Universidad de las Ciencias Informáticas
Facultad 2

Título: “Sistema para la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Yadira Ramón Galán
Yordany Bécquer del Toro

Tutores: Ing. Elieser Bello Ross
Co-tutor: Ing. Edgar González Blanco

Ciudad de La Habana, junio 2013
“Año 55 del Triunfo de la Revolución.”

"En la tierra hace falta personas que trabajen más y critiquen menos, que construyan más y destruyan menos, que prometan menos y resuelvan más, que digan mejor hoy que mañana."

Che

DECLARACIÓN DE AUTORÍA.

Declaramos ser los autores de la presente tesis, reconociendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de manera exclusiva.

Para que así conste, firmamos la presente a los ___ días del mes de _____ de _____

_____	_____
Yadira Ramón Galán	Yordany Bécquer del Toro
Firma Autor	Firma Autor
_____	_____
Elieser Bello Ross	Edgar González Blanco
Firma Tutor	Firma Co-tutor

AGRADECIMIENTOS.

A la Universidad de las Ciencias Informáticas *por formarnos como profesionales.*

A nuestro Tutor Eliser Bello y co-tutor Edgar González *por la dedicación y el trabajo diario junto a nosotros.*

A nuestros padres y familiares *por ser nuestra razón de existir.*

A los Amigos *que nos ayudaron con su apoyo incondicional.*

A los profesores *que nos brindaron su sabiduría en varios campos del conocimiento ayudándonos así en varios aspectos que requerimos para el desarrollo de nuestro proyecto.*

A nuestros compañeros *que de varias maneras siempre estuvieron acompañándonos y ayudándonos en los momentos que requeríamos ayuda, por compartir conocimientos y vivencias con nosotros y darnos sentimientos de alegría, amor y cariño que nos dejaran muchas enseñanzas y experiencias.*

A nuestra gran Revolución y a nuestro Comandante Fidel*, que hicieron posible que existiera esta Universidad.*

"Muchas Gracias"

Yadira y Yordany

DEDICATORIA.

***A toda mi familia**, por su gran apoyo y confianza en todo momento, por ser los ángeles que guían mis pasos, siendo la inspiración de mi vida, y por haberme dado los mejores consejos del mundo ayudándome a ser una mejor persona cada día.*

***A mi Mamá**, por ser no solo madre sino amiga, por mostrarme tantas cosas de la vida que no me alcanzan las palabras para decirlas.*

***A mi Papá**, por haberme enseñado que la vida es sacrificio y esfuerzo, y sobre todo por ser el gran padre que es.*

***A mi Abuela**, por cada rezo, por cada palabra de afecto, por tener siempre la palabra correcta que me diera el aliento suficiente para seguir adelante.*

***A mi compañero de tesis**, por formar parte de mi vida de una forma indescriptible e inexplicable, por enseñarme a ver el lado bueno de las cosas malas, y a mostrar siempre una sonrisa aunque el mundo se nos halla derrumbado encima, por ser el hombre que hoy es y la mujer que me hace ser ante la vida, por todo el esfuerzo y el empeño volcado en la realización de este trabajo, por formar junto a mí un gran equipo de trabajo, permitiendo lograr este resultado.*

***A mi profe Abel** quien fue, ha sido y será como un padre para mí desde mi ingreso a la Universidad, gracias a ti también por estar en los momentos difíciles, por los consejos y tus abrazos cuando más lo necesitaba.*

***A mis amigos**, por ser incondicionales en todo momento y haberme aceptado tal y como soy, por todos sus consejos y sus muestras de lo que es verdaderamente una gran amistad.*

***A todas aquellas personas** que he conocido y con las que he compartido estos 5 años de Universidad, aquellas que siguieron y otras que no están pero que nunca olvidaré.*

Yadira Ramón Galán

DEDICATORIA.

A mi familia *que gracias a su apoyo pude concluir mi carrera.*

A mi madre Zaida*, por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una mejor persona, por su preocupación, pero más que nada por su amor.*

A mi padre*, por hacer de mi lo que soy hoy, por la educación que me ha infundado siempre, por ser un ejemplo para mí, por su apoyo, su confianza, su preocupación constante, pero sobre todo por su amor.*

A mi hermano*, por ser la persona que es, por su apoyo incondicional, por brindarme su ayuda en los momentos que más la necesitaba, por su preocupación, por soportarme todos estos años, por ayudarme a ser una mejor persona cada día. No tengo palabras para describir mis agradecimientos hacia ti.*

A Julián*, por brindarme su apoyo en todo momento, por demostrarme que puedo contar con él en cualquier circunstancia, por su amistad y preocupación.*

A mi compañera de tesis*, a ella especialmente le dedico esta Tesis. Por su paciencia, por su comprensión, por su empeño, por su fuerza, por su amor, por ser tal y como es.*

A mis amistades Rodo, Lester, Jorgito, Yunet, Nanis, Yaser, Jose, Maikel, Sandy*, por tener siempre la mano extendida cuando los necesite, por sus consejos, por enseñarme el verdadero valor de la amistad.*

A todas aquellas personas *que de una forma u han contribuido al logro de este resultado y que también forman parte de él.*

Yordany Bécquer del Toro.

RESUMEN.

En la actualidad el mundo está guiado por el uso y desarrollo de las TIC[1], estas a su vez están regidas por la informática como ciencia fundamental y por el uso de programas de cómputo o software, que facilitan el trabajo en las diferentes actividades de la vida diaria.

Prueba de ello es el incremento sustancial del empleo de herramientas para el reconocimiento de patrones con el propósito de extraer información que permita localizar, lo antes posible los datos que se precisan de manera exclusiva, pero debido a que esta información es muy versátil y variable se crean a diario miles de algoritmos para el reconocimiento de patrones que se hacen necesarios probar y evaluar con la finalidad de conocer cuán buenos son sobre un determinado dominio de datos.

El presente trabajo permite desarrollar una aplicación desktop, obteniendo como resultado un sistema amigable y flexible que facilita la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones, con funcionalidades que garantizan la creación de experimentos adicionando al sistema algoritmos en forma de plug-ins y mostrando una gráfica con los resultados, permitiendo realizar cambios en la perspectiva de visualización del agrupamiento. El sistema brinda la posibilidad de validar la calidad del experimento a través de una medida que determina el acoplamiento de los datos resultantes. A su vez el sistema garantiza la realización de comparaciones entre experimentos, a partir de criterios altamente configurables seleccionados por el usuario.

Palabras claves: Algoritmo, agrupamiento, evaluación, reconocimiento de patrones, plug-ins.

[1] TIC: Tecnología de la Información y las Telecomunicaciones.

ÍNDICE.

ÍNDICE DE TABLAS.

INTRODUCCIÓN.

Uno de los principales problemas a los que se enfrenta la sociedad de la información en la actualidad, es la gestión óptima y productiva de la documentación disponible. En otras palabras, diariamente se generan grandes cantidades de datos y es imprescindible establecer técnicas que ayuden a localizar, lo antes posible, la información que resulta relevante según nuestras necesidades. En resumen, es necesaria una correcta organización de la información para que su recuperación sea lo más completa posible.

Es en este punto donde entran en juego los sistemas de reconocimiento de patrones basados en técnicas no supervisadas, empleados para optimizar el tratamiento (obtención, filtrado, clasificado y extracción) de la información, a fin de poder proporcionar al usuario, los datos que precisa de manera exclusiva.

El reconocimiento de patrones se ocupa de los procesos de ingeniería, computación y matemáticas relacionados con objetos físicos o abstractos, con el propósito de extraer información que permita establecer propiedades entre conjuntos de dichos objetos. Un sistema automático de reconocimiento de patrones se puede dividir en tres etapas fundamentales:

- Adquisición de datos, en la que se obtiene una representación del objeto como resultado de un conjunto de mediciones.
- Extracción de características, donde se realiza un proceso interpretativo, cuyo resultado se considera como una nueva representación del objeto de la que se extrae información relevante sobre el mismo.
- Toma de decisiones, que corresponde a la clasificación propiamente dicha o proceso de identificación (1).

El punto esencial del reconocimiento de patrones es la clasificación, la cual trata de asignar las diferentes partes del vector de características a grupos o clases, basándose en las especificidades extraídas de cada uno de los objetos. Según se tenga constancia o no de un conjunto previo que permita al sistema aprender, la clasificación puede ser supervisada, parcialmente supervisada o no supervisada.

La clasificación supervisada es conocida como clasificación con aprendizaje, se basa en la disponibilidad de áreas de entrenamiento en donde se conoce a priori la clase a la que pertenecen cada uno de los objetos y que servirán para generar una signatura espectral, que no son más que las características de cada una de las clases. La clasificación parcialmente supervisada es conocida como clasificación con aprendizaje parcial, en estos problemas existe una muestra de objetos sólo en algunas de las clases

definidas. La clasificación no supervisada es conocida como clasificación sin aprendizaje, en la que se utilizan algoritmos de clasificación automática multivariante en los que los individuos más próximos se van agrupando formando clases (2).

