una MS[9] es la matriz cuadrada que contiene la comparación de todos los objetos entre sí, a partir de los criterios de comparación por cada uno de los rasgos que describen a los objetos y de una función de semejanza entre objetos. (Pons Porrata, 2004)

1.3 Aprendizaje no supervisado

En las ciencias de la computación el aprendizaje automático o aprendizaje de máquinas es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. El aprendizaje no supervisado es un método de aprendizaje automático donde todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas. (Gálvez Cordero, y otros, 2004)

1.4 Algoritmos de Agrupamiento

Dado un conjunto de objetos definidos en términos de un conjunto de rasgos, los algoritmos de agrupamiento intentan construir particiones de este conjunto, donde la semejanza intra-grupo sea máxima y la semejanza inter-grupos sea mínima. (Pons Porrata, 2004)

1.4.1 Conceptos Básicos

Sea ζ una colección de objetos y S una función de semejanza entre objetos, es decir, para todo

[9] **MS:** Matriz de Semejanza.

$O_i, O_j \in \zeta$ se cumple que $S(O_i, O_j) = S(O_j, O_i)$. Sea, además, β_0 un umbral de semejanza definido por el usuario.

Definición 1.14 El umbral es la cota inferior de los posibles valores de semejanza existentes para cada uno de los objetos con respecto a otro, se obtiene a partir de una MS y puede ser introducido por el especialista o calculado, por ejemplo, de la siguiente manera (Ruiz Shulcloper, 2013):

$$A)\ \beta_0 = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta(I(O_i), I(O_j))$$

$$B)\ \beta_0 = Max_{\substack{i=1\ldots m-1 \\ i \neq j}} \{Min_{j=i+1\ldots m}\{(\beta(I(O_i), I(O_j))\}\}$$

$$C)\ \beta_0 = Min_{\substack{i=1\ldots m-1 \\ i \neq j}} \{Max_{j=i+1\ldots m}\{(\beta(I(O_i), I(O_j))\}\}$$

Definición 1.15 Un criterio agrupacional Π es aquel que define la estructura del universo a conformar. (Ruiz Shulcloper, 2013)

Definición 1.16 Se dice que dos objetos O_i y O_j son $\beta_0 - semejantes$ si $S(O_i, O_j) \geq \beta_0$. Si $\forall O_j$ de la colección de objetos ζ se cumple que $S(O_i, O_j) < \beta_0$ entonces O_i se denomina $\beta_0 - aislado$. (Pons Porrata, 2004)

Definición 1.17 Se dice que NU ⊆ ζ, NU ≠ ∅ es una componente $\beta_0 - conexa$ si se cumple que (Pons Porrata, 2004):

1. $\forall O_i, O_j \in NU, O_i \neq O_j, \exists O_{i_1}, \ldots, O_{i_q} \in NU \left[O_i = O_{i_1} \wedge O_j = O_{i_q} \wedge \forall p \in \{1, \ldots, q-1\}\ S(O_p, O_{p+1}) \geq \beta_0\right]$
2. $\forall O_i \in \zeta \left[O_j \in NU \wedge S(O_i, O_j) \geq \beta_0\right] \rightarrow O_i \in NU$
3. Todo elemento $\beta_0 - aislado$ es una componente conexa (degenerada).

El criterio agrupacional $\boldsymbol{\beta_0 -}$ **conexo** Realiza una búsqueda de aquellos objetos que son semejantes unos con otros y los coloca en un mismo grupo. Todos los objetos cuyas semejanzas sean igual o estén por encima del umbral serán considerados semejantes.

Definición 1.18 Se dice que NU ⊆ ζ, NU ≠ ∅, es un conjunto $\beta_0 - compacto$ si se cumple que (Pons Porrata, 2004):

1. $\forall O_j \in \zeta \left[O_i \in NU \wedge max\{S(O_i, O_t)\} = S(O_i, O_j) \geq \beta_0\right] \rightarrow O_j \in NU, O_t \in \zeta, O_t \neq O_i$
2. $\left[max\{S(O_p, O_i)\} = S(O_p, O_t) \geq \beta_0 \wedge O_t \in NU\right] \rightarrow O_p \in NU, O_i \in \zeta, O_i \neq O_p$

3. $|NU|$ es mínimo.
4. Todo elemento $\beta_0 - aislado$ constituye un conjunto compacto (degenerado).

El criterio $\boldsymbol{\beta_0 -}$ **Compacto** Realiza una búsqueda de aquellos objetos que son más semejantes entre ellos y los coloca en un mismo grupo. Para ello escoge de los objetos semejantes que cumplen con la condición de que su valor de semejanza es mayor o igual al umbral, aquellos que tengan mayor valor, pero si alguno de los valores que cumplen con la condición de ser mayor que el umbral, aun así no son los mayores, tienen la posibilidad de estar en el mismo grupo que el objeto en cuestión si tienen a este como mayor semejante a él.

Existen numerosos algoritmos de agrupamiento, pero la gran mayoría únicamente estructuran los grupos sin dar las características del conjunto de elementos.
Los algoritmos de agrupamiento conceptuales surgen como una alternativa en la clasificación no supervisada, para la solución de algunos problemas donde es necesario obtener los conceptos de los grupos resultantes. (Guerra Gandón, y otros, 2012)

1.4.2 Algoritmos de Agrupamiento Conceptual

Los algoritmos conceptuales son uno de los paradigmas existentes en la actualidad dentro de la clasificación no supervisada. Este tiene un gran impacto debido principalmente a su capacidad de revelar estructuraciones de los datos y sus conceptos, lo cual libera al especialista de la interpretación de los resultados que en muchas ocasiones es engorrosa.

Los algoritmos de agrupamiento conceptual se componen de dos tareas fundamentales (Reyes González, y otros, 2014):

1. La estructuración o determinación extensional: Se lleva a cabo el proceso de agrupar entidades, en el que se determinan grupos a partir de una colección de objetos.
2. La caracterización o determinación intencional: Se determina el concepto de cada grupo de la estructuración, surge en 1979 por Mishalski.

Los algoritmos de agrupamiento conceptual se pueden dividir en dos grandes grupos: los algoritmos incrementales y los no incrementales:

- ✓ Los algoritmos incrementales basan su funcionamiento en la adaptación de los grupos (o conceptos) con los nuevos objetos que se le van presentando, es decir, cada vez que se analiza un nuevo objeto éste se clasifica, mediante cierta estrategia, en uno de los grupos ya existentes o se crean nuevos grupos. Por lo general, estos algoritmos se han desarrollado para los casos en que el conjunto de objetos a estructurar no está completamente dado, es decir, es dinámico. (Guerra Gandón, y otros, 2012) Ejemplos de algoritmos de este tipo son: UNIMEM, COBWEB, Galois, EPAM, CLASSIT y LINNEO+ (Pérez Suárez, y otros, 2014).

- ✓ Por otro lado, los algoritmos no incrementales estructuran una muestra de objetos utilizando el conjunto de datos completo. (Guerra Gandón, y otros, 2012) Ejemplos de estos algoritmos son: CLUSTER/PAF, CLUSTER/2, y CLUSTER/S, WITT, K-Means conceptual, LC conceptual y RGC (Pérez Suárez, y otros, 2014).

1.4.2.1 Algoritmo LC-Conceptual

El algoritmo LC-Conceptual tiene la capacidad de operar tanto con atributos cualitativos como cuantitativos, además de admitir la ausencia de información y no requiere especificar el número de agrupamientos a priori ni semillas, lo que es muy ventajoso, pues es deseable organizar la BC en agrupaciones considerando las semejanzas entre los casos sin imponer una determinada cantidad de grupos a formar. El agrupamiento se realiza a través de la semejanza entre objetos y los conceptos construidos son propiedades lógicas apoyadas en los rasgos que describen a los objetos en análisis, estos conceptos son fáciles de interpretar por los usuarios y facilita comprender las características fundamentales.

Debido a sus características, este algoritmo permite resolver una gama amplia de problemas de estructuración conceptual que otros algoritmos no pueden resolver satisfactoriamente, aunque la etapa de caracterización que ejecuta para formar los conceptos es computacionalmente costosa, principalmente debido al cálculo de los testores, la determinación de la descripción extensional de los grupos es computacionalmente poco costosa.

Descripción del algoritmo

Sea $O = \{O_1, O_2, \dots, O_n\}$ una colección de objetos, descritos en términos de un conjunto de atributos que pueden ser cualitativos, numéricos o incluso puede desconocerse su valor. Sea además S una función tal que $S(O_i, O_j)$ establece el grado se semejanza que tiene un par de objetos $O_i, O_j \in O$. El algoritmo LC representa la colección de objetos como un grafo en el cual los objetos de la colección son los vértices del grafo y existe una arista entre dos objetos O_i y O_j si $S(O_i, O_j) \geq \beta$ o $S(O_j, O_i) \geq \beta$; donde β es un número real en el intervalo [0, 1] que determina el umbral mínimo de semejanza que deben tener dos objetos para ser considerados como semejantes. A partir del grafo anterior, los grupos son aquellos subconjuntos de O que satisfacen un conjunto de propiedades relativas a las semejanzas entre los objetos que constituyen cada subconjunto; estas propiedades se conocen como criterio de agrupamiento. El algoritmo LC propone dos criterios de agrupamiento que forman agrupamientos disjuntos. Uno de estos criterios forma grupos que son componentes β-conexas y el otro, grupos que son conjuntos β-compactos. (Pereira, y otros)

A partir de los grupos formados se construyen las propiedades (conceptos) que caracterizan a cada grupo de objetos. Se forma una matriz de aprendizaje MA a partir de MI y de los $K_1, \dots, K_c$, luego se calcula el conjunto de los testores más relevantes de MA $\tau = \{t_1, \dots, t_p\}$.

Para cada clase $K_i, i = 1, \dots, c$ se calcula la estrella $G_\tau(K_i \backslash K_1, \dots, K_{i-1}, K_{i+1}, \dots, K_c)$ que estará formada por p l-complejos (tantos como testores relevantes). Para el cálculo de los $^{R}\big|_{K_i}$ en cada $X \in t_q, t_q \in \tau$ se emplea el operador de refunión condicionada RUC. Los conceptos que caracterizan a cada agrupamiento K_i serán los l-complejos obtenidos en el paso anterior. (Reyes González, 2014)

1.5 Metodología de desarrollo de Software

Una metodología de desarrollo de software no es más que un conjunto de procedimientos, técnicas, herramientas y un soporte documental que guía a los desarrolladores en la realización del nuevo software. (Pressman , 2002) Debido a la responsabilidad que tiene sobre la vida del software es necesario que sea elegido el enfoque adecuado para el desarrollo eficiente de este. Existen dos enfoques: los ágiles y los tradicionales o prescriptivos.

Para la selección del enfoque se realiza el método Boehm y Turner (Boeras Velázquez, y otros, 2012) también conocido como método de la estrella. Este caracteriza al proyecto de software a partir de 5 criterios y realiza una estimación acerca de cuan ágil o prescriptivo debe ser el enfoque a utilizar. Tiene 5 ejes, en cada uno se coloca un criterio, estos se explican a continuación (Boeras Velázquez, y otros, 2012):

Tamaño: Este criterio se utiliza para representar el número de personas involucradas en el proyecto. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.

Criticidad: Se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.

Dinamismo: Representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

Personal: Representa la proporción del personal con experiencia alta, media y baja. Los métodos orientados al plan no se ven afectados negativamente por este factor pues no interesa el nivel de experiencia con la que cuenten los miembros del equipo.

Cultura: Las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones.

Es importante tener en cuenta el comportamiento de cada criterio en el proyecto, en correspondencia de los valores que alcancen se determinará el enfoque, mientras más cerca esté del centro el enfoque será el ágil, de lo contrario debe usarse un enfoque tradicional.

A continuación se describe el comportamiento de los criterios en el proyecto:

Tamaño: El equipo de desarrollo está formado por 2 estudiantes de quinto año para la implementación de un total de 17 funcionalidades con ciertos elementos de complejidad, características que permiten clasificar el equipo de desarrollo y al sistema como pequeños. Debido a esta descripción será posible ubicar el punto cercano al centro.

Criticidad: El equipo de desarrollo tiene una elevada responsabilidad con la calidad del producto a obtener, debido al impacto social del mismo. Este sistema será multipropósitos por tanto permitirá ser usado tanto en el campo de la medicina, o en la realización de diversas tareas en la educación, y otros. El valor para este criterio dentro de la estrella se etiqueta como medio, debido a que los efectos por errores del producto aunque no provocará pérdidas de vidas, provocarían otras.

Dinamismo: Pueden aparecer cambios en los requerimientos del sistema en cualquier fase del proceso de desarrollo, esto es un riesgo que se asumirá durante todo el ciclo de desarrollo del software. Para mitigarlo, deben adoptarse mecanismos que faciliten la asimilación y adaptación rápida a dichos cambios. Tomando en cuenta esta necesidad como una de las ventajas que brinda el enfoque ágil, se ha ubicado este punto bien cercano al centro de la estrella.

Personal: El valor para este criterio dentro de la estrella es medio porque los desarrolladores no son programador más experimentado, pero tampoco son programadores poco experimentado, estos dominan el trabajo con la tecnología que se pretende utilizar.

Cultura: Se posee un buen ambiente entre el equipo de desarrollo, existe buena comunicación y confianza entre sus miembros puesto que han trabajado juntos en otras ocasiones y ambos han cursado 3 años de la carrera en el mismo grupo. Las decisiones tomadas son previamente analizadas y consultadas en conjunto. El equipo presenta plena libertad en el desarrollo del proyecto, así como en la toma de decisiones de este, por lo que se distingue el valor de este criterio como pequeño, muy alejado al enfoque formal o prescriptivo.

La Figura 1.3 muestra la representación de los criterios analizados en la Estrella de Boehm y Turner:

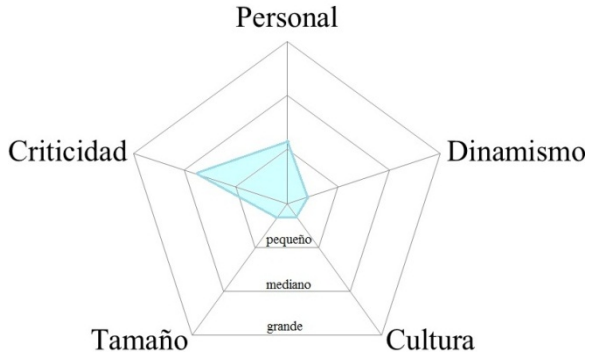


Figura 1.3 Estrella que representa las características del proceso de desarrollo de software según la aplicación de Boehm y Turner. (Elaboración propia)

Como se puede apreciar en la disposición de las aristas, se propone adoptar un enfoque ágil. Actualmente existen numerosas propuestas de metodologías ágiles que se adecuan al desarrollo del Sistema Inteligente Basado en Casos.