Entre los algoritmos que emplea la clasificación no supervisada se encuentran: Los algoritmos de agrupamiento o clustering, los cuales aplican un procedimiento de agrupación de una serie de vectores de acuerdo con un criterio de cercanía. Esta se define en términos de una determinada función de distancia, como la euclídea, aunque existen otras más robustas o que permiten extenderla a variables discretas (3). La evaluación en los algoritmos de agrupamiento es el proceso que a través de una medida comprueba cuán acoplados se encuentran los datos resultantes. Según su funcionamiento las distintas medidas internas intentan reflejar propiedades estructurales del resultado del agrupamiento, sin embargo, la presencia de estas propiedades estructurales no garantiza la usabilidad de los resultados para el usuario, mientras que una propiedad reflejada por medidas externas determinan hasta qué punto los grupos obtenidos se asemejan a los que se hubieran logrado con una categorización manual real (4).

Actualmente existen herramientas que realizan el reconocimiento de patrones utilizando clasificación no supervisada, contando a su vez con algoritmos de agrupamiento implementados, generalmente los denominados como clásicos, (K-means, DBSCAN) y una evaluación a priori del resultado del agrupamiento conocido también como validación de cluster; pero estas herramientas no se especializan en la evaluación de algoritmos de agrupamiento sino más bien en varios de los procesos que abarcan la minería y el análisis de datos(clasificación, clusterización, asociación). Como tampoco son herramientas especializadas en realizar comparaciones entre experimentos, los criterios de comparación con los que cuentan en su entorno no son configurables, limitando la realización de la selección de los criterios que desea comparar. A su vez la visualización de los resultados no son claros como interpretación más acertada del usuario sin conocimientos de minería o análisis de datos, lo que imposibilita que pueda establecerse de un criterio de manera intuitivo de lo expresado a través del resultado. Para desarrollar o adicionar un nuevo algoritmo a las herramientas ya existentes es necesario conocer su arquitectura y para que estas reconozcan la nueva inserción es necesario recompilar todo el sistema, lo podría representar una problemática en cuanto a tiempo y rendimiento se refiere. Por todo lo anteriormente descrito surge la necesidad de realizar un sistema basado en plug-ins que permita la inserción, evaluación y comparación de nuevos algoritmos o de otros ya existentes de una manera flexible, rápida y amigable.

Analizando la situación anteriormente expuesta, se define como **problema a resolver:** ¿Cómo facilitar la evaluación de los algoritmos de agrupamiento para el reconocimiento de patrones?, para enfrentar esta problemática es **objeto de estudio** de la investigación los procesos para la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones, enmarcando el **campo de acción** en los sistemas para la evaluación de algoritmos de agrupamiento en el reconocimiento de patrones.

A partir de la idea anteriormente expuesta se define el **objetivo general** del presente trabajo como: Desarrollar una aplicación informática para la evaluación de algoritmos de agrupamiento para reconocimiento de patrones.

Se plantean además como **objetivos específicos**:

- Caracterizar el marco teórico conceptual y el estado del arte de las tecnologías actuales que evalúan algoritmos de agrupamiento para el reconocimiento de patrones.
- Diseñar el sistema que permite evaluar algoritmos de agrupamiento para el reconocimiento de patrones.
- Implementar el sistema que permite evaluar algoritmos de agrupamiento para el reconocimiento de patrones.
- Validar el correcto funcionamiento del sistema base.

Para lograr dichos objetivos se plantearon las siguientes **tareas de investigación**:

- Caracterización de las plataformas más utilizadas que empleen la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones.
- Identificación de las tecnologías para la implementación de la plataforma base que permite evaluar algoritmos de agrupamiento para el reconocimiento de patrones.
- Definición del mecanismo de entrada, salida y persistencia de datos.
- Definición de las principales funcionalidades para la evaluación de algoritmos de agrupamiento.
- Definición de la estructura de la arquitectura base para la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones.
- Diseño de los prototipos de interfaz de usuario.
- Desarrollo de las funcionalidades que permitan la ejecución de los experimentos para la evaluación de algoritmos de agrupamiento.
- Desarrollo de las funcionalidades que permitan mostrar y almacenar el resultado de los experimentos realizados en la evaluación de los algoritmos de agrupamiento.

- Validación del sistema implementado mediante el diseño de un experimento.

Para desarrollar este trabajo se ha planteado la siguiente **idea a defender:**
El desarrollo de un sistema informático amigable y flexible facilitaría la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones, permitiendo la persistencia y comparación de los experimentos realizados.
La estructura del documento está definida de la siguiente manera:

Capítulo 1. Fundamentación teórica: Se exponen los conceptos fundamentales relacionados con el tema de investigación y se describen los principales aspectos de los lenguajes, herramientas y metodologías a utilizar para el desarrollo de la aplicación.

Capítulo 2. Descripción y análisis de la solución propuesta: Se realiza una valoración del modelo del negocio del sistema, los procesos que serán objeto de automatización, las fórmulas a aplicar y las estadísticas importantes que se deben obtener. Se describen los requerimientos con los que cumplirá el sistema. Se realiza una valoración del análisis y diseño del sistema en términos de componentes de implementación. Se describen los algoritmos a implementar analizándose la complejidad de los mismos. Se efectúa la selección de las estructuras de datos apropiadas para la implementación de estos algoritmos y se realiza la descripción de las clases u operaciones que se utilicen para representar computacionalmente dichas estructuras.

Capítulo 3. Implementación y validación de la solución propuesta Se desarrolla el estudio y diseño de la validación de la solución propuesta. Se realiza una descripción de estas teniendo en cuenta objetivo, alcance y detalles de la misma. Se efectúa la descripción de los valores utilizados para las validaciones, así como una evaluación de la ejecución de estas y de los resultados obtenidos.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

A lo largo de este tiempo, muchas ciencias han participado en el desarrollo exitoso de herramientas con el fin de solucionar diversas situaciones que permitan agilizar los procesos vinculados a la gestión de la información. En el presente capítulo se exponen los conceptos fundamentales relacionados con el tema de investigación. Se analizan herramientas que permiten evaluar algoritmos de agrupamiento para el reconocimiento de patrones de tal forma que puedan ser valoradas las deficiencias que estas presentan ante las ventajas de la solución que se propone. Se describen los principales aspectos de los lenguajes, herramientas y metodologías a utilizar para el desarrollo de la aplicación.

1.2. Clasificación No supervisada: Algoritmos de agrupamiento.

El aprendizaje no supervisado es muy importante cuando tenemos muestras sin etiquetas de clase, cuando el costo de etiquetarlas por un experto es alto o cuando los patrones pueden variar con el tiempo, por lo que es necesario primero procesar los datos para luego clasificar. La principal ventaja que presenta la clasificación no supervisada es que se puede obtener un conjunto de entrenamiento empleando muestras no etiquetadas (2).

Actualmente, en muchas aplicaciones reales (biometría, categorización de textos, búsqueda en bases de datos multimedia, reconocimiento de imágenes multiespectrales, etc.), el coste de un conjunto de entrenamiento resulta bastante alto, por lo que podría ser beneficioso aplicar primero a determinadas muestras cuya clase se desconoce, un algoritmo de agrupamiento para luego inferir propiedades en la población en estudio. Se trata de construir clasificadores sin información previa, o sea, a partir de objetos no etiquetados con el objetivo de descubrir la estructura de los datos.

Bajo el nombre genérico de algoritmos de agrupamiento, se incluyen todo un conjunto de procesos cuya finalidad general será dividir un conjunto de objetos en clases, para obtener un subconjunto representativo del conjunto de entrenamiento inicial que posteriormente pueda utilizarse en una regla de clasificación supervisada.

En general, la clasificación no supervisada o agrupamiento (clustering) consiste en dividir el conjunto de objetos en grupos de objetos similares llamados clusters, de modo tal que objetos pertenecientes a un mismo grupo son más similares que objetos de grupos diferentes.

El problema de formar grupos en un conjunto de datos es muy importante para el conocimiento del comportamiento de una población, de la cual sólo se tiene una cantidad N de sus elementos. Profundizar sobre el proceso de división en clases, permite definir que cada técnica está diseñada para realizar una clasificación de tal modo que cada grupo sea lo más homogéneo y lo más diferente de los demás como sea posible. El resultado de cada método de agrupamiento dependerá del algoritmo en concreto, del valor de los parámetros y de la medida de similaridad / disimilaridad adoptada (2).

1.2.1. Formulación de un problema de clasificación no supervisada.

Sea $X = \{x_1, x_2, \dots, x_N\}$ el conjunto de datos o, análogamente, objetos, ejemplos, casos, patrones, n-tuplas, puntos, donde$x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ pertenece a un espacio de atributos, para cada $i = 1, \dots, N$, y cada componente $x_{ij} = (j = 1, \dots n)$ es un atributo (análogamente, rasgo, variable, dimensión o componente) de modo tal que el conjunto de objetos forma una matriz Nxn empleada por la mayoría de los algoritmos de agrupamiento.

La meta de todo algoritmo de agrupamiento es asignar cada punto a un sistema finito de subconjuntos o clusters que usualmente no se intersectan entre sí y cuya unión es igual al conjunto de datos completo con la posible excepción de outliers[2], de modo tal que objetos similares pertenezcan al mismo cluster, mientras que los objetos de clusters diferentes sean lo menos parecidos posible.

Los algoritmos de agrupamiento pueden dividirse en varias categorías según el procedimiento que utilizan para agrupar los objetos:

1. Algoritmos jerárquicos, que pueden ser aglomerativos y divisivos.
2. Métodos por partición, entre ellos: algoritmos de reubicación, agrupamientos probabilísticos, métodos de k-medoides y métodos k-Medias (k-Means).
3. Algoritmos basados en densidad, entre ellos los algoritmos de agrupamiento por conectividad basados en densidad y los agrupamientos basados en funciones de densidad.
4. Métodos basados en rejillas.
5. Algoritmos mixtos (2).