Dentro de estas metodologías ágiles se destacan algunas como: las metodologías Crystal (Canós, y otros), en esta el equipo de desarrollo es un factor clave, pero dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).La metodología Lean Development (Canós, y otros) también es propicia, pero en LD[10], los cambios se consideran riesgos, aunque si se manejan adecuadamente se pueden convertir en oportunidades. Por ultimo la metodología XP[11] (Beck, 1999), esta se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico, como es en el caso del Sistema Inteligente Basado en Casos.

1.5.1 Metodología ágil seleccionada. Programación Extrema (XP)

Las metodologías ágiles han ganado popularidad desde que se publicó su manifiesto, siendo XP el miembro del grupo que goza de mayor aceptación en el área. Se selecciona esta metodología puesto que independientemente de ser una de las más exitosas en la actualidad, es especialmente adecuada para proyectos pequeños con requisitos imprecisos y muy cambiantes, en el desarrollo del Sistema Inteligente Basado en Casos los requisitos tienden a cambiar frecuentemente conforme avanza el trabajo. Es una metodología eficiente, de bajo riesgo y flexible, reduce el tiempo de implementación.

La metodología consiste en una programación rápida o extrema centrada en la simplicidad en las soluciones implementadas y versatilidad para enfrentar los cambios, en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. (Canós, y otros)

Al seleccionarse XP como metodología, el proceso de creación del Sistema se verá guiado por las siguientes fases (Beck, 1999):

Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

Planificación: Se deben realizar las historias de usuarios, con la creación de calendarios, se establece la prioridad de cada historia de usuario, y correspondientemente, los programadores

[10] **LD:** Lean Development (Desarrollo de delgado).
[11] **XP:** Extreme Programming (Programación Extrema).

realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

Diseño: El diseño debe ser simple, con soluciones rápidas para reducir el riesgo. El principal artefacto generado en esta fase son las tarjetas Clase, Responsabilidad y Colaboración.

Codificación: En esta fase se realiza la implementación del sistema guiado por las Tareas de Ingeniería y siguiendo estándares de codificación, esta requiere de pruebas adicionales y revisiones de rendimiento. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual.

Prueba: Todos los códigos sin excepción se les deben realizar pruebas de unidad, y una vez realizado esto pueden ponerse en funcionalidad, cuando un error es encontrado se realiza la corrección, se debe realizar las pruebas con frecuencia y los resultados deben ser publicados.

1.6 Tecnologías de Desarrollo

Para el desarrollo del Sistema Inteligente Basado en Casos se ha seleccionado el entorno de desarrollo integrado NetBeans 7.4, así como el lenguaje de programación Java.

1.6.1 Lenguaje de desarrollo

El Sistema Inteligente Basado en Casos es de propósito general y necesita un lenguaje de alto nivel que provea un programa que pueda ser ejecutado en diversas plataformas. Java es un lenguaje de programación de alto nivel de propósito general, orientado a objetos y basado en clases, diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, característica que lo convierte en muy popular y valorado ya que los programas Java se pueden ejecutar en diversas plataformas. Es por estas razones que se elige dicho lenguaje para el desarrollo del software.

1.6.2 Entorno Integrado de Desarrollo

NetBeans IDE[12] es un entorno integrado de desarrollo, una herramienta que posee un soporte óptimo para tecnologías Java. Editor de código rápido e inteligente y gestiona proyectos de manera fácil y eficiente, este es un producto libre y gratuito sin restricciones de uso que brinda una serie de ventajas que lo hacen realmente condicionado, dentro de estas: la agilidad y flexibilidad en la implementación, la creación de aplicaciones multiplataforma, facilidades y garantías para la migración, facilidad de uso, cumplimiento de regulaciones y flexibilidad entre plataformas, además de ser muy compatible con el lenguaje de desarrollo seleccionado. Dadas estas características se determina utilizar la herramienta NetBeans en su versión 7.4.

[12] **IDE:** Integrated Development Environment (entorno integrado de desarrollo).

1.6.3 Weka

WEKA es una herramienta de aprendizaje automático y minería de datos, escrita en lenguaje Java y gratuita. Nativamente Weka trabaja con un formato denominado arff[13] , para que la presente aplicación cargue los datos en este formato es necesario la utilización de la clase Instances para la representación en memoria de una colección de ejemplos descrito por un conjunto de atributos, y el paquete weka.core, específicamente subpaquete weka.core.converters que implementan clases auxilares para leer y escribir datasets desde distintas fuentes de datos (ficheros ARFF, bases de datos).

1.6.4 Herramienta de modelado

Dentro de la familia de las herramientas CASE[14], Visual Paradigm es una aplicación profesional que soporta el ciclo de vida completo del desarrollo de software, presenta licencia gratuita y comercial, es fácil de instalar, actualizar y es compatible entre sus distintas versiones. En desarrollo del sistema se utiliza en diseño del diagrama de clases. (Company Headquarters, 2013)

1.7 Conclusiones parciales del capítulo

A partir del análisis de los referentes teóricos de los Sistemas Basados en Casos y del Agrupamiento Conceptual, específicamente del algoritmo LC-Conceptual se arriba a las siguientes conclusiones:

- ✓ En la literatura científica consultada no se hallaron evidencias de sistemas automatizados para la toma de decisiones preparados para soportar Sistemas Basados en el Conocimiento que permitan el tratamiento de valores mezclados e incompletos y a su vez tengan implementaciones del algoritmo LC-Conceptual.
- ✓ Para la toma de decisiones en dominios de aplicación de datos mezclados y con ausencia de información es adecuado utilizar el enfoque lógico-combinatorio.
- ✓ El algoritmo LC-Conceptual constituye una alternativa a ser empleada en los sistemas basados en casos para resolver el problema de agrupamiento con datos mezclados y ausencia de información.
- ✓ La metodología ágil XP es adecuada para el desarrollo del software, ya que se adapta a las necesidades del mismo por las características del sistema a implementar, así como el entorno de desarrollo integrado NetBeans.

[13] **ARFF:** Acrónimo de Attribute-Relation File Format.
[14] **CASE:** Ingeniería de Software Asistida por Computadoras (Computer Aided Software Engineering)

CAPÍTULO 2: Descripción de la propuesta de solución

En este capítulo se describe la propuesta de solución del Sistema Inteligente Basado en Casos y se realizan las fases de exploración, planificación, diseño e implementación pertenecientes a la metodología XP, generando los artefactos correspondientes a cada una de ellas.

2.1 Características del modelo a desarrollar

La herramienta está basada en el modelo propuesto en (Reyes González, 2014), se apoya sobre las ideas del algoritmo LC-Conceptual fusionado al ciclo de un RBC, consta de dos fases fundamentales, cada una compuesta por conjunto de algoritmos fundamentados en el enfoque lógico combinatorio del reconocimiento de patrones, a continuación se describe:

Fase I. Organización de la Base de Casos

Esta fase consta de dos etapas, inicialmente se realiza la estructuración extensional que es donde se forman los grupos, luego la estructuración intencional donde se determina el concepto de cada grupo formado en la etapa anterior. Esta fase consta de cuatro algoritmos (Reyes González, 2014):

Algoritmo 1: Agrupamiento de los casos.

Entrada: MI

Salida: MA

Paso 1: *Construir la Matriz de Semejanza utilizando una función de semejanza* β.

Paso 2: *Calcular el umbral de semejanza utilizando un criterio* β_0*:*

Paso 3: *Agrupar siguiendo un criterio agrupacional II (Ruiz-Shulcloper 2013).*

Algoritmo 2: Cálculo de los testores.

Entrada: MA

Salida: Conjunto de Testores (TT[15]*)*

Paso 1: *Calcular la Matriz de Diferencia*

Paso 2: *Calcular la Matriz Básica*

Algoritmo 3: Cálculo de los testores de mayor utilidad.

Entrada: TT

Salida: Conjunto de testores de mayor utilidad (TT´)

Paso 1: *Calcular el peso* ε_i *de los r asgos* x_i *que aparecen en la familia de testores.*

[15] **TT:** Las filas de la MB conforman el conjunto de Testores (TT).

Paso 2: *Seleccionar el conjunto de rasgos relevantes ($\psi_i(t_i)$).*

Paso 3: *Seleccionar los p*[16] *testores de mayor $\psi_i(t_i)$.*

Algoritmo 4: Construcción de los conceptos.

Entrada: MA, TT´

Salida: Conjunto de los l-complejos para cada grupo K_i (LC´).

Paso 1:*Para cada clase K_i, donde i = 1,…, r calcular la estrella $G_t\ (K_i/K_1, \dots, K_{i-1}, K_{i+1}, \dots, K_r)$ que estará formada por los $|TT'|$ l-complejos.*

Paso 2: *Los conceptos que caracterizan a cada agrupamiento K_i serán los l-complejos obtenidos en el paso anterior.*

Fase II. Ciclo de funcionamiento del RBC

En esta fase se realiza el ciclo del razonador basado en casos, la cual comprende el acceso y recuperación de los casos más semejantes y la aplicación de un nuevo procedimiento para la adaptación de las soluciones teniendo en cuenta los conceptos que describen los casos similares recuperados de la BC. Luego se cierra el ciclo del RBC cuando se incorpora el nuevo caso a la memoria de casos. Esta fase contiene tres algoritmos, los cuales se describen a continuación (Reyes González, 2014):

Algoritmo 5: Recuperación de los casos más semejantes utilizando los l-complejos calculados con los testores comunes para todas las clases.

Entrada: MA, l-complejos asociados a cada clase K_i (LC'), Nuevo Caso (O_t).

Salida: Clase seleccionada K_i.

Paso 1: *Para cada K_i de MA calcular:*

a) *$\beta_j(O_j, O_l)$, donde O_l son los casos que se corresponden con los l-complejos. Sólo se comparan los rasgos que conforman el l-complejo.*
b) *$\lambda_i(O_t) = \frac{\sum \beta_j(O_j, O_l)}{|l|}$, donde O_t es el nuevo caso, O_l los conceptos y $|l|$ el número de conceptos asociados a la clase i.*

Paso 2:

a) *Seleccionar el $Max\ (\lambda_i(O_t))$.*
b) *Seleccionar la clase K_i // los casos de la clase i se corresponden con los objetos más semejantes al nuevo caso.*

Con la salida que proporciona este algoritmo es posible realizar un nuevo tipo de adaptación nula: la conceptual. Además de recuperar los casos más semejantes al nuevo caso se ofrece al

[16] **p:** Parámetro que determina la cantidad de testores de mayor utilidad.

especialista los conceptos distintivos para ese grupo de casos recuperados. Esto facilita el proceso de toma de decisiones e incorpora un valor agregado a la tan utilizada adaptación nula, pues los conceptos constituyen elementos descriptivos a tener en cuenta para realizar la adaptación de la solución.

Algoritmo 6: Adaptación Nula Conceptual para la toma de decisiones.

Entrada: Objetos de la clase K_i seleccionada, l-complejos obtenidos para la claseK_i, Nuevo Caso (O_t).

Salida: O_t´´ // nuevo caso con la solución asignada aplicando la adaptación nula conceptual.

***Paso 1:** Asignar al caso O_t la solución del caso más parecido según los l-complejos que describen la clase y los casos más parecidos. // (Esta es la decisión que debe adoptar el especialista) El nuevo caso obtenido con la solución asignada es O_t´´.*

Algoritmo 7: Autoaprendizaje de la Base de Casos.

Entrada: O_t´´// nuevo caso modificado con la solución asignada.

Salida: clase K_i con el caso O_t´´asignado.

***Paso 1:** Asignar el nuevo caso O_t´´a la clase K_i.*

La herramienta permite obtener una estructura jerárquica de la BC que permite una búsqueda más eficiente basada en los conceptos, facilitando la implementación del ciclo de vida del Sistema Inteligente Basado en Casos, pues dado un nuevo caso se compara con los conceptos representativos de cada grupo facilitando así el proceso de recuperación y se decide a cual pertenece, de esta manera se toman los casos que conforman dicho grupo como los más semejantes al nuevo problema, para finalmente adaptar el caso y almacenarlo.

En la figura 2.1 se muestra la secuencia que sigue la solución propuesta, donde se evidencian las dos fases del modelo (Reyes González, 2014) y cada una de sus etapas:

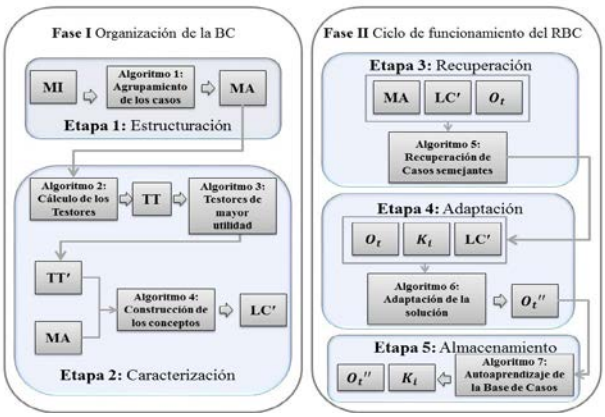


Figura 2.1 Propuesta de solución. Basado en (Reyes González, 2014).

2.2 Fase de exploración

En esta fase se precisan las historias de usuario, determinando su prioridad en correspondencia con la importancia que posee para el desarrollo del sistema. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, la tecnología y las prácticas a ser utilizadas durante el desarrollo de software. Consecutivamente se fundamentan los elementos y artefactos generados en esta fase.

2.2.1 Características no funcionales del sistema

Los requisitos no funcionales son características que se desean del sistema, señalan una restricción del mismo y son fundamentales en el éxito del producto, son propiedades o cualidades que el producto debe tener y normalmente se vinculan a requerimientos funcionales.

Para el Sistema Inteligente Basado en Casos se han determinado los siguientes requisitos no funcionales:

Interfaz de Usuario: Las vistas correspondientes al Sistema Inteligente Basado en Casos deben ser visualizadas en diversos dispositivos electrónicos (computadora, celular o tableta electrónica) sin que su apariencia se vea afectada.

Portabilidad: El Sistema Inteligente Basado en Casos debe tener la capacidad de funcionar en diversas plataformas.

Usabilidad: Se debe poseer un nivel bajo o medio de conocimientos de computación para el manejo del Sistema Inteligente Basado en Casos.

Hardware: Computadoras: debe contar con un microprocesador P4 o superior, CPU ≥ 1.5 GHz, memoria RAM ≥ 1 GB. Tabletas electrónicas: debe contar con un CPU ≥ 400MHz, memoria RAM ≥ 128 MB.

2.2.2 Características funcionales del sistema

En la metodología XP los requerimientos funcionales del sistema son modelados como HU[17], describen lo que se espera de cada funcionalidad y a su vez son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos.

Los títulos de las HU generadas son:

- **HU 1.** Cargar datos.
- **HU 2.** Gestionar datos.
- **HU 3.** Eliminar múltiples objetos.
- **HU 4.** Abrir una Base de Casos.

[17] **HU:** Historia de Usuario.

- **HU 5.** Gestionar los rasgos.
- **HU 6.** Eliminar múltiples rasgos.
- **HU 7.** Definir dominio y criterio de comparación por rasgo.
- **HU 8.** Definir umbral y criterio agrupacional.
- **HU 9.** Mostrar el agrupamiento de los objetos.
- **HU 10.** Calcular los Testores más relevantes.
- **HU 11.** Generar los conceptos de los grupos formados.
- **HU 12.** Organizar la base de casos en forma jerárquica y conceptual.
- **HU 13.** Recuperar los Casos Semejantes.
- **HU 14.** Realizar la adaptación de un caso.
- **HU 15.** Gestionar la solución adaptada.
- **HU 16.** Almacenar el nuevo caso en la base de casos.
- **HU 17.** Guardar una Base de Casos.

2.2.3 Especificación de las Historias de Usuario

La especificación de las HU se realizan con el objetivo de determinar cuales de estas son más vitales resolver y poder realizar una correcta planificación de su implementación, cada HU es clasificada según su prioridad para el negocio en:

Alta: Se le otorga a las HU que resultan fundamentales en el desarrollo del sistema. Estas son: cargar datos, definir dominio y criterio de comparación por rasgo, definir umbral y criterio agrupacional, mostrar el agrupamiento de los objetos, calcular los testores más relevantes, generar los conceptos de los grupos formados, organizar la base de casos en forma jerárquica y conceptual, recuperar los casos semejantes, realizar la adaptación de un caso, gestionar la solución adaptada y almacenar el nuevo caso en la base de casos.

Media: Se le otorga a las HU que resultan necesarias pero no imprescindibles para el funcionamiento del sistema. Estas son: gestionar datos, eliminar múltiples objetos, abrir una base de casos, gestionar los rasgos, eliminar múltiples rasgos y guardar una base de casos.

Baja: Se le otorga a las HU que son de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo. En las HU propuestas no se encuentra ninguna con esta clasificación.

De igual modo se tiene en cuenta la dificultad y posible existencia de errores durante la implementación de cada HU, por ello se clasifica a cada una según el riesgo de su desarrollo:

Alta: Cuando en la implementación de las HU se considera la posible existencia de errores que lleven a la inoperatividad del sistema. Las HU que se encuentran en esta clasificación son: cargar datos, definir dominio y criterio de comparación por rasgo, definir umbral y criterio agrupacional, mostrar el agrupamiento de los objetos, calcular los testores más relevantes,

generar los conceptos de los grupos formados, organizar la base de casos en forma jerárquica y conceptual, recuperar los casos semejantes, realizar la adaptación de un caso, gestionar la solución adaptada y almacenar el nuevo caso en la base de casos.

Media: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión. En el sistema se han clasificado las siguientes como riesgo medio: gestionar datos, eliminar múltiples objetos, abrir una base de casos, gestionar los rasgos, eliminar múltiples rasgos y guardar una base de casos.

Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto. En el sistema Inteligente Basado en Casos no hay ninguna HU clasificada como bajo riesgo.

Las HU son representadas mediante tablas con la estructura de la tabla 2.1, donde los puntos estimados y reales son representados en semanas:

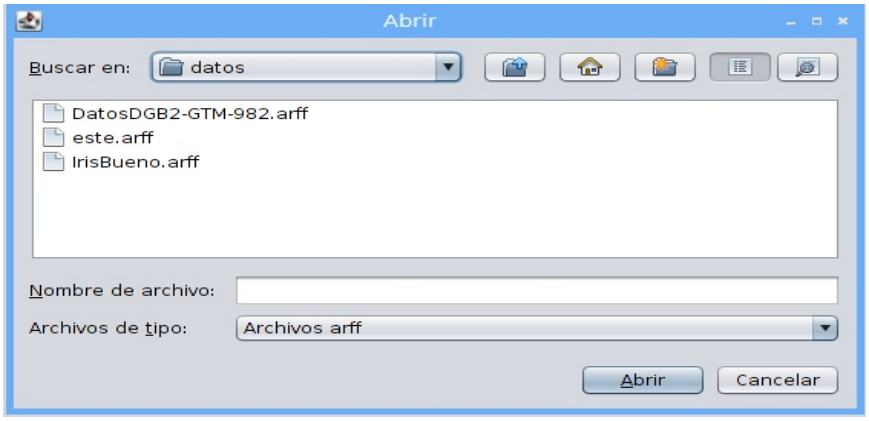
Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Cargar datos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede elegir cargar los datos en el sistema en formato ARFF. Una vez elegida la base de datos que este desea la aplicación debe visualizar dichos datos.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 2.1 HU 1. Cargar datos.

Para consultar el resto de las HU ver **Anexo I.**

2.3 Fase de planificación

Durante esta fase se realiza una estimación del esfuerzo que sostiene la implementación de cada HU, se crea el plan de iteraciones y se acuerda la posible fecha de entrega del producto.

2.3.1 Estimación del esfuerzo de cada Historia de Usuario

Es necesaria la medición del desempeño del proyecto, para ello se utiliza la estimación por puntos. Un punto es valorado como una semana de trabajo (5 días), estos fueron asignados a cada una de las HU teniendo en cuenta su prioridad para el negocio y el riesgo en el desarrollo.

Historias de Usuario	Puntos de Estimación
HU 1. Cargar datos.	4/5
HU 2. Gestionar datos.	2/5
HU 3. Eliminar múltiples objetos.	3/5
HU 4. Abrir una Base de Casos.	2/5
HU 5. Gestionar los rasgos.	1
HU 6. Eliminar múltiples rasgos.	2/5
HU 7. Definir dominio y criterio de comparación por rasgo.	1
HU 8. Definir umbral y criterio agrupacional.	1
HU 9. Mostrar el agrupamiento de los objetos.	1
HU 10. Calcular los Testores más relevantes.	2
HU 11 Generar los conceptos de los grupos formados.	3
HU 12. Organizar la base de casos en forma jerárquica y conceptual.	1
HU 13. Recuperar los Casos Semejantes.	1
HU 14. Realizar la adaptación de un caso.	1
HU 15. Gestionar la solución adaptada.	1/5
HU 16. Almacenar el nuevo caso en la base de casos.	3/5
HU 17. Guardar una Base de Casos.	1

Tabla 2.2 Estimación del esfuerzo por cada HU.

2.3.2 Plan de iteraciones

Una vez identificadas las HU y estimado el esfuerzo dedicado a la realización de cada una, se efectúa la planificación de la etapa de implementación. En esta las HU serán desarrolladas y probadas en un ciclo iterativo, representadas en TI[18]. En la tabla 2.3 se muestra el plan de iteraciones, donde se representa la duración de cada una de las tres iteraciones y las HU asociadas a cada una de estas.

[18] **TI:** Tareas de Ingeniería.

Iteración	HU a implementar	Duración de la iteración
Iteración 1	**HU 1.** Cargar datos.	4 semanas
	HU 2. Gestionar datos.	
	HU 3. Eliminar múltiples objetos.	
	HU 4. Abrir una Base de Casos.	
	HU 5. Gestionar los rasgos.	
	HU 6. Eliminar múltiples rasgos.	
	HU 7. Definir dominio y criterio de comparación por rasgo.	
Iteración 2	**HU 8.** Definir umbral y criterio agrupacional.	7 semanas
	HU 9. Mostrar el agrupamiento de los objetos.	
	HU 10. Calcular los Testores más relevantes.	
	HU 11. Generar los conceptos de los grupos formados.	
Iteración 3	**HU 12.** Organizar la base de casos en forma jerárquica y conceptual.	5 semanas
	HU 13. Recuperar los Casos Semejantes.	
	HU 14. Realizar la adaptación de un caso.	
	HU 15. Gestionar la solución adaptada.	
	HU 16. Almacenar el nuevo caso en la base de casos.	
	HU 17. Guardar una Base de Casos.	

Tabla 2.3 Plan de Iteraciones.

2.3.3 Plan de entregas

En el Plan de entregas se establecen aquellas HU que serán creadas para cada versión del producto, así como las fechas en las que se publicarán estas.

La tabla 2.4 muestra la planificación de las entregas establecidas para el Sistema Inteligente Basado en Casos.

Fecha	Historia de Usuario	Versión
Finalizada la iteración 1 10/03/2015	**HU 1.** Cargar datos.	v0.0.1
	HU 2. Gestionar datos.	
	HU 3. Eliminar múltiples objetos.	
	HU 4. Abrir una Base de Casos.	
	HU 5. Gestionar los rasgos.	
	HU 6. Eliminar múltiples rasgos.	
	HU 7. Definir dominio y criterio de comparación por rasgo.	

Finalizada la iteración 2 28/04/2015	**HU 8.** Definir umbral y criterio agrupacional.	
	HU 9. Mostrar el agrupamiento de los objetos.	v0.0.2
	HU 10. Calcular los Testores más relevantes.	
	HU 11. Generar los conceptos de los grupos formados.	
Finalizada la iteración 3 02/06/2015	**HU 12.** Organizar la base de casos en forma jerárquica y conceptual.	v1.0
	HU 13. Recuperar los Casos Semejantes.	
	HU 14. Realizar la adaptación de un caso.	
	HU 15. Gestionar la solución adaptada.	
	HU 16. Almacenar el nuevo caso en la base de casos.	
	HU 17. Guardar una Base de Casos.	

Tabla 2.4 Plan de entregas.

2.4 Fase de Diseño

En esta fase se realiza el diseño del Sistema Inteligente Basado en Casos. Con las facilidades que brinda la metodología XP la estrategia de diseño es simple, con el menor número de clases y métodos posibles, lo que propicia su fácil implementación en menos tiempo y esfuerzo.

2.4.1 Tarjetas Clase – Responsabilidad –Colaborador

Una Tarjeta CRC[19] es en realidad una colección de tarjetas índices estándar que representan clases. Las tarjetas se dividen en tres secciones, generalmente el nombre de la clase se coloca en el borde superior en forma de título, en la parte derecha los colaboradores (clases que se implican en cada funcionalidad) y las responsabilidades (funcionalidades) de la clase en el extremo izquierdo. (Pressman, 2011)

Objeto	
Descripción: Esta clase contiene la descripción de un objeto, compuesta por aquellos rasgos que lo integran.	
Responsabilidad: Adicionar un rasgo. Devolver y cambiar el identificador del objeto. Devolver el valor de un rasgo especificado. Devolver y cambiar el listado de rasgos que lo componen.	**Colaborador:** Rasgo.

Tabla 2.5 Tarjeta CRC 5. Objeto.

Para consultar el resto de las tarjetas CRC y el diagrama de clases ver **Anexo IV**.

[19] **CRC:** Clase-Responsabilidad-Colaborador.

2.4.2 Arquitectura de Software

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Es decir, la arquitectura brinda una visión global del sistema. Esto permite entenderlo, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar. Se puede ver que la noción clave de la arquitectura es la organización y está relacionada con aspectos de rendimiento, usabilidad, reutilización, limitaciones económicas y tecnológicas. (Almeira, y otros, 2007)

Para la implementación del presente sistema fue necesaria la aplicación de la arquitectura orientada a objetos, debido a las ventajas que esta presenta:

En la Arquitectura Orientada a Objetos los objetos están débilmente acoplados, debido a ello la implementación de estos puede modificarse sin afectar a los otros. Los objetos son a menudo representaciones de entidades del mundo real por lo que la estructura del sistema es fácilmente comprensible. Permite la reutilización del código, lo que facilita el trabajo posterior cuando se produzca un crecimiento en las fases futuras del desarrollo de software. (Sommerville, 2005)

La Arquitectura Orientada a Objetos facilita la incorporación de nuevos componentes y la reutilización de los existentes. Proporciona a los usuarios una colección de clases bases que pueden ser extendidas para el diseño de nuevos algoritmos, la implementación de nuevos problemas y operaciones. (Reynoso, y otros, 2004)

En la AOO[20] los componentes (objetos) de un sistema, encapsulan los datos y las operaciones que deben aplicarse para manipular los datos. La comunicación y coordinación entre los componentes se consigue mediante el paso de mensaje. Esta arquitectura es eficiente para muchos sistemas de software, pero está especialmente equipada para los sistemas de Desktop[21]. (Pressman , 2005)

Los componentes del estilo se basan en principios Orientado a Objeto: abstracción, encapsulamiento, herencia, desacoplamiento y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación. Los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación. En cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases. En tanto a los componentes, los objetos interactúan a través de invocaciones de funciones y procedimientos; siendo un objeto ante todo, una entidad reutilizable en el entorno de desarrollo (REYNOSO, 2006).

[20] **AOO:** Arquitectura Orientada a Objetos.
[21] **Desktop:** Se refiere al término en inglés para sistemas de escritorio.