[2] Outliers: Ruido o valores atípicos arrojados por el agrupamiento.

1.2.2. Distancias y Métricas.

Para realizar el agrupamiento de los objetos, es necesario determinar cuándo dos objetos del espacio son "parecidos" y cuándo no. Con este fin, se definen las métricas o distancias. Muchos de los algoritmos de agrupamiento basan su efectividad en la distribución de los objetos del conjunto de datos en el espacio y en cuán alejados están entre ellos. Es por ello que es preciso definir alguna medida de distancia entre los objetos de X, mediante la cual pueda asignársele a cada muestra, una clase determinada (2).

Definición: Un espacio métrico es un par(X, d) donde X es un conjunto $(X \neq \emptyset)$y d una distancia o métrica definida sobre X. Una función $\mathrm{d}: \mathrm{XxX} \rightarrow R^{+}$, se dice que es una distancia o métrica si satisface los siguientes axiomas:

1. d(x, y) ≥0 ∀ x, y ∈ X, y d(x, y) = 0 si y solo si x = y
2. d(x, y) = d(y, x) ∀ x, y ∈ X (simetría)
3. d(x, z) ≤ d(x, y)+ d(y, z) ∀ x, y, z ∈ X (desigualdad triangular)

La métrica más frecuentemente utilizada es la métrica Euclídea:

$$(\mathrm{O_i}, \mathrm{O_j}) \sqrt{\sum_{k=1}^{n} (\mathrm{x_k}(\mathrm{O_i}) - \mathrm{x_k}(\mathrm{O_j}))^2}$$

donde $\mathrm{O_i}$ y $\mathrm{O_j}$ son los objetos para los cuales se desea calcular la distancia, n es el número de características de los objetos del espacio y $\mathrm{x_k}(\mathrm{O_i})$, $\mathrm{x_k}(\mathrm{O_j})$ es el valor del atributo k-ésimo en los objetos $\mathrm{O_i}$ y $\mathrm{O_j}$ respectivamente (2).

1.3. Sistemas Informáticos vinculados al campo de acción.

Una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Hoy en día existen disímiles herramientas de este tipo que permiten realizar experimentos evaluando algoritmos de agrupamiento para el reconocimiento de patrones.

Weka[3]: es una plataforma de software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato. El paquete Weka contiene una colección de herramientas

[3] WEKA: Siglas del inglés Waikato Environment for Knowledge Analysis

de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades.

Características:

- Está disponible libremente bajo la licencia pública general de GNU.
- Multiplataforma.
- Contiene una extensa colección de técnicas para preprocesamiento de datos y modelado (5).

Validación de los resultados:

El resultado de aplicar el algoritmo de clasificación se efectúa comparando la clase predicha con la clase real de las instancias. Esta evaluación puede realizarse de diferentes modos:

- **Use training set:** esta opción evalúa el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo para determinar el error, que en este caso se denomina "error de resustitución". Por tanto, esta opción puede proporcionar una estimación demasiado optimista del comportamiento del clasificador, al evaluarlo sobre el mismo conjunto sobre el que se hizo el modelo.
- **Supplied test set**: evaluación sobre conjunto independiente. Esta opción permite cargar un conjunto nuevo de datos. Sobre cada dato se realizará una predicción de clase para contar los errores.
- **Cross-validation:** Evaluación con validación cruzada. Esta opción es la más elaborada y costosa. Se realizan tantas evaluaciones como se indica en el parámetro Folds. Se dividen las instancias en tantas carpetas como indica este parámetro y en cada evaluación se toman las instancias de cada carpeta como datos de test, y el resto como datos de entrenamiento para construir el modelo. Los errores calculados son el promedio de todas las ejecuciones.
- **Percentage split :** esta opción divide los datos en dos grupos, de acuerdo con el porcentaje indicado (%). El valor indicado es el porcentaje de instancias para construir el modelo, que a continuación es evaluado sobre las que se han dejado aparte. Cuando el número de instancias es suficientemente elevado, esta opción es suficiente para estimar con precisión las prestaciones del clasificador en el dominio (6).

Matlab[4]: es una herramienta de software matemático con lenguaje de programación propio. Está disponible para las plataformas Unix, Windows y Mac OS X.

[4] Matlab: Siglas del inglés MATriz LABoraty

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden dichas prestaciones: las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets) (7). Una de las cajas de herramientas denominada: caja de herramienta de agrupamiento porporciona cuatros catergorías de funciones para el manejo de datos a clusterizar:

1. Algoritmos de agrupamiento.
2. Evaluación con los prototipos del racimo.
3. Validación.
4. Visualización (8).

Validación de los resultados:

La función de la validez proporciona las medidas de la validez del racimo para cada partición. Es útil cuando el número de racimo es a priori desconocido. La partición óptima se puede determinar por el punto de los extremos de los índices de la validación en la dependencia del número de racimos. Los índices calculados son: Repartir el coeficiente (PC), la entropía de la clasificación (CE), el índice de la partición (SC), el índice de separación (s), el índice de Xie y de Beni (XB), el índice de Dunn (DI) y el índice de Dunn de la alternativa (DII) (8).

Rapidminer: es un programa informático para el análisis y minería de datos. Permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico. Se usa en investigación educación, capacitación, creación rápida de prototipos y en aplicaciones empresariales. La versión inicial fue desarrollada por el departamento de inteligencia artificial de la Universidad de Dortmund en 2001.

RapidMiner proporciona más de 500 operadores orientados al análisis de datos, incluyendo los necesarios para realizar operaciones de entrada y salida, preprocesamiento de datos y visualización. También permite utilizar los algoritmos incluidos en Weka.

Características:

- Desarrollado en Java.
- Multiplataforma.
- Representación interna de los procesos de análisis de datos en ficheros XML.

- Permite el desarrollo de programas a través de un lenguaje de script.
- Incluye gráficos y herramientas de visualización de datos.
- Dispone de un módulo de integración con R (9).

Validación de los resultados:

En muchos casos el modelo aprendido no es de interés sino la exactitud del modelo. Una posible solución para estimar la precisión del modelo aprendido es aplicarlo a datos de prueba etiquetados y calcular la cantidad de errores de predicción (u otros criterios de performance). Debido a que los datos etiquetados son poco frecuentes, a menudo se usan otros enfoques para estimar la performance de un esquema de aprendizaje. Este proceso muestra la "validación cruzada" en RapidMiner.

La validación cruzada divide los datos etiquetados en conjuntos de entrenamiento y de prueba. Los modelos se aprenden sobre los datos de entrenamiento y se aplican sobre los datos de prueba. Los errores de predicción se calculan y promedian para todos los subconjuntos. Este bloque de construcción se puede utilizar como operador interno para varios wrappers (contenedores) como los operadores de generación/selección de características (9).

R: es un lenguaje y entorno de programación para análisis estadístico y gráfico. Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S.

Características:

- R proporciona un amplio abanico de herramientas estadísticas (modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas.
- Al igual que S, se trata de un lenguaje de programación, lo que permite que los usuarios lo extiendan definiendo sus propias funciones. De hecho, gran parte de las funciones de R están escritas en el mismo R, aunque para algoritmos computacionalmente exigentes es posible desarrollar bibliotecas en C, C++ o Fortran que se cargan dinámicamente. Los usuarios más avanzados pueden también manipular los objetos de R directamente desde código desarrollado en C. R también puede extenderse a través de paquetes desarrollados por su comunidad de usuarios (10).
- Capacidad gráfica, que permite generar gráficos con alta calidad. R posee su propio formato para la documentación basado en LaTeX.
- Herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas. Se ha desarrollado una interfaz, RWeka para interactuar con Weka que permite

leer y escribir ficheros en el formato .arff y enriquecer R con los algoritmos de minería de datos de dicha plataforma (11).

Validación de los resultados:

La solución de agrupación puede ser contrastada, validada, con respecto a un criterio interno o bien externo.

1. La validación cruzada o replicación, siempre que el número de sujetos sea suficientemente grande.
2. Alternativamente aplicar un AD[5] para establecer si los grupos obtenidos son realmente diferentes y cuáles son las variables que contribuyen a esa diferencia.
3. Validación externa. Valorar la solución del AC[6] en comparación a un criterio externo a la investigación (12).

1.4. Evaluación del resultado del agrupamiento.

La evaluación del resultado del agrupamiento a veces se conoce como validación de clúster. Este proceso utiliza una medida que permite evaluar cuán acoplados se encuentran los datos resultantes. Según su funcionamiento las distintas medidas internas (u objetivas) como el índice de Davies-Boulding o el índice de Dunn, intentan reflejar propiedades estructurales del resultado del agrupamiento, sin embargo, la presencia de estas propiedades estructurales no garantiza la usabilidad de los resultados para el usuario, mientras que una propiedad reflejada por medidas externas como la medida F determinan hasta qué punto los grupos obtenidos se asemejan a los que se hubieran logrado con una categorización manual real (4).