2.4.3 Patrones de Diseño

Los patrones de diseño se han convertido en una técnica importante para el reúso del conocimiento de software. Cada patrón provee información sobre su diseño, describiendo las clases, métodos y relaciones que resuelven un problema de diseño en particular, estos han sido agrupados y organizados en catálogos cada uno dando diferentes clasificaciones y descripciones. (CAMACHO, y otros, 2004)

2.4.3.1 Patrones para asignar responsabilidades

Los patrones GRASP[22] describen los principios fundamentales de la asignación de responsabilidades a objetos en una aplicación. Se considera que más que patrones, son una serie de "buenas prácticas" recomendadas en el diseño de software. (RODRIGUEZ, 2014)

A continuación se describen los patrones GRASP presentes en al Sistema Inteligente Basado en Casos:

Experto: Su principio es asignar una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para realizar la responsabilidad. Una de las evidencias de este patrón se encuentra en la clase Objeto debido a que esta es la responsable de conocer los atributos y el total de los mismos y la clase FormatoTabla al conocer las posiciones de una tabla y colorearla.

```
public class Objeto {
    String Id;
    LinkedList<Rasgo> rasgos;

    public Objeto(String Id, LinkedList<Rasgo> rasgos) {...31 lines }
    public Objeto(String Id){...5 lines }
    public void addRasgo(Rasgo rasgo){...19 lines }
    public void print() {...7 lines }
    public void println() {...8 lines }
    public String getId() {...3 lines }
    public void setId(String Id) {...3 lines }
    public LinkedList<Rasgo> getRasgos() {...3 lines }
    public void setRasgos(LinkedList<Rasgo> rasgos) {...6 lines }
```

Creador: debería ser el responsable de la creación de una nueva instancia de alguna clase, o sea la clase B es un creador de los objetos de la clase A. A continuación se brinda una muestra de este patrón en la aplicación. La clase Rasgo es la encargada de crear instancias de las clases FuncionComparacion e Importancia.

```
public abstract class Rasgo {
    private String nombre;
    private String valor;
    private FuncionComparacionRasgo funcionComparacion;
    private ImportanciaInformacional importancia;
    boolean activo;

    public Rasgo(String nombre) {
        this.nombre = nombre;
        this.valor = "";
        this.funcionComparacion = new Funcion1_ConparacionRasgo();
        this.importancia = new ImportanciaInformacionalCalculada(0.5, 0.5);
        this.activo = true;
    }
```

[22] **GRASP:** del inglés General Responsibility Assignment Software Patterns/Patrones Generales de Software para la Asignación de Responsabilidades.

Bajo Acoplamiento: Se encarga de soportar bajas dependencias, bajo impacto del cambio e incrementar la reutilización, o sea asigna una responsabilidad de manera que el acoplamiento[23] permanezca bajo. Una de las evidencias de esta patrón se encuentra en las clases Main, CriterioAgrupacional y Grupo; esto se debe a que la clase Main necesita la clase Grupo para gestionar el agrupamiento; pero la experta en crear los grupos es la clase CriterioAgrupacional, por lo cual la clase Main lleva a cabo la creación de CriterioAgrupacional (manteniendo el acoplamiento global más bajo) y CriterioAgrupacional crea los objetos de la clase Grupo. Reduciendo así el impacto del cambio, al mantener las clases de forma independiente.

```
public class Main extends javax.swing.JFrame {
    private LinkedList<Rasgo> rasgosList;
    private SQLConextion conextion;
    private LinkedList<Objeto> objetosList;
    private String ultimaDir;

    private FuncionComparacionObjeto FCO;
    private Umbral umbral;
    private CriterioAgrupacional criterioAgrupac
```

```
public abstract class CriterioAgrupacional {
    protected double[][] matrizSemejanza;
    protected double valorUmbral;
    protected LinkedList<Objeto> objetList;

    public CriterioAgrupacional(double[][] matrizSemejanza,
            double valorUmbral, LinkedList<Objeto> objetList)
    public abstract LinkedList<Grupo> Agrupar();
```

Alta cohesión: Mantiene la complejidad manejable, o sea asigna una responsabilidad de manera que la cohesión[24] permanezca alta. La clase Main presenta la funcionalidad de ejecutar el agrupamiento y colabora con las clases FuncionComparacionObjeto (para comparar dos objetos respecto a sus rasgos), Umbral (para definir en que medida se considera a dos objetos similares o diferentes) y CriterioAgrupacional para crear los grupos y representarlos.

Controlador: Es el responsable de gestionar un evento de entrada al sistema, por tanto asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase. Las clases Main y Agrupamiento implementan el patrón Controlador ya que son ellas las encargadas de gestionar los eventos que realice el usuario sobre la aplicación.

```
private void jComboBox3ItemStateChanged(java.awt.event.ItemEvent evt) {...4 lines }
private void jComboBox3ActionPerformed(java.awt.event.ActionEvent evt) {...23 lines }
private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }
private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {...6 lines }
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {...5 lines }
private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {...4 lines }
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {...5 lines }
```

[23] **Acoplamiento:** Es una medida de la fuerza con que un elemento está conectado, tiene conocimiento o confía en otros elementos. Un elemento con bajo acoplamiento no depende de muchas clases.
[24] **Cohesión:** Es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes. (Una clase debe de hacer lo que respecta a su entidad, y no hacer acciones que involucren a otra clase ó entidad).

2.5 Fase de Implementación

Para la implementación de las HU se chequea a cada una en conjunto con el plan de iteraciones, de esta forma inicia la creación de las TI. Estas son para el uso estricto de los programadores, pueden ser escritas en lenguaje técnico. Teniendo en cuenta las iteraciones determinadas para la implementación, se exponen en la tabla 2.6, las tareas correspondientes a cada HU.

Iteración	HU	Tareas de Ingeniería
Iteración 1	**HU 1.** Cargar datos.	**TI 1.** Cargar datos en formato ARFF.
	HU 2. Gestionar datos.	**TI 2.** Eliminar un objeto.
	HU 3. Eliminar múltiples objetos.	**TI 3.**Eliminar múltiples objetos.
	HU 4. Abrir una Base de Casos.	**TI 4.** Abrir una Base de Casos.
	HU 5. Gestionar los rasgos.	**TI 5.** Insertar un rasgo.
		TI 6. Eliminar un rasgo.
	HU 6. Eliminar múltiples rasgos.	**TI 7.** Eliminar múltiples rasgos.
	HU 7. Definir dominio y criterio de comparación por rasgo.	**TI 8.** Establecer dominio de un rasgo.
		TI 9. Establecer la función de comparación por rasgo.
Iteración 2	**HU 8.** Definir umbral y criterio agrupacional.	**TI 10.** Establecer un umbral de semejanza.
		TI 11. Establecer un criterio agrupacional.
	HU 9. Mostrar el agrupamiento de los objetos.	**TI 12.** Mostrar el agrupamiento.
	HU 10. Calcular los Testores más relevantes.	**TI 13.** Cálculo de la matriz de diferencia.
		TI 14. Cálculo de la matriz básica.
		TI 15. Selección de los testores.
		TI 16. Calcular la relevancia de un rasgo.
		TI 17. Selección de los testores más relevantes.
	HU 11. Generar los conceptos de los grupos formados.	**TI 18.**Generación de los l-complejos de cada grupo.
		TI 19. Generación de la estrella de cada grupo.
Iteración 3	**HU 12.** Organizar la base de casos en forma jerárquica y conceptual.	**TI 20.** Organizar la estructura de la base de casos.
		TI 21. Mostrar la estructura de la base de casos.
	HU 13. Recuperar los Casos Semejantes.	**TI 22.** Entrar un nuevo caso al sistema.
		TI 23. Modificar el nuevo caso.
		TI 24. Buscar casos semejantes en la base se casos.
	HU 14. Realizar la adaptación de un caso.	**TI 25.** Realizar la adaptación del especialista.
	HU 15. Gestionar la solución adaptada.	**TI 26.** Mostrar la solución adaptada.

	HU 16. Almacenar el nuevo caso en la base de casos.	**TI 27.** Almacenar el caso resuelto.
	HU 17. Guardar una Base de Casos.	**TI 28.** Guardar la base de casos.

Tabla 2.6 Tareas de Ingeniería por HU.

Las TI implementadas y definidas en cada iteración serán representadas mediante una tabla, a continuación se presenta la TI correspondiente a la HU Cargar datos:

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Cargar datos en formato ARFF	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 4/5
Fecha Inicio: 06 / 02 / 2015	**Fecha Fin:** 11 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: En la pestaña Datos se elige la opción Cargar datos ARFF, se busca la localización del archivo en la computadora, se escoge el fichero arff deseado y se pulsa el botón aceptar. Una vez que el sistema logre cargar el fichero este formato se visualizan los datos.	

Tabla 2.7 TI 1. Cargar datos en formato ARFF.

Para consultar el resto de las Tareas de Ingeniería ver **Anexo III**.

En el desarrollo del sistema se utiliza la programación multihilo para la implementación del proceso de cálculo de los l-complejos.

2.6 Conclusiones Parciales del Capítulo

Con el desarrollo de las fases de: exploración, planificación, diseño e implementación, propuestas por la metodología XP:

- ✓ Se determinaron las características funcionales del sistema representadas en historias de usuario y se planificaron las diferentes iteraciones permitiendo organizar el desarrollo del sistema.
- ✓ Se eligió la arquitectura de software orientada a objetos por sus características determinantes en la implementación del sistema inteligente y el uso de patrones de diseño que contribuyen a la implementación de buenas prácticas en el desarrollo del sistema.
- ✓ Se implementaron las tareas de ingeniería en correspondencia con los requerimientos funcionales descritos, lo que permitió obtener una primera versión funcional del producto.

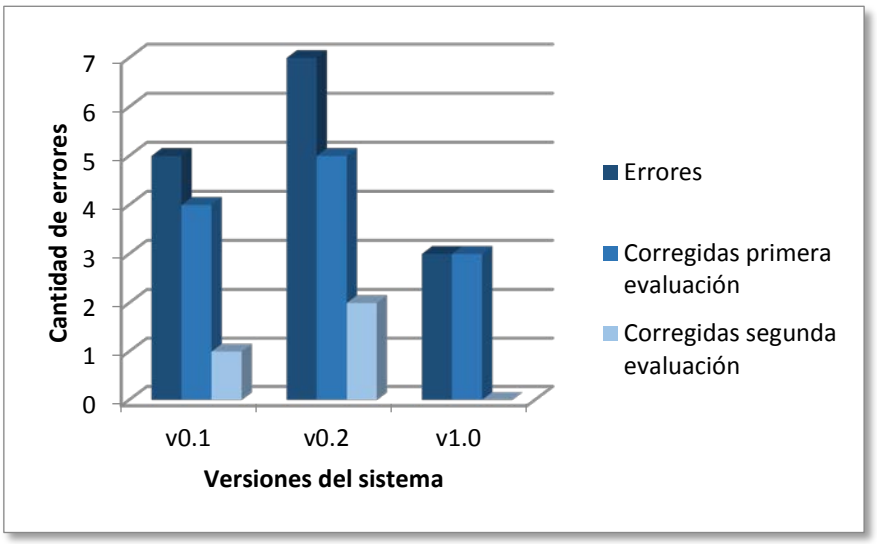
CAPÍTULO 3: Pruebas y experimentación

En este capítulo se realizan las pruebas del sistema para comprobar el correcto funcionamiento del software. Se especifican las pruebas unitarias pertenecientes a la última fase propuesta por la metodología XP así como las pruebas de aceptación, además se exponen los resultados obtenidos con respecto a la base de dato utilizada en la experimentación.

3.1 Pruebas Unitarias

Las pruebas unitarias están dirigidas a probar clases de manera aislada y se relacionan con el código, es una forma de comprobar el correcto funcionamiento de un módulo de código, de esta manera se puede asegurar que cada uno funcione correctamente por separado.

Para automatizar las pruebas unitarias se utilizó una librería que permite realizar la ejecución de clases Java de manera controlada, esta evalúa si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera: Junit en su versión 4.10. Una vez detectados los errores se realizaron dos fases de evaluaciones, una primera donde se procede a la corrección de los errores detectados y una segunda fase para erradicar los errores que no pudieron ser eliminados en la primera y para comprobar todas las corregidas. La Figura 3.1 muestra un resumen del comportamiento de errores encontrados en las dos fases de validación durante cada una de las versiones del Sistema Inteligente Basado en Casos:



Gráfica 3.1 Resultado de las pruebas unitarias. (Elaboración propia)

A continuación se muestran evidencias de algunas de las pruebas realizadas utilizando la librería Junit 4.10 en a las funcionalidades del sistema.

Para consultar el resto de las funcionalidades que fueron probadas con la librería Junit ver **Anexo VI**.

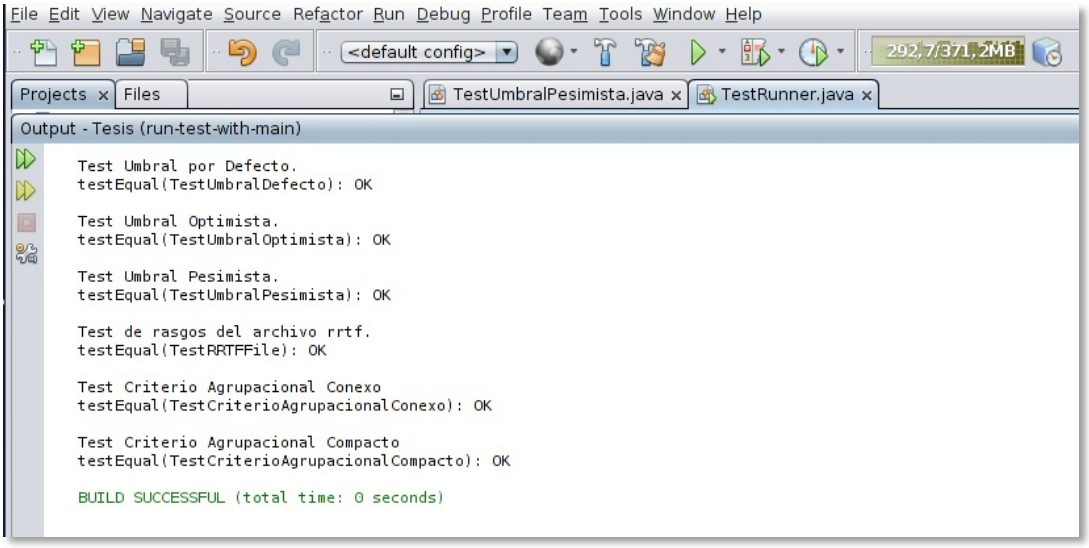


Figura 3.1 Resultados Satisfactorios 5.

3.2 Pruebas de Aceptación

Las pruebas de aceptación son aquellas con carácter previo al paso de producción de una nueva versión del producto. Con ellas se confirma si las funcionalidades pactadas para las entregas recogidas en las HU cumplen con las expectativas del cliente.

Las pruebas de aceptación realizadas al Sistema Inteligente Basado en Casos son especificadas en un modelo cuya muestra se presenta a continuación en la tabla 3.1:

Prueba de aceptación	
Código: HU8_P1	**Historia de Usuario:** Definir umbral y criterio agrupacional.
Nombre: Establecer el umbral de semejanza con valores incorrectos.	
Descripción: Tratar de establecer el umbral de semejanza con literales o valores mayores que 1 y menores que 0.	
Condiciones de Ejecución: El usuario debe cargar previamente los datos.	
Entradas / Pasos de Ejecución: Seleccionar el umbral de usuario y entrar un literal cualquiera o un valor fuera del rango comprendido de 0 a 1. Dar clic en el botón Aceptar.	
Resultado esperado: el sistema no establece el umbral, se mantiene en la misma vista y muestra el mensaje: “Solo se admiten números en el rango [0,1)”	
Evaluación de la prueba: Satisfactoria	
Resultado obtenido: el sistema no establece el umbral, se mantiene en la misma vista y muestra el mensaje: “Solo se admiten números en el rango [0,1)”	

Tabla 3.1 Prueba de aceptación HU8_P1.

Para consultar el resto de las pruebas de aceptación ver **Anexo V**.

3.3 Experimentación

Para la experimentación del sistema se selecciona la base de datos Zoo tomada del repositorio UCI (**http://archive.ics.uci.edu/ml/**). A continuación se muestra la descripción de esta:

Zoo: La base de datos Zoo consta de 101 registros de animales con 17 atributos, de estos solo 1 es numérico y el resto son no numéricos. Esta base de datos contiene información que describe las diferentes características de los animales y permite clasificarlos según su tipo.

3.3.1 Resultados de la experimentación

Para realizar la experimentación del sistema, se utiliza la base de datos descrita previamente como un caso de estudio, se establecieron los criterios de comparación por cada rasgo, los valores de umbral y criterio agrupacional, de esta manera se pudo realizar el ciclo del razonamiento basado en casos con una organización jerárquica y conceptual de la base de casos.

3.3.2 Caso de Estudio

Se toma para este caso de estudio la base de datos Zoo que consta de 101 registros de animales, descritos en 17 rasgos, el nombre del animal constituye el primero de estos, el cual es ignorado para realizar el agrupamiento al igual que el rasgo objetivo que se encuentra al final.

Primeramente se carga la base de datos en el sistema, luego se establece el criterio de comparación por cada rasgo.

El criterio de comparación por rasgos utilizado en este caso de estudio fue la igualdad estricta:

$$C_s\big(X_s(O_i), X_s(O_j)\big) = \begin{cases} 1 \;\; si\; X_s(O_i) = X_s(O_j) \vee \; X_s(O_i) = * \; \vee X_s(O_j) = * \\ \\ 0 \; en\; otro\; caso \end{cases}$$

Se establecen entonces el umbral de usuario 0.85 y el criterio agrupacional β_0-Conexo obteniéndose un total de 7 grupos.

Luego la aplicación determina el testor más relevante: {hair, eggs, airborne, predator, legs, tail}

A partir de este se calculan los conceptos por cada clase. De esta manera queda organizada la base de casos en forma jerárquica y conceptual.

A continuación se muestran los conceptos de cada grupo:

Grupo 0 (41 casos)

[Hair={ true}] [eggs={ false}] [airborne={ false, true} [predator={ true, false}][legs={ 0}]

[tail={ false, true}]

Grupo 1 (13 casos)

[Hair={ false}] [eggs={ true}] [airborne={ false} [predator={ true, false}][legs={ 4, 2, 0}] [tail={ true}]

Grupo 2 (21 casos)

[Hair={ false}] [eggs={ true}] [airborne={ true} [predator={ true, false}][legs={ 2, 4}] [tail={ true}]

Grupo 3 (17 casos)

[Hair={ true, false}] [eggs={ true}] [airborne={ true} [predator={ true, false}][legs={ 0, 4, 5, 6, 8}] [tail={ false}]

Grupo 4 (7 casos)

[Hair={ false}] [eggs={ true}] [airborne={ false} [predator={ true, false}][legs={ 0, 4}] [tail={ false}]

Grupo 5 (1 caso)

[Hair={ false}] [eggs={ false}] [airborne={ false} [predator={ true}][legs={ 8}] [tail={ true}]

Grupo 6 (1 caso)

[Hair={ false}] [eggs={ false}] [airborne={ false} [predator={ true}][legs={ 0}] [tail={ true}]

Se presenta un nuevo caso al sistema con los valores siguientes:

Hair	false
Feathers	false
Eggs	false
Milk	false
Airborne	false
Aquatic	false
predator	true
Toothed	false

Backbone	false
Breathes	true
Venomous	true
Fins	false
Legs	8.0
Tail	true
Domestic	false
Catsize	false

Luego de esto el sistema brinda el listado de los casos semejantes al caso nuevo. El especialista determina la adaptación de la solución del caso presentado a partir del listado de casos semejantes mostrado.

De esta manera se toma como solución del nuevo caso el rasgo objetivo del caso 073, clasificando el tipo de animal como ***invertebrate***.

Finalmente el caso resuelto se almacena en la base de casos como miembro del ***grupo 5***, cuyos conceptos son:

[hair={ false}] [eggs={ false}] [airborne={ false} [predator={ true}][legs={ 8}] [tail={ true}]

3.4 Conclusiones Parciales del Capítulo

Luego de haber realizado las pruebas unitarias y las experimentaciones a la propuesta de solución se llega a las siguientes conclusiones:

- ✓ Las pruebas unitarias realizadas haciendo uso de la librería Junit ofrecieron una muestra detallada de los errores encontrados en las funcionalidades facilitando así la detección de los mismos en la implementación.
- ✓ Los resultados de la experimentación con la base de datos demuestran que la utilización del algoritmo LC-Conceptual permite lograr una estructura jerárquica de la base de casos, donde en el primer nivel se sitúan los conceptos que describen a los grupos, favoreciendo el acceso y la recuperación de los casos.

Conclusiones Generales

Con la investigación realizada se logró como producto final una herramienta de software que apoya la toma de decisiones integrando el RBC y el algoritmo LC-Conceptual cuando se presentan datos mezclados e incompletos. De esta manera se cumple con el objetivo general trazado, permitiendo arribar a las siguientes conclusiones:

- ✓ Se desarrolló una herramienta computacional que permite el procesamiento de datos mezclados e incompletos así como la utilización de diferentes criterios de comparación por rasgos en dependencia de su naturaleza.
- ✓ El sistema permite la selección de rasgos relevantes y el cálculo de su importancia haciendo uso de la teoría de testores.
- ✓ A través de la herramienta se logra organizar la base de casos en una estructura jerárquica utilizando los conceptos generados a partir del algoritmo LC-Conceptual.
- ✓ Se implementó el ciclo completo del Razonador Basado en Casos integrado con las ideas básicas del algoritmo LC-Conceptual.

Recomendaciones

Se proponen como futuros trabajos en esta área de investigación, incorporar a la herramienta las siguientes funcionalidades:

- ✓ Otras funciones de semejanza entre objetos que permita realizar comparaciones con los resultados que se obtienen en las distintas bases de datos.
- ✓ Implementar los testores típicos por clase para la generación de los conceptos.
- ✓ Implementar otros criterios de agrupamientos reportados en la literatura.
- ✓ Incorporar un módulo para la comparación de los resultados obtenidos en relación con otros modelos de organización de la memoria de casos.

Referencias

Agnar, Aamodt y Plaza, Enric . 1994. *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.* s.l. : Sensagent, 1994.

D'Aquila, Raimundo O. 2010. *Acerca de la INTELIGENCIA...de los SISTEMAS INTELIGENTES.* 2010.

Eom, Ahn Heum. 1998. 1998.

Kolodner. 1992. *An Introduction to Case-Based Reasoning.* 1992. págs. 3-34.

Krishnamoorthy, C S y Rajeev, S. 1996. *Artificial Intelligence and Expert Systems for Engineers.* 1996. pág. 320. ISBN: 0849391253.

López de Mántaras, R. 1997. *Case- Based Reasoning: An Overwiew.* 1997.

Martínez Ladrón de Guevara, Jorge. 2011. *Fundamentos de programación en java.* 2011.

Pereira, José Manuel, Crespo Domínguez, Miguel Ángel y Sáez Ocejo, José Luís. *PROPUESTA DE CLASIFICACIÓN DE LOS MODELOS DE PREDICCIÓN DEL FRACASO EMPRESARIAL.* s.l. : Universidad de Vigo. pág. 19.

Quintero, Juan Bernardo. 2007. *Arquitectura de Software.Patrones en la Arquitectura.* 2007.

Rivadeneira Molina, Silvia Gabriela. 2012. *METODOLOGÍAS ÁGILES ENFOCADAS AL MODELADO DE REQUERIMIENTOS.* Santa Cruz(Argentina) : s.n., 2012.

Sánchez, Emilio A, Letelier, Patricio y Canós, José H. 2006. *Mejorando la gestiˊn de historias de usuario en eXtreme Programming.* Valencia : s.n., 2006.

Santos, José Á, Carrasco, Ariel y Martínez, José F. 2004. *Feature Selection using Typical Testors applied to Estimation of Stellar Parameters.* 2004. ISSN.

Sumathi, S y Sivanandam, S N. 2006. *Introduction to Data Mining and its Applications.* [ed.] Janusz Kacprzyk. 2006. ISBN 3-540-34350-4.

Witten, Ian H y Frank, Eibe . 2005. *Data Mining. Practical Machine Learning Tools and Techniques.* s.l. : MORGAN KAUFMANN PUBLISHERS , 2005. ISBN: 0-12-088407-0.

2013. ¿Qué es NetBeans? . *Sitio Oficial de NetBeans.* [En línea] 2013. www.netbeans.org.

Alba, Eduardo y Santana, Roberto . 2010. *Generación de matrices para evaluar el desempeño de estrategias de búsqueda de testores típicos.* 2010.

—. 2010. *Generación de matrices para evaluar el desempeño de estrategias de búsqueda de testores típicos.* 2010.

Almeira, Adriana Sandra y Perez Cavenago, Vanina . 2007. *Tesina de Arquitectura de Software: Estilos y Patrones.* Argentina : s.n., 2007.

Barlow, Horace , Kaushal y Mitchison. 1989. *Finding minimum entropy codes. .* 1989.

Beck, K. 1999. *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.

Béjar, J. 1992. *LINNEO+: Herramienta para la adquisición de conocimiento y generación de reglas de clasificación en dominios poco estructurados.* 1992.

Bello, R. 2002. *Aplicaciones de la Inteligencia Artificial.* Jalisco (México) : s.n., 2002. ISBN: 970-27-0177-5.

Berzal, Fernando . *CLUSTERING.* pág. 125.

Boeras Velázquez, Mairelys , y otros. 2012. Aplicando el método de Boehm y Turner. *Serie Científica de la Universidad de las Ciencias Informáticas.* [En línea] 15 de junio de 2012. http://publicaciones.uci.cu | http:// seriecientifica@uci.cu. ISSN | RNPS.

Bregón , Anibal , y otros. 2005. Un sistema de razonamiento basado en casos para la clasificación de fallos en sistemas dinámicos. 2005.

Calabria , Luis y Píriz , Pablo . 2003. *Metodología XP.* Uruguay : Universidad ORT Uruguay, 2003.

Calderón, Amaro , Valverde , Sarah Dámaris y Rebaza, Jorge Carlos . 2007. *Metodologías Ágiles.* Perú : s.n., 2007.

CAMACHO, ERIKA , CARDESO, FABIO y NUÑEZ, GABRIEL . 2004. *ARQUITECTURAS DE SOFTWARE.* 2004.

Canós, José H, Letelier , Patricio y Penad, Mª Carmen. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n.

Carrasco Ochoa, Jesús Ariel . 2009. *Reconocimiento de Patrones.* 2009.

Company Headquarters. 2013. Sitio Oficial de Visual Paradigm. *Visual Paradigm.* [En línea] 13 de Enero de 2013. http://www.visual-paradigm.com.

Del Coz Velasco, Juan José y Díez Peláez, Jorge. 2008. *SISTEMAS INTELIGENTES.Introducción al Aprendizaje Automático.* Oviedo : s.n., 2008.

Elaborado y diseñado en formato PDF por Oficina General del Sistema de Bibliotecas y Biblioteca Central UNMSM. Teoría de Sistemas Expertos.

Funes , Ana . Diciembre de 2008. *Agrupamiento Conceptual Jerárquico Basado en Distancias.* Valencia : s.n., Diciembre de 2008.

Gálvez Cordero, Jaime , y otros. 2004. *Un Sistema Inteligente para el Aprendizaje de Fundamentos de Programación Orientada a Objetos.* Málaga. : s.n., 2004.

Gálvez Lio, Daniel . 2008. Razonamiento Basado en Casos (TEMA 8). 2008.

Garlan , David y Shaw, Mary . 1994. *An Introduction to Software Architecture.* New Jersey : s.n., 1994.

Gennari, J. 1990. *Models of Incremental Concept Formation.* 1990.

GONZÁLEZ, Ezequiel . Diciembre, 2013. *APRENDIZAJE Y PLANIFICACIÓN EN SISTEMAS INTELIGENTES AUTÓNOMOS.* BUENOS AIRES : s.n., Diciembre, 2013.

Guerra Gandón, Alejandro , Vega Pons, Sandro y Ruiz Shulcloper, José . 2012. *Reconocimiento de Patrones.Algoritmos de agrupamiento conceptuales: un estado del arte.* La Habana : s.n., 2012. ISSN.

Gutiérrez, M I. 2003. *Un Modelo para la Toma de Decisiones usando Razonamiento Basado en Casos en condiciones de Incertidumbre. Trabajo de Tesis en opción al grado científico de Doctor en Ciencias Técnicas. UCLV.* 2003.

Kolodner, Janet L y Jona, Menachem Y. 2000. *Case-based reasoning - An overview.* 2000.

Letelier, Patricio y Penadés, Mª Carmen . 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *CyTA.* [En línea] 15 de enero de 2006. http://www.cyta.com.ar. ISSN 1666-1680.

Mark, Bill . 1989. *Proceedings of the Case-Based Reasoning Workshop.* s.l. : Sensagent, 1989.

Martínez Trinidad, J. 1998. Fuzzy LC-Conceptual Algorithm. 1998.

Méndiz Noguera, Inés . 2007. *UN ESTUDIO SOBRE LA APLICACION DEL RAZONAMIENTO BASADO EN CASOS A LA CONSTRUCCION DE PROGRAMAS.* Madrid : s.n., 2007.

Mitra, Rudradeb y Basak, Jayanta . 2005. *Methods of Case Adaptation: A Survey.* Delhi : s.n., 2005.

Monroy Vázquez, Juan Olegario . 2011. *Minería de datos con conjuntos aproximados para clasificación de imágenes satelitales.* s.l. : Universidad de Manizales, 2011. págs. 129-158. ISSN: 0123-9678.