1.4.1. La evaluación interna.

Cuando el resultado de un agrupamiento se evalúa sobre los datos que se agrupan en sí, se le denomina evaluación interna. Estos métodos suelen asignar la mejor puntuación en el algoritmo que produce racimos con alta similitud dentro de un clúster y baja similitud entre los grupos. Uno de los inconvenientes de la utilización de criterios internos de evaluación es que altas puntuaciones en una medida interna no resultan necesariamente en las aplicaciones eficaces de recuperación de

[5] AD: Criterio de formación de los grupos que determina la importancia que cada variable medida tiene para discriminar a los grupos entre sí.

[6] AC: Criterio de formación de grupos donde se concede a todas las variables el mismo peso

información (13). Además, esta evaluación está sesgada hacia los algoritmos que utilizan el modelo de clúster. Por ejemplo k-means clustering naturalmente optimiza distancias del objeto y de un criterio interno basado en la distancia probable que sobrevalora el agrupamiento resultante.

Por lo tanto, las medidas de evaluación interna son los más adecuadas para conseguir una cierta penetración en situaciones en las que un algoritmo tiene un mejor rendimiento que otro, pero esto no implica que un algoritmo produzca resultados más válidos que otros (14). La validez de la medida por este índice depende de la afirmación de que este tipo de estructura existe en el conjunto de datos. Un algoritmo diseñado para algún tipo de modelos no tiene ninguna posibilidad si el conjunto de datos contiene un conjunto radicalmente diferente de modelos, o si la evaluación mide un criterio radicalmente diferente. Por ejemplo, la agrupación de k-medias sólo puede encontrar cluster convexos, y muchos índices de evaluación asumen grupos convexos. Los siguientes métodos se pueden utilizar para evaluar la calidad de algoritmos de agrupamiento basado en criterio interno:

- Índice Davies-Bouldin:

El índice de Davies-Bouldin se puede calcular por la siguiente fórmula: $DB = \frac{1}{n}\sum_{i=1}^{n} max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$

donde n es el número de grupos, c_i es el centroide del cluster i , σ_i es la distancia media de todos los elementos en el cluster i de centroide c_i , y $d(c_i, c_j)$ es la distancia entre los centroides c_i y c_j. Dado que los algoritmos que producen grupos con una baja distancia intra-grupos (alta similitud intra-grupo) y altas distancias inter-grupos (baja similitud entre clusters) tendrá un bajo índice de Davies-Bouldin, el algoritmo de agrupamiento que produce una colección de grupos con el más pequeño índice de Davies-Bouldin es considerado el mejor algoritmo basado en este criterio.

- Índice Dunn:

El índice de Dunn pretende identificar grupos densos y bien separados. Se define como la relación entre la mínima distancia entre clusters a la máxima distancia dentro del clúster. Para cada partición de clúster, el índice de Dunn se puede calcular por la siguiente fórmula: $D = min_{1 \leq i \leq n} \left\{ min_{1 \leq j \leq n, i \neq j} \left\{ \frac{d(i,j)}{max_{1 \leq k \leq n} d'(k)} \right\} \right\}$

donde $d(i,j)$ representa la distancia entre los clusters i y j, y $d'(k)$ la distancia intra-grupo de clúster k. La distancia intra-clusters $d(i,j)$ entre dos grupos puede ser cualquier número de medidas de distancia, tales como la distancia entre los centroides de los clusters. De manera similar, la distancia intra-cluster $d'(k)$ se puede medir en una variedad de formas, tales como la distancia máxima entre cualquier par de elementos en el grupo k. Como criterio interno de buscar grupos con alta similitud

intra-cluster y baja similitud entre clusters, los algoritmos que producen grupos con alto índice de Dunn son más deseables.

1.4.2. La evaluación externa.

En la evaluación externa, los resultados del agrupamiento se evalúan con datos que no se utilizan para la realización del propio agrupamiento, tales como etiquetas de clase conocidas y puntos de referencia externos. Estos tipos de métodos de evaluación deben medir la proximidad del agrupamiento a las clases de referencia predeterminadas. Sin embargo, recientemente se ha discutido si esto es adecuado para datos reales, o sólo en los conjuntos de datos sintéticos, ya que las clases pueden contener estructura interna, los atributos actuales no pueden permitir la separación de los grupos o las clases pueden contener anomalías (15). Además, desde un punto de vista del descubrimiento de conocimiento, la reproducción del conocimiento conocido puede no ser necesariamente el resultado pretendido. Algunas de las medidas de calidad de un algoritmo de agrupamiento con criterio externo incluyen:

- Medida Rand:

El índice de Rand calcula la similitud de las agrupaciones (devuelto por el algoritmo de agrupamiento) según las clasificaciones de referencia. También se puede ver el índice de Rand como una medida del porcentaje de decisiones correctas producidas por el algoritmo. Se puede calcular utilizando la siguiente fórmula (16): $RI = \frac{TP+TN}{TP+FP+FN+TN}$

Donde TP es el número de verdaderos positivos, TN es el número de verdaderos negativos, FP es el número de falsos positivos, y FN es el número de falsos negativos. Un problema con el índice de Rand es que los falsos positivos y falsos negativos son igualmente ponderados. Esto puede ser una característica indeseable para algunas aplicaciones de agrupamiento. La medida-F responde a esa preocupación.

- Medida-F:

La Medida-F puede utilizarse para equilibrar la aportación de falsos negativos por retirada de ponderación a través de un parámetro $\beta \geq 0$, se definen como sigue (13): $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$

Se debe tener en cuenta que cuando $\beta = 0$, $F_0 = P$. En otras palabras, la rellamada no tiene impacto en la medida-F cuando $\beta = 0$, y el aumento β asigna una cantidad cada vez mayor de peso para recordar en la final de la medida-F.

- Par de conteo de medida-F:

Par de conteo de medida-F es la medida aplicada para el conjunto de pares de objetos, donde los objetos son emparejados entre sí cuando son parte del mismo agrupamiento. Esta medida es capaz de comparar clusterings con diferente número de racimos.

- Índice Jaccard:

El índice de Jaccard se utiliza para cuantificar la similitud entre dos conjuntos de datos. El índice de Jaccard toma un valor entre 0 y 1. Un índice de 1 significa que los conjuntos de datos son idénticos, y un índice de 0 indica que los datos no tienen elementos comunes. El índice de Jaccard se define por la fórmula siguiente: $J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP+FP+FN}$

Esto es simplemente el número de elementos únicos comunes a ambos conjuntos dividido por el número total de elementos únicos en ambos conjuntos.

- Índice Fowlkes-Mallows:

El índice Fowlkes-Mallows calcula la similitud entre los grupos devueltos por el algoritmo de agrupamiento y las clasificaciones de referencia. Cuanto mayor sea el valor del índice de Fowlkes-Mallows son más similares los grupos y las clasificaciones de referencia. Se puede calcular utilizando la siguiente fórmula (17): $FM = \sqrt{\frac{TP}{TP+FP} * \frac{TP}{TP+FN}}$

donde TP es el número de verdaderos positivos, FP es el número de falsos positivos, y FN es el número de falsos negativos. El índice FM es la media geométrica de la precisión y la recuperación P y R, mientras que el F-medida es su media armónica (18). Además, la precisión y la recuperación son también conocidos como índices de Wallace β^I y β^{II}.

- Matriz de confusión:

Una matriz de confusión se puede utilizar para visualizar rápidamente los resultados de una clasificación (o agrupamiento) algoritmo. Se muestra cómo un clúster es diferente del clúster estándar.

- Información mutua:

La información mutua es una medida teórica de la información de la cantidad de información compartida entre una agrupación y una clasificación real que puede detectar una similitud no lineal

entre dos agrupamientos. La información mutua ajustada es la variante corregida de cambios de la información mutua, que tiene un sesgo reducido para números variables de cluster.

1.5. Plug-ins.

La palabra plug-in viene del inglés y significa: complemento para aportarle una nueva funcionalidad muy específica a una aplicación. Esta funcionalidad adicional es ejecutada por la aplicación principal e interactúan por medio de la API[7] (19).

Beneficios de un sistema de plug-ins:

Desde el punto de vista del desarrollo, un sistema de plug-ins tiene diversas ventajas:

- Alta independencia entre módulos (también baja cohesión entre módulos). Puesto que cada módulo es, en principio, relativamente independiente de todos los demás es más fácil desarrollar pruebas mediante stubs y drivers sin que se vean afectadas por el resto de componentes de la aplicación
- Facilidad para extender la aplicación sin necesidad de redistribuir un nuevo ejecutable.
- Si los plug-ins que son desarrollados son los suficientemente genéricos pueden reutilizarse en otras aplicaciones con lo que el periodo de desarrollo es significativamente menor (el necesario para adaptar el plug-in al nuevo sistema).
- Facilidad para adaptarse a cambios en los requisitos añadiendo o modificando funcionalidad mediante la creación o modificación de un plug-in adecuado (20).

1.6. Valoración crítica de los sistemas analizados.

Los algoritmos de agrupamiento son técnicas muy utilizadas en distintos contextos como la recuperación de información, la minería de textos, la segmentación de imágenes y la compresión de datos con el objetivo de lograr una gestión óptima de la información que resulta relevante. Las herramientas que hoy en día utilizan clusterización para el reconocimiento de patrones tiene un funcionamiento similar: traen consigo implementados algoritmos de agrupamiento que permiten recrear dicho proceso para un determinado dominio de datos, pero si se desea añadir un nuevo algoritmo es necesario recompilar toda la aplicación y conocer a fondo el modelo arquitectónico en el que fue desarrollado dicho algoritmo.