Morales , Eduardo . 2009. [En línea] 25 de 08 de 2009. http://ccc.inaoep.mx.

Ochoa, Alberto, y otros. 2006. Más allá del Razonamiento Basado en Casos y una Aproximación al Modelado de Sociedades Utilizando Minería de Datos. Zacatecas : Editorial ADC, 2006.

Pascual y Sánchez. 2007. *Algoritmos de agrupamiento.* 2007.

Peña Ayala, Alejandro . 2006. *Sistemas basados en Conocimiento: Una Base para su Concepción y Desarrollo.* México : s.n., 2006. ISBN: 970-94797-4-1.

Pérez Suárez, Airel y Medina Pagola, José E. 2014. *Algoritmos para el agrupamiento conceptual de objetos.* s.l. : Centro de Aplicaciones de Tecnolog´as de Avanzada (CENATAV), 2014. ISSN 2072 6260.

Pomerol, J. 1995. *Artificial intelligence and human decision making.* s.l. : European Journal of Operation Research 99, 1995. págs. 3-25.

Pons Porrata, Aurora . 2004. *TESIS DOCTORAL: DESARROLLO DE ALGORITMOS PARA LA ESTRUCTURACIÓN DINÁMICA DE INFORMACIÓN Y SU APLICACIÓN A LA DETECCIÓN DE SUCESOS.* 2004.

Pons Porrata, Ruiz Shulcloper y Martínez Trinidad. 2002. *RGC:a new conceptual clustering algorithm for mixed incomplete data sets.* 2002.

Pressman , Roger S. 2005. *Agile Development.* 2005.

Pressman , Roger S. 2002. *INGENIERÍA DE SOFTWARE. Un enfoque práctico.(quinta edición).* [ed.] Concepción Fernández Madrid. 5. Madrid : s.n., 2002. ISBN 0-079677-0.

Pressman, Roger S. 2011. *Software Engineering, a practitioner's approach. (7th edición).* 2011. ISBN 9780071267823.

Pressman, Roger S. 2006. *Ingeniería de Software. Un enfoque práctico.(sexta edición).* 6. México : Editorial McGrawHill, 2006. pág. 980.

—. 2010. *Software Engineer: A Practitiorer´s Approach (SEVENTH ADITION).* 7. s.l. : McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 2010. ISBN 978-0-07-337 597-7.

Razonamiento Basado en Casos(CBR). **Díaz Gómez , Fernando . 2013.** Universidad de Valladolid : s.n., 2013. Conferencia E.U de informática-Segovia.

Reyes González, Yunia . 2014. *Modelo para la adaptación de las soluciones en un Sistema Basado en Casos utilizando el agrupamiento conceptual.* La Habana : s.n., 2014.

Reyes González, Yunia y Martínez Sánchez, Natalia . 2014. *Algoritmos Conceptuales: Una perspectiva para el modelado.* 2014.

REYNOSO. 2006. Arquitectura de Software. [En línea] 2006. www.microsft.com/spanish/msdn/arquitectura/arquitectura_de_software.

Reynoso, Carlos y Kiccillof, Nicolás . 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* BUENOS AIRES : s.n., 2004.

RODRIGUEZ, J. 2014. PATRONES DE DISEÑO Y FRAMEWORKS. *ingenieriasw2.* [En línea] 1 de abril de 2014. http://ingenieriasw2.blogspot.com.

Ruiz Shulcloper. 2013. *RECONOCIMIENTO LÓGICO COMBINATORIO DE PATRONES: TEORÍA Y APLICACIONES. TESIS EN OPCIÓN AL GRADO CIENTÍFICO DE DOCTOR EN CIENCIAS.* 2013.

Ruiz Shulcloper, José . 2000. *Logical Combinatorial Pattern Recognition.* Barcelona : s.n., 2000. págs. 122-138.

Ruíz Shulcloper, José , Alba Cabrera, Eduardo y Lazo Cortés, Manuel . 1995. *INTRODUCCION AL RECONOCIMIENTO DE PATRONES.* 1995.

Ruíz Shulcloper, José, Guzmán Arenas, Adolfo y Martínez Trinidad, Francisco. 1999. *Enfoque Lógico Combinatorio al Reconocimiento de Patrones.* 1999. ISBN.

Schmidhuber, Jürgen . 1992. *Learning factorial codes by predictability minimization.* 1992.

Sommerville, Ian. 2005. *Ingeniería de Software.* [ed.] Miguel Martín Romo. 7. Madrid : Person Education Limited, 2005. pág. 712. ISBN 84-7829-074-5.

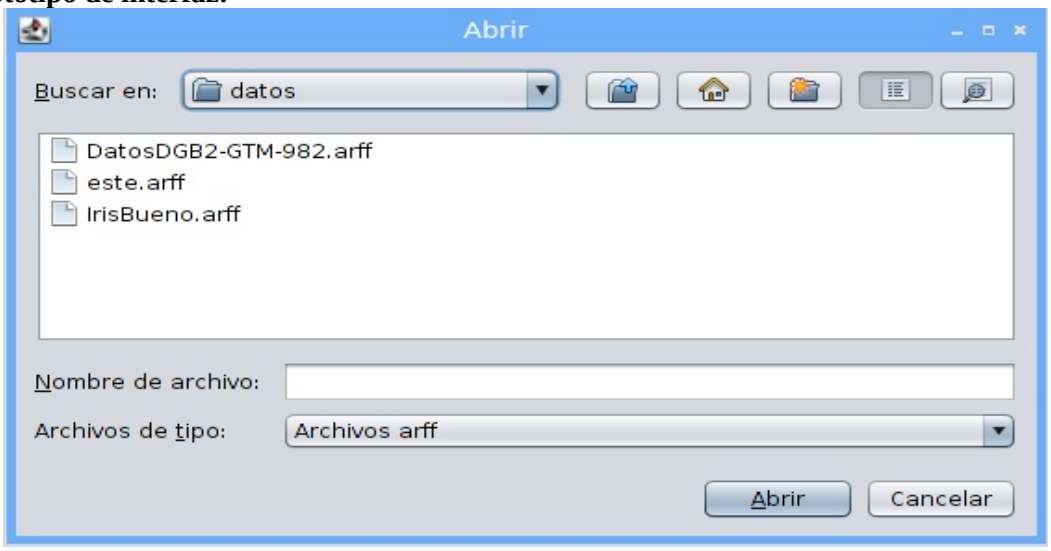
—. 2007. *Software Engineering.* 8. China : Person Edication Limited, 2007. ISBN 10: 0-321-31379-8.

Vazquez, Luis . *Aprendizaje no supervisado y análisis de agrupamientos.* págs. 1-21.

XU, RUI y WUNSCH, DONALD C. 2009. *CLUSTERING.* [ed.] David B Fogel. s.l. : IEEE Press Editorial Board, 2009. ISBN: 978-0-470-27680-8.

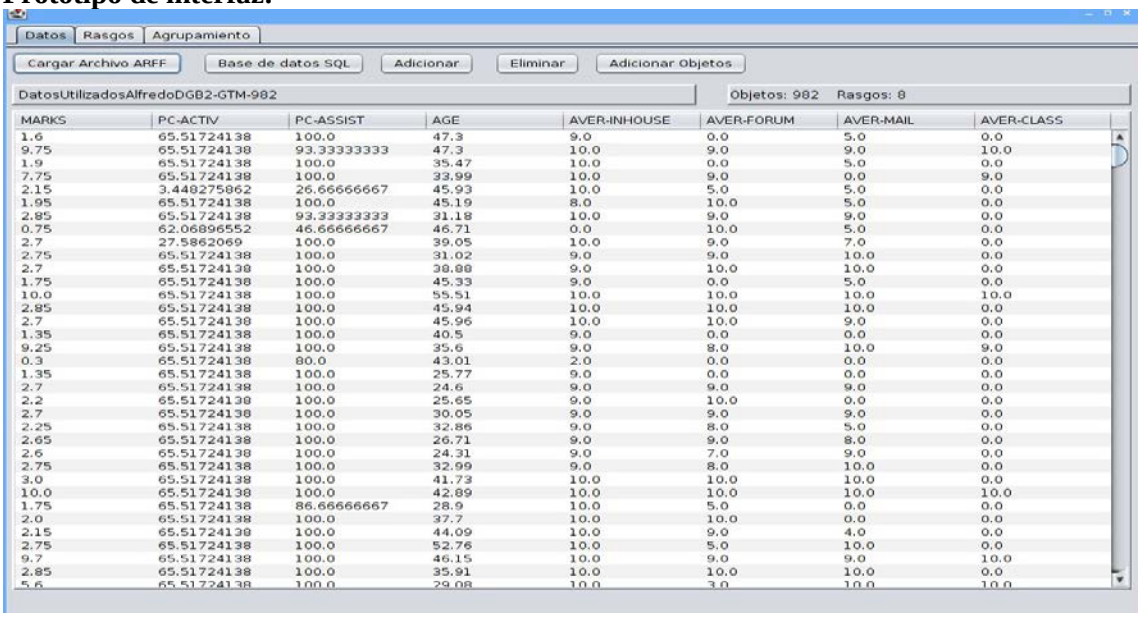
Anexo I

HU 1. Cargar datos.

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Cargar datos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede elegir cargar los datos en el sistema en formato ARFF. Una vez elegida la base de datos que este desea la aplicación debe visualizar dichos datos.	
Observaciones:	
Prototipo de interfaz: 	

HU 2. Gestionar datos.

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Gestionar datos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede visualizar y/o eliminar objetos del conjunto de datos del sistema.	

Observaciones: El usuario debe cargar previamente los datos.
Prototipo de interfaz: 

HU 3. Eliminar múltiples objetos.

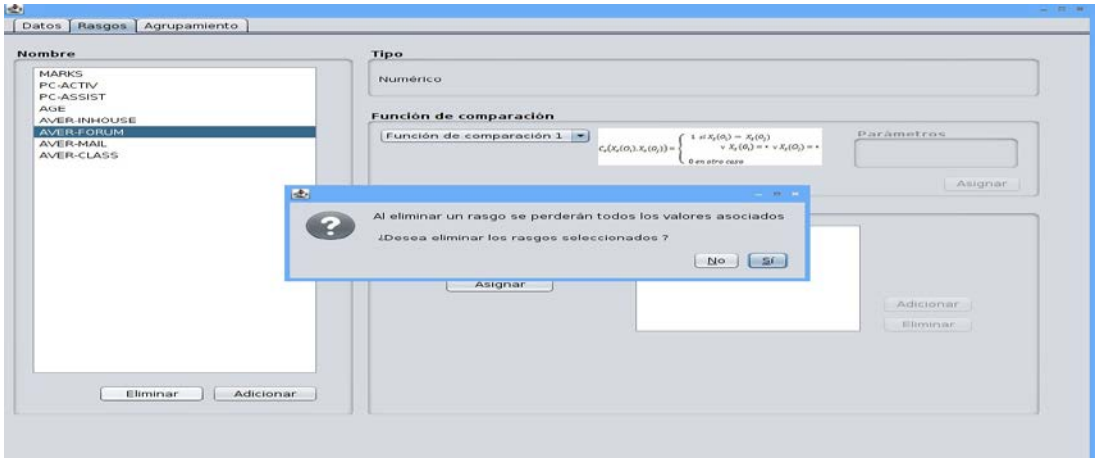
Historia de Usuario	
Número: 3	**Nombre de Historia de Usuario:** Eliminar múltiples objetos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 3/5
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2/5
Descripción: El usuario (Especialista) puede seleccionar un grupo de objeto del conjunto de datos y proceder a su eliminación.	
Observaciones: El usuario debe cargar previamente los datos.	
Prototipo de interfaz:	

HU 4. Abrir una Base de Casos.

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Abrir una Base de Casos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 1/5
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1/5

Descripción: El usuario (Especialista) puede abrir una Base de Casos en el sistema. Una vez elegida debe visualizar dichos datos.
Observaciones: El usuario debe cargar previamente los datos.
Prototipo de interfaz:

HU 5. Gestionar los rasgos.

Historia de Usuario	
Número: 5	**Nombre de Historia de Usuario:** Gestionar los rasgos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede visualizar, insertar y/o eliminar los rasgos de un objeto contenido en el conjunto de datos que se encuentra en el sistema.	
Observaciones: El usuario debe cargar previamente los datos.	
Prototipo de interfaz: 	

HU 6 Eliminar múltiples rasgos.

Historia de Usuario	
Número: 6	**Nombre de Historia de Usuario:** Eliminar múltiples rasgos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 2/5
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1/5
Descripción: El usuario (Especialista) puede seleccionar de un objeto del conjunto de datos un	

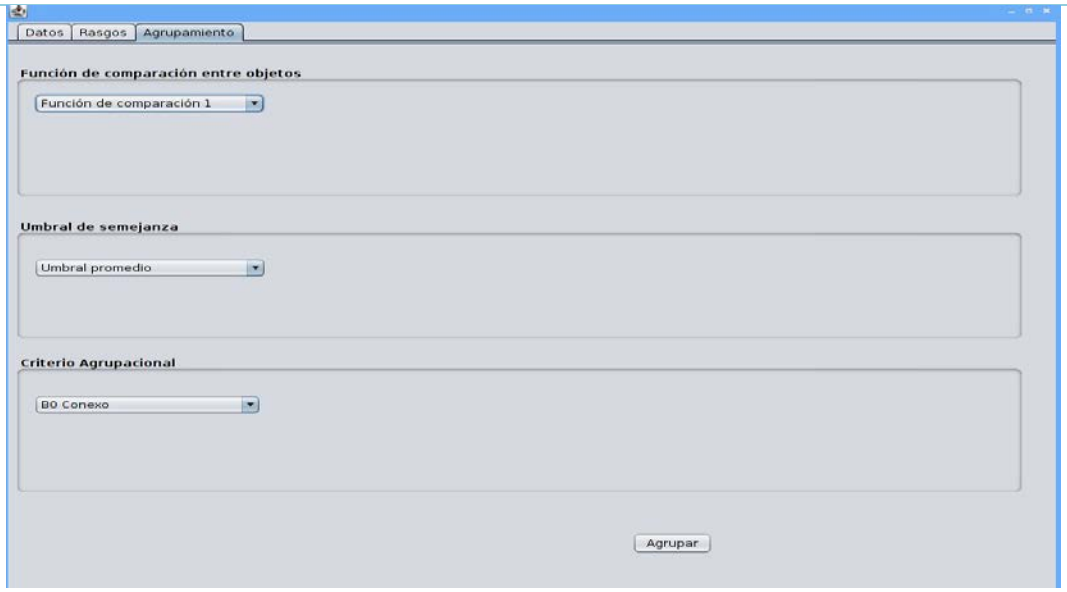
grupo de rasgos y proceder a su eliminación.
Observaciones: El usuario debe cargar previamente los datos.
Prototipo de interfaz:

HU 7. Definir dominio y criterio de comparación por rasgo.

Historia de Usuario	
Número: 7	**Nombre de Historia de Usuario:** Definir dominio y criterio de comparación por rasgo
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede establecer de cada rasgo de los objetos contenidos en el conjunto de datos el dominio, función de comparación y en caso de ser necesario entrar un parámetro de un rasgo.	
Observaciones: El usuario debe cargar previamente los datos.	
Prototipo de interfaz:	

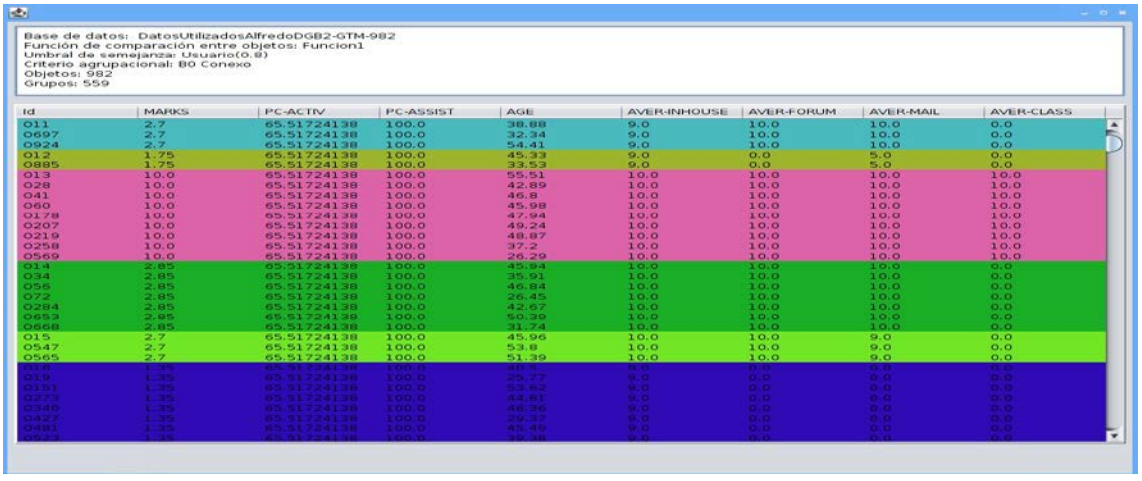
HU 8. Definir umbral y criterio agrupacional.

Historia de Usuario	
Número: 8	**Nombre de Historia de Usuario:** Definir umbral y criterio agrupacional,
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede seleccionar la función de comparación entre objetos, el umbral de semejanza y el criterio agrupacional que desea para realizar el agrupamiento de los objetos.	
Observaciones: El usuario debe cargar previamente los datos y establecer las peculiaridades de los rasgos.	
Prototipo de interfaz:	



HU 9. Mostrar el agrupamiento de los objetos.

Historia de Usuario	
Número: 9	**Nombre de Historia de Usuario:** Mostrar el agrupamiento de los objetos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede visualizar los grupos de objetos que se forman así como la información del agrupamiento realizado.	
Observaciones: El usuario debe cargar previamente los datos, establecer las peculiaridades de los rasgos y erigir los aspectos del agrupamiento.	
Prototipo de interfaz:	



HU 10. Calcular los Testores más relevantes.

Historia de Usuario	
Número: 10	**Nombre de Historia de Usuario:** Calcular los Testores más relevantes.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 2
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2
Descripción: Debe permitir disminuir el número de variables del problema, de esta forma facilita el trabajo y se obtiene un resultado más eficiente.	
Observaciones: El usuario debe cargar previamente los datos, establecer las peculiaridades de los rasgos y erigir los aspectos del agrupamiento.	
Prototipo de interfaz:	

HU 11. Generar los conceptos de los grupos formados.

Historia de Usuario	
Número: 11	**Nombre de Historia de Usuario:** Generar los conceptos de los grupos formados.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 3
Riesgo en Desarrollo: Alto	**Puntos Reales:** 3
Descripción: Debe permitir extraer aquellos conceptos (características) distintivos de cada grupo formado en el agrupamiento de los objetos.	
Observaciones: El usuario debe cargar previamente los datos, establecer las peculiaridades de los rasgos, erigir los aspectos del agrupamiento.	
Prototipo de interfaz:	

HU 12. Organizar la base de casos en forma jerárquica y conceptual.

Historia de Usuario	
Número: 12	**Nombre de Historia de Usuario:** Organizar la base de casos en forma jerárquica y conceptual.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Puntos estimados:** 1

Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Debe permitir organizar los grupos con sus conceptos en forma jerárquica para formar la base de casos del razonador basado en casos.	
Observaciones: El usuario debe cargar previamente los datos, establecer las peculiaridades de los rasgos, erigir los aspectos del agrupamiento.	
Prototipo de interfaz:	

HU 13. Recuperar los Casos Semejantes.

Historia de Usuario	
Número: 13	**Nombre de Historia de Usuario:** Recuperar los casos semejantes.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede presentar un nuevo caso para encontrar su solución, el sistema debe hacer una búsqueda en la base de casos de aquellos casos que son similares al nuevo y finalmente visualizar los casos semejantes.	
Observaciones: La base de casos debe estar organizada en forma jerárquica y conceptual.	
Prototipo de interfaz:	

HU 14. Realizar la adaptación de un caso.

Historia de Usuario	
Número: 14	**Nombre de Historia de Usuario:** Realizar la adaptación de los casos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede elegir el caso que sea más semejante de todos para adaptarlo a la nueva situación. Finalmente realiza la adaptación y visualiza el resultado de la misma.	
Observaciones: La base de casos debe estar organizada en forma jerárquica y conceptual.	
Prototipo de interfaz:	

HU 15. Gestionar la solución adaptada.

Historia de Usuario	
Número: 15	**Nombre de Historia de Usuario:** Gestionar la solución adaptada.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede visualizar, modificar u eliminar elementos de la adaptación realizada recientemente que aún no ha sido almacenada.	
Observaciones: La base de casos debe estar organizada en forma jerárquica y conceptual, además se debe haber realizado una adaptación recientemente que no halla sido almacenada en la base.	
Prototipo de interfaz:	

HU 16. Almacenar el nuevo caso en la base de casos.

Historia de Usuario	
Número: 16	**Nombre de Historia de Usuario:** Almacenar el nuevo caso en la base de casos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario (Especialista) puede guardar en la base de casos la adaptación.	
Observaciones: La base de casos debe estar organizada en forma jerárquica y conceptual, además se debe haber realizado una adaptación recientemente que no halla sido almacenada.	
Prototipo de interfaz:	

HU 17. Guardar una Base de Casos.

Historia de Usuario	
Número: 17	**Nombre de Historia de Usuario:** Guardar una Base de Casos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Especialista	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1

Descripción: El usuario (Especialista) puede guardar la Base de Casos organizada en forma jerárquica y conceptual en un fichero.
Observaciones: El usuario debe cargar previamente los datos.
Prototipo de interfaz:

Anexo II

Tabla 2.2 Estimación del esfuerzo por cada HU.

Historias de Usuario	Puntos de Estimación
HU 1. Cargar datos.	4/5
HU 2. Gestionar datos.	2/5
HU 3. Eliminar múltiples objetos.	3/5
HU 4. Abrir una Base de Casos.	2/5
HU 5. Gestionar los rasgos.	1
HU 6. Eliminar múltiples rasgos.	2/5
HU 7. Definir dominio y criterio de comparación por rasgo.	1
HU 8. Definir umbral y criterio agrupacional.	1
HU 9. Mostrar el agrupamiento de los objetos.	1
HU 10. Calcular los Testores más relevantes.	2
HU 11 Generar los conceptos de los grupos formados.	3
HU 12. Organizar la base de casos en forma jerárquica y conceptual.	1
HU 13. Recuperar los Casos Semejantes.	1
HU 14. Realizar la adaptación de un caso.	1
HU 15. Gestionar la solución adaptada.	1/5
HU 16. Almacenar el nuevo caso en la base de casos.	3/5
HU 17. Guardar una Base de Casos.	1

Tabla 2.3 Plan de Iteraciones.

Iteración	HU a implementar	Duración de la iteración
Iteración 1	**HU 1.** Cargar datos.	4 semanas
	HU 2. Gestionar datos.	
	HU 3. Eliminar múltiples objetos.	
	HU 4. Abrir una Base de Casos.	
	HU 5. Gestionar los rasgos.	
	HU 6. Eliminar múltiples rasgos.	

	HU 7. Definir dominio y criterio de comparación por rasgo.	
Iteración 2	**HU 8.** Definir umbral y criterio agrupacional.	7 semanas
	HU 9. Mostrar el agrupamiento de los objetos.	
	HU 10. Calcular los Testores más relevantes.	
	HU 11. Generar los conceptos de los grupos formados.	
Iteración 3	**HU 12.** Organizar la base de casos en forma jerárquica y conceptual.	5 semanas
	HU 13. Recuperar los Casos Semejantes.	
	HU 14. Realizar la adaptación de un caso.	
	HU 15. Gestionar la solución adaptada.	
	HU 16. Almacenar el nuevo caso en la base de casos.	
	HU 17. Guardar una Base de Casos.	

Tabla 2.4 Plan de Entregas.

Fecha	Historia de Usuario	Versión
Finalizada la iteración 1 10/03/2015	**HU 1.** Cargar datos.	v0.0.1
	HU 2. Gestionar datos.	
	HU 3. Eliminar múltiples objetos.	
	HU 4. Abrir una Base de Casos.	
	HU 5. Gestionar los rasgos.	
	HU 6. Eliminar múltiples rasgos.	
	HU 7. Definir dominio y criterio de comparación por rasgo.	
Finalizada la iteración 2 28/04/2015	**HU 8.** Definir umbral y criterio agrupacional.	v0.0.2
	HU 9. Mostrar el agrupamiento de los objetos.	
	HU 10. Calcular los Testores más relevantes.	
	HU 11. Generar los conceptos de los grupos formados.	
Finalizada la iteración 3 02/06/2015	**HU 12.** Organizar la base de casos en forma jerárquica y conceptual.	v1.0
	HU 13. Recuperar los Casos Semejantes.	
	HU 14. Realizar la adaptación de un caso.	
	HU 15. Gestionar la solución adaptada.	
	HU 16. Almacenar el nuevo caso en la base de casos.	
	HU 17. Guardar una Base de Casos.	

Tabla 2.6 Tareas de Ingeniería.

Iteración	HU	Tareas de Ingeniería
Iteración 1	**HU 1.** Cargar datos.	**TI 1.** Cargar datos en formato ARFF.

	HU 2. Gestionar datos.	**TI 2.** Eliminar un objeto.
	HU 3. Eliminar múltiples objetos.	**TI 3.**Eliminar múltiples objetos.
	HU 4. Abrir una Base de Casos.	**TI 4.** Abrir una Base de Casos.
	HU 5. Gestionar los rasgos.	**TI 5.** Insertar un rasgo.
		TI 6. Eliminar un rasgo.
	HU 6. Eliminar múltiples rasgos.	**TI 7.** Eliminar múltiples rasgos.
	HU 7. Definir dominio y criterio de comparación por rasgo.	**TI 8.** Establecer dominio de un rasgo.
		TI 9. Establecer la función de comparación por rasgo.
Iteración 2	**HU 8.** Definir umbral y criterio agrupacional.	**TI 10.** Establecer un umbral de semejanza.
		TI 11. Establecer un criterio agrupacional.
	HU 9. Mostrar el agrupamiento de los objetos.	**TI 12.** Mostrar el agrupamiento.
	HU 10. Calcular los Testores más relevantes.	**TI 13.** Cálculo de la matriz de diferencia.
		TI 14. Cálculo de la matriz básica.
		TI 15. Selección de los testores.
		TI 16. Calcular la relevancia de un rasgo.
		TI 17. Selección de los testores más relevantes.
	HU 11. Generar los conceptos de los grupos formados.	**TI 18.**Generación de los l-complejos de cada grupo.
		TI 19. Generación de la estrella de cada grupo.
Iteración 3	**HU 12.** Organizar la base de casos en forma jerárquica y conceptual.	**TI 20.** Organizar la estructura de la base de casos.
		TI 21. Mostrar la estructura de la base de casos.
	HU 13. Recuperar los Casos Semejantes.	**TI 22.** Entrar un nuevo caso al sistema.
		TI 23. Modificar el nuevo caso.
		TI 24. Buscar casos semejantes en la base se casos.
	HU 14. Realizar la adaptación de un caso.	**TI 25.** Realizar la adaptación del especialista.
	HU 15. Gestionar la solución adaptada.	**TI 26.** Mostrar la solución adaptada.
	HU 16. Almacenar el nuevo caso en la base de casos.	**TI 27.** Almacenar el caso resuelto.
	HU 17. Guardar una Base de Casos.	**TI 28.** Guardar la base de casos.

Anexo III

TI 1. Cargar datos en formato ARFF.

Tarea de Ingeniería

Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Cargar datos en formato ARFF	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 4/5
Fecha Inicio: 06 / 02 / 2015	**Fecha Fin:** 11 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: En la pestaña Datos se elige la opción Cargar datos ARFF, se busca la localización del archivo en la computadora, se escoge el fichero arff deseado y se pulsa el botón aceptar. Una vez que el sistema logre cargar el fichero este formato se visualizan los datos.	

TI 2. Eliminar un objeto.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Eliminar un objeto.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2/5
Fecha Inicio: 12 / 02 / 2015	**Fecha Fin:** 13 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se destaca el objeto que se desea eliminar, se pulsa el botón eliminar y se confirma la eliminación, posteriormente se pueden contemplar los datos actualizados con la ausencia de este objeto.	

TI 3.Eliminar múltiples objetos.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 3
Nombre Tarea: Eliminar múltiples objetos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3/5
Fecha Inicio: 16 / 02 / 2015	**Fecha Fin:** 18 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se seleccionan todos los objetos que se desea eliminar, se pulsa el botón eliminar y se confirma la eliminación múltiple, posteriormente se pueden contemplar los datos actualizados con la ausencia de estos objetos.	