[7] API: Siglas del inglés Application Programming Interface

A su vez estas herramientas no fueron desarrolladas con la finalidad de permitir al usuario realizar evaluaciones y comparaciones sobre los distintos experimento realizados, lo que provoca que las interpretaciones de los resultados no sean las adecuadas ni las mejores, pues en ocasiones el usuario no logra llevar a cabo una interpretación de manera intuitiva lo que le hace recurrir a métodos manuales. La herramienta que se propone permite que el usuario se centre única y exclusivamente en la realización del nuevo algoritmo haciendo uso de un alto grado de abstracción con relación a las funcionalidades del sistema que más tarde le permitirán realizar el nuevo experimento.

Por otra parte, desarrollar un sistema basado en plug-ins mejora las capacidades de la aplicación en cuanto rendimiento y tiempo de ejecución se refiere. Las comparaciones que permite el sistema realizar son claras y precisas pues el usuario no necesita desarrollar una nueva investigación para hacerse de una valoración crítica acerca de lo arrojado por los datos resultantes. Por último, al usar una medida para validar la calidad del agrupamiento el sistema garantiza la usabilidad de los resultados para el usuario, siendo este el objetivo final de la realización de los experimentos.

1.7. Técnicas y tecnología utilizada para la implementación del sistema.

El desarrollo y el avance alcanzado por la informática ha brindado la posibilidad de impulsar el perfeccionamiento de las metodologías, herramientas, lenguajes y tecnologías para la construcción y desarrollo de aplicaciones que posibilitan un mejor manejo y procesamiento de la información.

1.7.1. Herramientas.

Actualmente se consideran a las Herramientas de Desarrollo de Software (HDS) como herramientas basadas en computadoras que asisten el proceso de ciclo de vida de una aplicación, consolidadas en la literatura en la forma de Ingeniería de software asistida por computadora (CASE, por sus siglas en inglés) (21). Entre las HDS que automatizan metodologías de software se escogió a Visual Paradigm.

1.7.1.1. Herramienta de Modelado.

Visual Paradigm v6.4: es una herramienta CASE[8]. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista (22).

[8] CASE: Ingeniería de Software Asistida por Computación

Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos.

Se caracteriza por:

- Software libre.
- Disponibilidad en múltiples plataformas (Windows, Linux).
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Varios idiomas.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento (22).

1.7.1.2. Entorno de desarrollo integrado.

Un IDE[9] es un software compuesto por diferentes herramientas y funciones que permiten el trabajo con uno o más lenguajes de programación. Es un entorno que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Provee un marco de trabajo amigable para los lenguajes de programación. Entre los IDE que existen actualmente se encuentran: NetBeans, Eclipse y Visual Studio (23), para el desarrollo de la aplicación se escogió el Visual Studio.

Visual Studio 2010.

Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET., aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles (24).

[9] IDE: Siglas del inglés *integrated development environment*)

1.7.2. Lenguajes.

Un lenguaje en el contexto informático es un idioma artificial diseñado para expresar procesos que pueden ser llevadas a cabo por las máquinas, lo que proporciona un vocabulario para lograr una comunicación entre el equipo de desarrollo y los ordenadores (25).

1.7.2.1. Lenguaje de Modelado.

Lenguaje Unificado de Modelado (Unified Modeling Language, UML)

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y una regla para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema.

Este lenguaje nos indica cómo crear y leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Está compuesto por tres clases de bloques de construcción:

- Elementos: Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: Son colecciones de elementos con sus relaciones (26).

1.7.2.2. Lenguaje de Programación

C Sharp

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

Dentro de sus características fundamentales están:

- Lenguaje de programación orientado a objetos simple, moderno y de propósito general.
- Inclusión de principios de ingeniería de software tales como revisión estricta de los tipos de datos, revisión de límites de vectores, detección de intentos de usar variables no inicializadas, y recolección de basura automática.
- Capacidad para desarrollar componentes de software que se puedan usar en ambientes distribuidos.
- Portabilidad del código fuente.
- Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C, C++ y Java.
- Soporte para internacionalización.
- Adecuación para escribir aplicaciones de cualquier tamaño: desde las más grandes y sofisticadas como sistemas operativos hasta las más pequeñas funciones.
- Aplicaciones económicas en cuanto a memoria y procesado (24).

1.7.3. Metodología de desarrollo.

El desarrollo de un software confiable y eficiente depende en gran medida de la definición de una metodología de desarrollo adecuada, en correspondencia con las necesidades del equipo de trabajo. Una metodología es un conjunto de procedimientos, técnicas, herramientas que definen: qué debe hacerse, cuándo, quién y cómo debe hacerlo, con el objetivo de ayudar a los desarrolladores a realizar un software. Las metodologías de desarrollo se dividen en dos grandes grupos: las metodologías robustas y las metodologías ágiles.

Para el caso de la aplicación que se desarrollará, debido a que es un proyecto pequeño que necesita ser elaborado en el menor tiempo posible y los requisitos pueden cambiar frecuentemente, se necesita una metodología ágil que permita la interacción constante del equipo de desarrollo con el cliente, y el desarrollo de un software incremental, sencillo y adaptable en poco tiempo con pocas personas.

Dentro de las metodologías ágiles o ligeras más usadas está Extreme Programming ó Programación Extrema (XP) donde el cliente forma parte del equipo de desarrollo, realizando pequeñas iteraciones y

mini entregas cada dos semanas, donde no existe más documentación que el propio código. Permite realizar cambios rápidamente al producto y finalizarlo en un corto plazo de tiempo con la mayor calidad posible (27).

"Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la in capacidad de adaptarnos a dicho cambio cuando éste tiene lugar."

Kent Beck (28).

Las características fundamentales de XP son:

- Desarrollo iterativo e incremental: Pequeñas mejoras, unas tras otras.
- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes tan pequeñas como sea posible.
- Estándares de codificación: Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte de este, es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP (29).

Para el desarrollo del sistema actual se decidió por todas las características antes mencionadas utilizar la Metodología de Desarrollo XP.

1.7.3.1. Proceso de desarrollo de Software.

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas (30).

Para el desarrollo del sistema fue seleccionado el proceso de desarrollo propuesto por una de las metodologías ágiles (XP), este a su vez utiliza un desarrollo iterativo como base para abogar por un punto de vista más ligero y más centrado en las personas que en el caso de las soluciones tradicionales. Los procesos ágiles utilizan retroalimentación en lugar de planificación, como principal mecanismo de control. La retroalimentación se canaliza por medio de pruebas periódicas y frecuentes versiones del software. Este proceso de desarrollo cuenta con los roles siguientes (29):

- **Programador:** El programador escribe las pruebas unitarias y produce el código del sistema.
- **Cliente:** Escribe las historias de usuario y las pruebas funcionales para validar su implementación.
- **Encargado de pruebas (Tester):** Ayuda al cliente a escribir las pruebas funcionales.
- **Encargado de seguimiento (Tracker):** Proporciona realimentación al equipo.
- **Entrenador (Coach):** Es responsable del proceso global.
- **Consultor:** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (B ig boss):** Es el vínculo entre clientes y programadores.

El ciclo de vida ideal para el proceso de desarrollo en el que se aplica un modelo ágil consiste de seis fases:

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 de la Entrega (Release): En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores 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. Una entrega debería obtenerse en no más de tres meses.

Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. Al final de la última iteración el sistema estará listo para entrar en producción.

Producción: La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción.

Muerte del Proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura (28).

1.7.4. Marco de Trabajo.

Los Frameworks (Marcos de Trabajo), no son más que arquitecturas definidas para un determinado dominio de la aplicación que contiene un conjunto de componentes implementados y sus interfaces bien definidas, estos componentes se pueden utilizar, redefinir y crear nuevos componentes (31).

Framework de .net

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación.
- Biblioteca de clases de .Net.
- CLR (Common Language Runtime).

Actualmente, el Framework de .Net es una plataforma no incluida en los diferentes sistemas operativos distribuidos por Microsoft, por lo que es necesaria su instalación previa a la ejecución de programas creados mediante .Net. Este framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes (24).

1.7.4.1. Arquitectura.

El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura de software es el conjunto de decisiones significativas sobre la organización de un sistema, la selección de los elementos estructurales y sus interfaces de los cuales el sistema está compuesto junto con su comportamiento. Describe los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente (32).

"… es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos (...)"

PRESSMAN (33)

“Conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema, y las interfaces entre ellos, junto con su comportamiento, (...)”

RUMBAUGH (34)

La arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las siguientes superiores a esta.

Los beneficios de trabajar un sistema en capas son:

1. Se puede entender una capa como un todo, sin considerar las otras.
2. Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
3. Se minimizan dependencias entre capas.
4. Las capas posibilitan la estandarización de servicios.
5. Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel (32).

1.8. Conclusión parcial.

En este capítulo se exponen conceptos y definiciones que están relacionados con el marco teórico conceptual y el estado de arte de las nuevas tecnologías que evalúan algoritmos de agrupamiento para el reconocimiento de patrones, así como la especificación de las tecnologías y herramientas que se utilizaron para el desarrollo de la solución.

CAPÍTULO 2

CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción.