TI 4. Abrir una Base de Casos.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 4
Nombre Tarea: Abrir una Base de Casos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2/5
Fecha Inicio: 19 / 02 / 2015	**Fecha Fin:** 20 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: En la pestaña Datos se elige la opción Abrir una Base de Casos, se busca la localización del archivo en la computadora, se escoge el fichero deseado y se pulsa el botón aceptar. Una vez que el sistema logre cargar el fichero se visualizan los datos en la pestaña de agrupamiento.	

TI 5. Insertar un rasgo.

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 5
Nombre Tarea: Insertar un rasgo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 23 / 02 / 2015	**Fecha Fin:** 25 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se pulsa el botón adicionar, se establece el tipo de rasgo y el nombre, luego se pulsa el botón aceptar y se visualiza el nuevo rasgo en el conjunto de rasgos.	

TI 6. Eliminar un rasgo.

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 5
Nombre Tarea: Eliminar un rasgo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 25 / 02 / 2015	**Fecha Fin:** 27 / 02 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se destaca el rasgo que se desea eliminar, se pulsa el botón eliminar y se confirma la eliminación, posteriormente se pueden contemplar los rasgos actualizados sin la presencia del eliminado.	

TI 7. Eliminar múltiples rasgos.

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 6
Nombre Tarea: Eliminar múltiples rasgos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2/5
Fecha Inicio: 02 / 03 / 2015	**Fecha Fin:** 03 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se seleccionan todos los rasgos que se desea eliminar, se pulsa el botón eliminar y se confirma la eliminación múltiple, posteriormente se pueden contemplar los rasgos actualizados con la ausencia de aquellos eliminados.	

TI 8. Establecer dominio de un rasgo.

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 7
Nombre Tarea: Establecer dominio de un rasgo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 04 / 03 / 2015	**Fecha Fin:** 06 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se inserta el máximo y mínimo valor que puede tomar el rasgo en caso de ser numérico, en otro caso se inserta los valores que este puede tomar, se pulsa el botón aceptar y se actualizan los campos.	

TI 9. Establecer la función de comparación por rasgo.

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 7
Nombre Tarea: Establecer la función de comparación por rasgo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 06 / 03 / 2015	**Fecha Fin:** 10 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se elige la función de comparación por rasgo que desea aplicar al rasgo seleccionado y se visualiza una imagen de esta función, si la función de comparación	

seleccionada contiene parámetros de entrada entonces se introducen dichos parámetros.

TI 10. Establecer un umbral de semejanza.

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** 8
Nombre Tarea: Establecer un umbral de semejanza	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 11 / 03 / 2015	**Fecha Fin:** 13 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se elige el umbral de semejanza que desea aplicar al agrupamiento o en otro caso puede entrar uno de preferencia, luego se visualiza el nombre del seleccionado.	

TI 11. Establecer un criterio agrupacional.

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** 8
Nombre Tarea: Establecer un criterio agrupacional	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 13 / 03 / 2015	**Fecha Fin:** 17 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se elige el criterio agrupacional que desea aplicar al agrupamiento, luego de visualizar el nombre del criterio seleccionado.	

TI 12. Mostrar el agrupamiento.

Tarea de Ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** 9
Nombre Tarea: Mostrar el agrupamiento	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 18 / 03 / 2015	**Fecha Fin:** 24 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Luego de haber establecido un umbral y un criterio agrupacional, se pulsa el botón Agrupar que se encuentra en la pestaña de Agrupamiento, entonces se muestra toda la	

información del agrupamiento: nombre de la base de datos, función de comparación entre objetos, umbral de semejanza, criterio agrupacional, cantidad de objetos y cantidad de grupos formados, visualizando los grupos en una escala de colores.

TI 13. Cálculo de la matriz de diferencia.

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** 10
Nombre Tarea: Cálculo de la matriz de diferencia	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 25 / 03 / 2015	**Fecha Fin:** 26 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: A partir de los grupos formados anteriormente se realiza una comparación entre los objetos de diferentes grupos a través de un criterio de comparación booleano con el fin de obtener la matriz de diferencia.	

TI 14. Cálculo de la matriz básica.

Tarea de Ingeniería	
Número Tarea: 14	**Número Historia de Usuario:** 10
Nombre Tarea: Cálculo de la matriz básica	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 27 / 03 / 2015	**Fecha Fin:** 30 / 03 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: A partir de la matriz de diferencia calculada anteriormente se realiza una comparación entre las filas de la misma con la finalidad de obtener aquellas filas que no son superfilas de nadie, las filas encontradas constituyen las filas básicas que forman la matriz básica.	

TI 15. Selección de los testores.

Tarea de Ingeniería	
Número Tarea: 15	**Número Historia de Usuario:** 10
Nombre Tarea: Selección de los testores	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 31 / 03 / 2015	**Fecha Fin:** 01/ 04 / 2015
Programador Responsable:	

Erisel Almarales Vázquez Liyanis Peláez Baños
Descripción: A partir de la matriz básica calculada anteriormente se realiza una selección por fila de aquellos rasgos con valor uno. De esta manera, por cada fila básica se conforman los testores.

TI 16. Calcular la relevancia de un rasgo.

Tarea de Ingeniería	
Número Tarea: 16	**Número Historia de Usuario:** 10
Nombre Tarea: Calcular la relevancia de un rasgo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 02 / 04 / 2015	**Fecha Fin:** 03 / 04 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Una vez conformados los testores se debe calcular la frecuencia de aparición y la longitud de un rasgo i, luego se calcula la relevancia o peso de este rasgo a partir de los valores de frecuencia y longitud encontrados.	

TI 17. Selección de los testores más relevantes.

Tarea de Ingeniería	
Número Tarea: 17	**Número Historia de Usuario:** 10
Nombre Tarea: Selección de los testores más relevantes	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 06 / 04 / 2015	**Fecha Fin:** 07 / 04 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se debe calcular la relevancia de un testor i a partir del peso de los rasgos que lo componen, en dependencia de la importancia de este testor, si es el de mayor valor será seleccionado como el más relevantes.	

TI 18.Generación de los l-complejos de cada grupo.

Tarea de Ingeniería	
Número Tarea: 18	**Número Historia de Usuario:** 11
Nombre Tarea: Generación de los l-complejos de cada grupo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Fecha Inicio: 08 / 04 / 2015	**Fecha Fin:** 17 / 04 / 2015

Programador Responsable:
Erisel Almarales Vázquez Liyanis Peláez Baños
Descripción: A partir de la matriz de aprendizaje y el testor más relevante se aplica el operador de refunión condicionada (RUC) a cada clase de la matriz, como resultado de este se obtienen los l-complejos (conceptos) de cada grupo.

TI 19.Generación de la estrella de cada grupo.

Tarea de Ingeniería	
Número Tarea: 19	**Número Historia de Usuario:** 11
Nombre Tarea: Generación de la estrella de cada grupo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Fecha Inicio: 17 / 04 / 2015	**Fecha Fin:** 28 / 04 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: A partir de los l-complejos (conceptos) de un grupo se forma la estrella de este, cada estrella G_τ se forma con el l-complejo que le corresponde al grupo.	

TI 20. Organizar la estructura de la base de casos.

Tarea de Ingeniería	
Número Tarea: 20	**Número Historia de Usuario:** 12
Nombre Tarea: Organizar la estructura de la base de casos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 29 / 04 / 2015	**Fecha Fin:** 04 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se debe organizar cada clase de la matriz de aprendizaje con sus l-complejos correspondientes en forma jerárquica, donde el l-complejo constituya el nodo inicial.	

TI 21. Mostrar la estructura de la base de casos.

Tarea de Ingeniería	
Número Tarea: 21	**Número Historia de Usuario:** 12
Nombre Tarea: Mostrar la estructura de la base de casos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 04 / 05 / 2015	**Fecha Fin:** 06 / 05 / 2015

Programador Responsable: Erisel Almarales Vázquez, Liyanis Peláez Baños
Descripción: Una vez organizada cada clase de la matriz de aprendizaje con sus l-complejos correspondientes en forma jerárquica, se debe mostrar esta información en la interfaz de usuario.

TI 22. Entrar un nuevo caso al sistema.

Tarea de Ingeniería	
Número Tarea: 22	**Número Historia de Usuario:** 13
Nombre Tarea: Entrar un nuevo caso al sistema	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 07 / 05 / 2015	**Fecha Fin:** 08 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez, Liyanis Peláez Baños	
Descripción: Se debe dar la posibilidad al usuario de insertar la descripción de un nuevo caso en el sistema, de esta manera el sistema pide los datos correspondientes, los inserta y finalmente muestra el nuevo caso con su descripción.	

TI 23. Modificar el nuevo caso.

Tarea de Ingeniería	
Número Tarea: 23	**Número Historia de Usuario:** 13
Nombre Tarea: Modificar el nuevo caso	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 08 / 05 / 2015	**Fecha Fin:** 11 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez, Liyanis Peláez Baños	
Descripción: Se debe dar la posibilidad al usuario de modificar la descripción del nuevo caso, luego de realizar las modificaciones deseadas muestra los cambios del nuevo caso con su descripción.	

TI 24. Buscar casos semejantes en la base se casos.

Tarea de Ingeniería	
Número Tarea: 24	**Número Historia de Usuario:** 13
Nombre Tarea: Buscar casos semejantes en la base se casos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1

Fecha Inicio: 11 / 05 / 2015	**Fecha Fin:** 13 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se debe brindar la posibilidad al usuario de buscar casos semejantes al nuevo, el sistema debe hacer una búsqueda en la base de casos de aquellos casos que son similares al nuevo y finalmente visualizar los casos semejantes.	

TI 25. Realizar la adaptación del especialista.

Tarea de Ingeniería	
Número Tarea: 25	**Número Historia de Usuario:** 14
Nombre Tarea: Realizar la adaptación del especialista	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 14 / 05 / 2015	**Fecha Fin:** 20 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se debe presentar al usuario el listado de los casos semejantes, si este determina que los casos presentados no cumplen con la expectativa del nuevo caso, entonces puede entrar el resultado manualmente, finalmente se visualiza el caso resuelto.	

TI 26. Mostrar la solución adaptada.

Tarea de Ingeniería	
Número Tarea: 26	**Número Historia de Usuario:** 15
Nombre Tarea: Mostrar la solución adaptada	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/5
Fecha Inicio: 21 / 05 / 2015	**Fecha Fin:** 22 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se debe presentar al usuario el caso resuelto, luego de realizar las modificaciones.	

TI 27. Almacenar el caso resuelto.

Tarea de Ingeniería	
Número Tarea: 27	**Número Historia de Usuario:** 16
Nombre Tarea: Almacenar el caso resuelto	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3/5

Fecha Inicio: 22 / 05 / 2015	**Fecha Fin:** 26 / 05 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: Se debe brindar la posibilidad al usuario de almacenar el nuevo caso resuelto en la base de casos, finalmente luego de seleccionar el almacenamiento se visualiza el caso resuelto en la base de casos.	

TI 28. Guardar la base de casos.

Tarea de Ingeniería	
Número Tarea: 28	**Número Historia de Usuario:** 17
Nombre Tarea: Guardar la base de casos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 27 / 05 / 2015	**Fecha Fin:** 02 / 06 / 2015
Programador Responsable: Erisel Almarales Vázquez Liyanis Peláez Baños	
Descripción: En la pestaña de agrupamiento se pulsa el botón Guardar Base de Casos, se selecciona el destino deseado para salvar los datos y se elige el botón guardar. El fichero se guarda exitosamente en la dirección deseada.	

Anexo IV

Tarjeta CRC 1. Rasgo.

Rasgo	
Descripción: Contiene la información sobre los rasgos.	
Responsabilidad: Devolver y cambiar el nombre del rasgo Devolver y cambiar el valor del rasgo. Devolver y cambiar la función de comparación de un rasgo.	**Colaborador:** FunciónComparacionRasgo RasgoLiteral RasgoNumerico RasgoNominal RasgoOrdinal

Tarjeta CRC 2. FuncionComparacionRasgo.

FuncionComparacionRasgo
Descripción: Esta clase se encarga de dar una medida de la similitud o diferencia entre dos

valores de un rasgo en dependencia de la función de comparación seleccionada.	
Responsabilidad: Devolver la semejanza entre los valores de un rasgo. Devolver la diferencia entre los valores de un rasgo.	**Colaborador:** FunciónComparacionRasgo1 FunciónComparacionRasgo2 FunciónComparacionRasgo3 FunciónComparacionRasgo4 FunciónComparacionRasgo5 FunciónComparacionRasgo6

Tarjeta CRC 3. Umbral.

Umbral	
Descripción: Esta clase devuelve el valor del umbral en dependencia del tipo de umbral seleccionado.	
Responsabilidad: Devolver el valor del umbral.	**Colaborador:** UmbralPesimista UmbralOptimista UmbralPromedio UmbralUsuario

Tarjeta CRC 4. CriterioAgrupacional.

CriterioAgrupacional	
Descripción: Esta clase es la encargada de realizar el agrupamiento en dependencia del tipo de criterio agrupacional seleccionado.	
Responsabilidad: Devolver los grupos formados.	**Colaborador:** CriterioAgrupacionalConexo CriterioAgrupacionalCompacto

Tarjeta CRC 5. Objeto

Objeto	
Descripción: Esta clase contiene la descripción de un objeto, compuesta por aquellos rasgos que lo integran.	
Responsabilidad: Adicionar un rasgo. Devolver y cambiar el identificador del objeto. Devolver el valor de un rasgo especificado.	**Colaborador:** Rasgo.

Devolver y cambiar el listado de rasgos que lo componen.	

Tarjeta CRC 6. Grupo.

Grupo	
Descripción: Esta clase contiene la información del grupo de objetos que lo integran.	
Responsabilidad: Adicionar un objeto. Devolver y cambiar el color del grupo. Devolver localización de un objeto especificado. Devolver y cambiar el listado de objetos que lo componen. Comprobar la existencia de un objeto según su identificador.	**Colaborador:** Objeto.

Tarjeta CRC 7. Función_Comparacion_Objeto.

Función_Comparacion_Objeto	
Descripción: Esta clase se encarga de dar una medida de la similitud entre dos objetos.	
Responsabilidad: Devolver el valor de similitud entre dos objetos dados.	**Colaborador:** FuncionComparacionObjeto1

Tarjeta CRC 8. Programa.

Programa	
Descripción: Esta clase se encarga de controlar toda la información del sistema.	
Responsabilidad: Controlar el sistema.	**Colaborador:** Funcion_Comparacion_Objeto Grupo CriterioAgrupacional Umbral

Diagrama de clases