En el presente capítulo se conforma una propuesta del sistema a implementar, se expresan las características del sistema para un mejor entendimiento, así como la descripción y modelación de los procesos, detallando las historias de usuarios que brindan una mejor visión sobre lo que el cliente quiere. Se hace mención además la fase de diseño propia de la metodología de desarrollo utilizada. Se exponen los artefactos generados durante el transcurso de la misma.

2.2. Flujo del proceso.

2.2.1. Valoración crítica de los requerimientos propuestos.

Los requisitos funcionales descritos del sistema facilitaron el entendimiento de los procesos a implementar permitiendo una buena comprensión del problema y posibilitando la identificación de las clases y las funcionalidades a desarrollar. Tomándose como base la descripción detallada de los requisitos funcionales, se puede establecer una estrategia de trabajo que permita la implementación de la capa lógica, de presentación y de datos de la aplicación.

Especificación de los requerimientos del software:

Un requerimiento según la IEEE Standard Glossary of Software Engineering Terminology se puede definir como una:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 o 2.

Los requisitos se pueden clasificar en funcionales y no funcionales. Los requisitos funcionales descritos que posibilitaron la identificación de las clases y las funcionalidades a implementar son:

RF1.Gestionar plug-in.

RF1.1.Adicionar plug-in.

RF1.1.1.Buscar plug-in.

RF1.1.2.Validar adición.

RF1.2.Listar plug-ins.

RF1.3.Eliminar plug-in.

RF2.Crear experimento.

RF2.1.Guardar experimento.

RF2.1.1.Buscar ubicación del experimento.

RF2.2.Seleccionar dominio de datos.

RF2.3.Buscar dominio de datos.

RF2.4.Seleccionar medida de distancia.

RF2.5.Seleccionar algoritmo de agrupamiento.

RF2.6.Ejecutar algoritmo de agrupamiento.

RF2.7.Registrar descripción general del experimento.

RF3.Ver experimento.

RF3.1.Cargar experimento.

RF3.1.1.Buscar experimento.

RF3.2.Mostrar datos del experimento.

RF3.3.Mostrar descripción del experimento.

RF3.4.Mostrar agrupamiento.

RF3.5.Mostrar gráfica del resultado del algoritmo de agrupamiento realizado.

RF3.6.Mostrar evaluación de medida de calidad.

RF4.Comparar experimento.

RF4.1.Cargar experimento a comparar.

RF4.1.1.Buscar experimento a comparar.

RF4.2.Incluir experimento.

RF4.3.Excluir experimento.

RF4.4.Excluir todos los experimentos.

RF4.5.Seleccionar criterios de comparación.

RF4.6.Guardar resultado de comparación.

RF5.Generar ayuda.

Los requisitos no funcionales descritos que posibilitaron describir las propiedades que el producto deberá tener son:

1. **Requerimientos de software:**

RNF1.1.Sistema Operativo: Windows.

2. **Requerimientos de hardware:**

RNF2.1.Procesado mínimo: procesadora 2.0 GHz; recomendado superior.

RNF2.2.RAM mínima: 1GB; recomendada: 2GB.

RNF2.3Espacio mínimo de disco duro: 1GB disponibles; recomendado: 2 GB disponible.

3. Requerimientos de usabilidad:

RNF3.1.Se colocará la menor cantidad posible de campos en los formularios del sistema, solo aquellos que sean necesarios y suficientes.

RNF3.2.Los campos de texto tendrán un tamaño estándar de acuerdo con el espacio con que se cuente en el área de la página y en la medida que se llene esa área primaria agregar la barra de desplazamiento vertical o algún mecanismo que permita visualizar la información introducida.

RNF3.3.No se utilizarán textos extensos para las etiquetas de la interfaz de usuario.

RNF3.4.Los campos obligatorios serán señalados de forma tal que los usuarios logren identificarlos. Solo se señalarán los campos obligatorios luego de que el usuario los haya obviado en la realización de alguna funcionalidad.

RNF3.5.El sistema diferenciará los mensajes de información de los mensajes de error y de advertencia, mostrando mensajes de textos personalizados para cada uno de ellos.

4. Requerimientos de diseño y de implementación:

RNF4.1.El sistema implementado será una aplicación desktop.

RNF4.2.El sistema se implementará usando Visual Studio 2010.

RNF4.3.El sistema estará basado en un estilo arquitectónico en capas.

RNF4.4.El sistema será diseñado siguiendo los principios de diseño orientado a objetos

5. Requerimientos de apariencia o interfaz externa:

RNF5.1.El sistema brindará una interfaz amigable para sus usuarios.

RNF5.2.El sistema brindará una interfaz configurable para sus usuarios.

RNF5.3.El sistema proporcionará claridad y organización en la información, permitiendo la interpretación correcta e inequívoca de la misma.

RNF5.4.Todos los textos y mensajes en pantalla aparecerán en idioma español. Los errores serán visibles al usuario e incluirán sugerencias de las posibles soluciones.

2.2.2. Propuesta del sistema.

El sistema se encuentra conformado por una ventana principal que contiene un botón inicio que permitirá acceder a las funcionalidades y configuraciones del sistema, los accesos directos que garantizan la usabilidad de estas funcionalidades de forma fácil para el usuario, permitiendo ocultar o mostrar las funcionalidades que el usuario encuentre necesarias o no, además de poder minimizar o maximizar la barra de botones. Esta barra cuenta con dos pestañas: Principal y Configuración, la

primera contiene las funcionalidades del sistema y la segunda permite cambiar el entorno de trabajo del sistema convirtiendo su usabilidad en algo aún más agradable para el usuario.

Para llevar a cabo la evaluación de los algoritmos de agrupamiento para el reconocimiento de patrones, inicialmente el sistema debe permitir crear un nuevo experimento, para el cual el usuario deberá registrar un nombre, seleccionar una dirección para almacenarlo, un dominio de datos que contendrá los vectores de características a agrupar, la medida de distancia que permitirá definir cuan cercanos o no son eso vectores, el algoritmo de agrupamiento que finalmente agrupa a los vectores y la medida de calidad que evaluará el acoplamiento de los datos resultantes. El usuario también podrá introducir una descripción a cerca del experimento realizado y el sistema genera un resumen con las propiedades más significativas de dicho experimento.

Luego de creado el experimento se dará la posibilidad de visualizar toda su información o la de otro ya existente realizando una búsqueda. Entre la información que será mostrada se encuentran: los datos del experimento y su descripción, el resultado del agrupamiento realizado, una gráfica que no solo permite visualizar el resultado sino que brinda la opción de cambiar las perspectivas de visualización del agrupamiento permitiendo mostrar gráficamente las características más influyentes y las menos influyentes que posibilitaron que ese vector perteneciera al grupo en el que se encuentra y la evaluación de la medida de calidad que garantiza la usabilidad de los resultados para el usuario.

Otras de las funcionalidades con la que cuenta el sistema es la de comparar el resultado de dos experimentos en cuanto a una serie de criterios altamente configurables, mostrándose las similitudes y las diferencias entre estos. El usuario tendrá la posibilidad de adicionar al sistema diferentes plug-ins (así como también eliminarlos), quienes serán los encargados de calcular las distancias entre los vectores y de ejecutar el agrupamiento sobre un dominio de datos determinados. Para adicionar los plug-ins la aplicación brinda la posibilidad de realizar la nueva inserción de dos maneras: una a través del gestionar y otra más directa y rápida a través del instalar plug-ins.

A su vez el sistema contiene a disposición del usuario una ayuda para los que presenten dificultades en la utilización de dicho sistema.

2.3. Descripción de la arquitectura.

La Arquitectura de un Sistema (AS), en términos generales, es la organización estructural del mismo, representada por sus componentes, las relaciones entre ellos, el ambiente y los principios que gobiernan el diseño y su evolución; es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el

resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (32).

El sistema para la evaluación de algoritmos de agrupamiento, presenta una arquitectura basada en capas: presentación, negocio, persistencia y plug-ins de modo que cada una interactúa solo con las adyacentes inferiores, donde el negocio es el centro de toda la estructura y con la cual interactúan todas las demás.

La capa de presentación se ocupa de la captura de los datos necesarios enviándolos luego a la capa de negocio la cual se encarga de enviarlos a la capa de persistencia si es necesario, de no serlo los procesa y los envía de vuelta a la capa de presentación. La capa de persistencia se ocupa de atender las peticiones de la capa de negocio enviándole los datos que esta solicita. La capa plug-ins atiende las solicitudes de la capa de presentación enviándole todos los datos solicitados por la misma, esta los procesa y muestra un resultado.

Ilustración 1 Arquitectura por capas.

2.4. Fases del proceso de desarrollo.

2.4.1. Exploración.

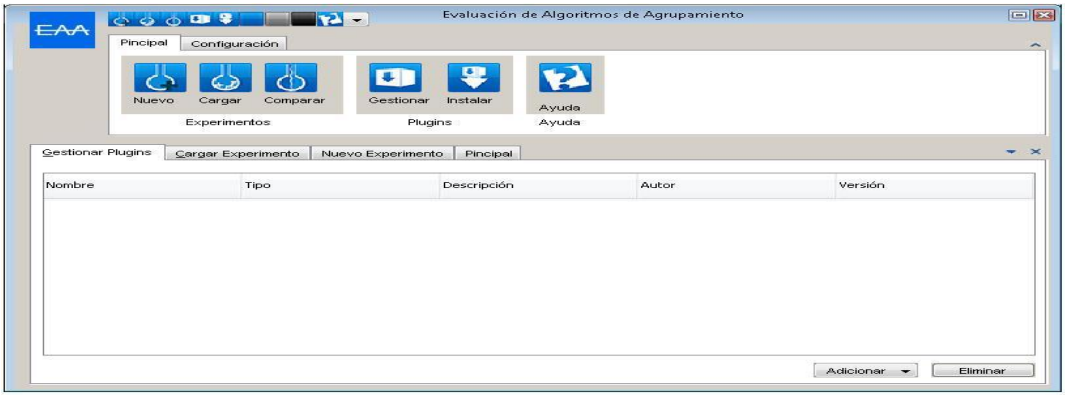
En esta fase los clientes se describen las HU (Historias de Usuario) definiendo las características que van a tener cada una de ellas.

2.4.1.1. Historia de usuario.

Las historias de usuario (HU) son realizadas por el cliente, el cual en su propio lenguaje, describe lo que el sistema debe cumplir. Las mismas deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo.

Cuando llega el momento de la implementación, los desarrolladores se reunirán directamente con el cliente para obtener toda una lista de detalles necesarios para satisfacer sus necesidades (28).

Durante el análisis en la fase de exploración se identificaron un total de cinco HU, a continuación se muestran las que poseen prioridad alta, para el cliente ver las restantes ver Anexo 1.

Historia de Usuario	
Número: 1	**Nombre de Historia:** Gestionar plug-ins.
Usuario: Usuario Común.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta.	**Riesgo en Desarrollo:** Alta.
Puntos Estimados: 2	**Puntos Reales:** 2
Programador Responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	
Descripción: El sistema debe permitir adicionar, listar o eliminar plug-ins. Para adicionar un plug-in se selecciona el tipo de plug-in que se desea añadir, se realiza una búsqueda y se validan los datos del plug-in a adicionar. Para eliminar se selecciona un plug-in de la lista de plug-ins disponibles.	
Observaciones: • El sistema debe poseer un directorio para los algoritmos de agrupamiento y otro para la medida de distancia. • Los nombres de los plug-ins deben ser únicos. • Cuando se elimina un plug-in que se encuentra en uso el sistema debe mostrar un mensaje de información: “El plug-in está en uso. Al cerrarse el sistema será eliminado”.	
Prototipo de Interfaz: 	

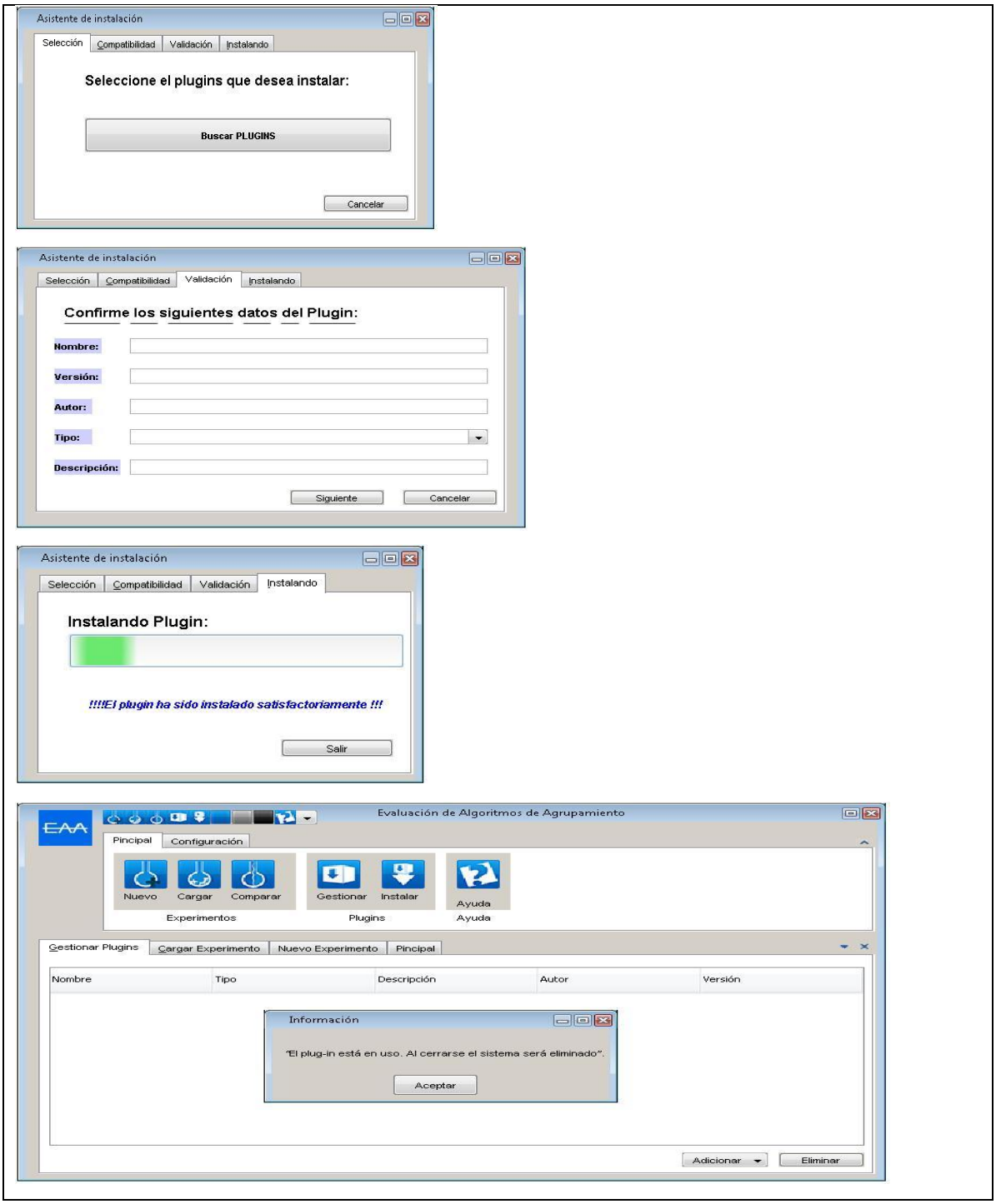


Tabla 1 Descripción HU Gestionar plug-ins.

Historia de Usuario:	
Número: 2	**Nombre de Historia:** Crear experimento.
Usuario: Usuario Común.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta.	**Riesgo en Desarrollo:** Alto.
Puntos Estimados: 3	**Puntos Reales:** 3
Programador Responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	

Descripción: El sistema debe permitir registrar el nombre para el experimento y almacenarlo en una dirección seleccionada por el usuario. Además se debe seleccionar el dominio de datos (el cual contiene objetos con un conjunto de características en forma de vectores) realizando una búsqueda, la medida de distancia (que permite calcular cuan similares pueden ser estos objetos de acuerdo a un criterio de cercanía), el algoritmo de agrupamiento(que escoge a los objetos más próximo formando clases), con una interfaz la cual puede solicitar parámetros si el algoritmo así lo requiere, la o las medidas de calidad para evaluar el acoplamiento de los datos resultantes y se introduce una breve descripción del experimento. Está funcionalidad genera de manera automática un resumen de los datos más relevantes del experimento (nombre del experimento, nombre del dominio de datos, nombre de la medida de distancia, nombre del algoritmos de agrupamiento y el o los nombres de las medidas de calidad).

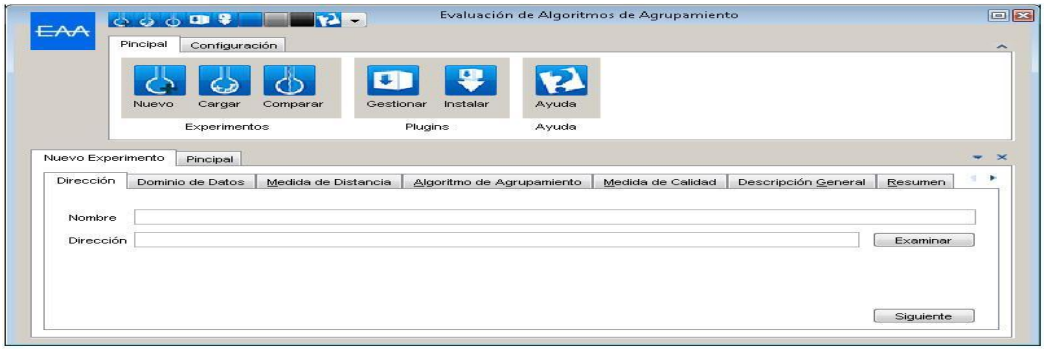
Observaciones:

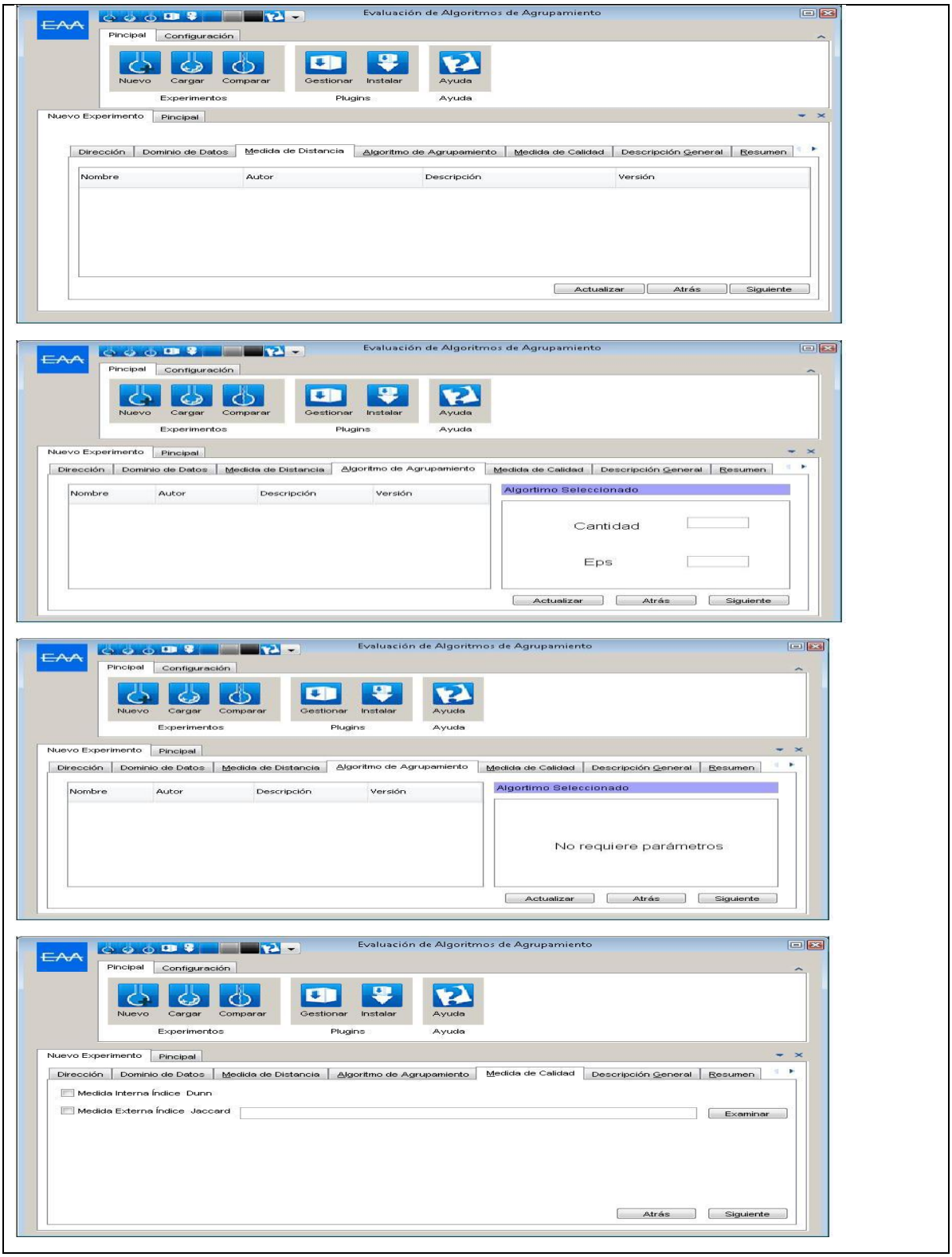
- El sistema debe permitir al usuario seleccionar una ubicación para el experimento.
- Los nombres de los experimentos deben ser únicos.
- Si falta un campo por registrar debe ser lanzado un mensaje de alerta.
- El dominio de datos debe contener cantidad de filas, cantidad de columnas y los datos de la matriz. La cantidad de datos de la matriz debe ser el producto de la cantidad de filas por la cantidad de columnas.
- El dominio de los datos debe tener formato XML.
- Debe existir cargado al menos una medida de distancia y un algoritmo de agrupamiento en el sistema.
- El usuario deberá seleccionar al menos una medida de distancia, de no realizar una selección el sistema debe mostrar un mensaje de alerta.
- El usuario deberá seleccionar al menos un algoritmo de agrupamiento, de no

realizar una selección el sistema debe mostrar un mensaje de alerta.

- El usuario podrá seleccionar al menos una medida de calidad por cada uno de los experimentos realizados, de no realizar una selección el sistema debe mostrar un mensaje de alerta.
- Si el usuario selecciona una medida externa debe introducir un fichero con un agrupamiento manual en formato txt, el grupo debe contener el siguiente formato: $[v_1, v_2, + \cdots +, v_n]$ y al terminar la entrada de los grupos escribir un punto y coma (;) que representará la finalidad de los grupos que conforman el agrupamiento.
- El usuario debe registrar obligatoriamente una descripción del experimento, de no ingresar dicha descripción el sistema deberá mostrar un mensaje de alerta.
- La descripción del experimento debe ser corta y precisa.

Prototipo de Interfaz:




Evaluación de Algoritmos de Agrupamiento
Principal
Configuración
Nuevo
Cargar
Comparar
Experimentos
Gestionar
Instalar
Plugins
Ayuda
Ayuda
Nuevo Experimento
Principal
Dirección
Dominio de Datos
Medida de Distancia
Algoritmo de Agrupamiento
Medida de Calidad
Descripción General
Resumen
Nombre
Autor
Descripción
Versión
Actualizar
Atrás
Siguiente
Algortimo Seleccionado
Cantidad
Eps
No requiere parámetros
Medida Interna Índice Dunn
Medida Externa Índice Jaccard
Examinar

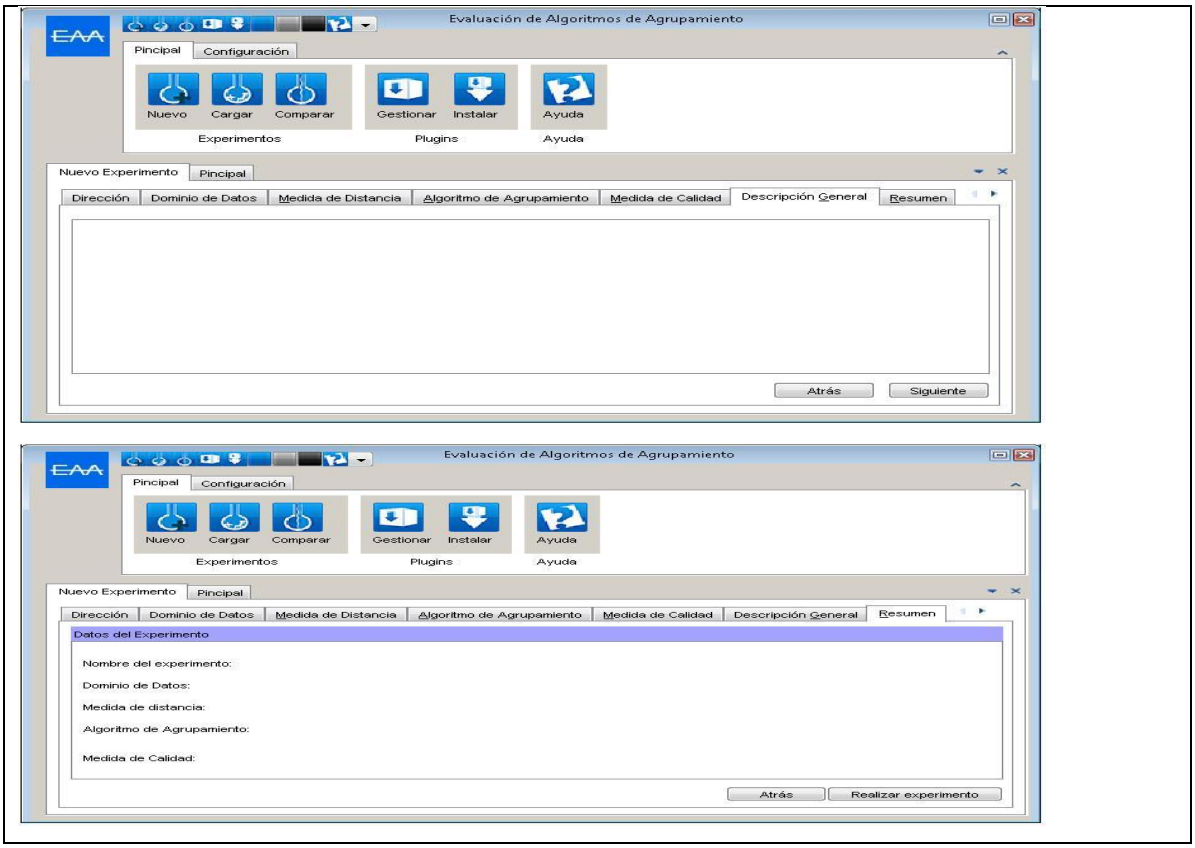


Tabla 2 Descripción HU Crear experimento.

2.4.2. Planificación.

En la fase de planificación se realiza la estimación del esfuerzo que causará la implementación de cada historia de usuario, como en la metodología ágil XP las métricas son libres, puede utilizarse cualquier criterio definido para medir el desempeño del proyecto en cuestión.

2.4.2.1. Estimación de esfuerzo por historias de usuarios.

Para la realización de la aplicación propuesta se efectuó una estimación de esfuerzo por cada una de las historias de usuario identificadas, donde a continuación se muestra los resultados.

Historia de Usuario	Puntos de estimación
Gestionar plug-ins.	2
Crear experimento.	3
Ver experimento.	2
Comparar experimento.	2
Generar ayuda.	2

Tabla 3 Estimación del Esfuerzo por HU.

2.4.3. Iteración.

En esta fase después de ser identificadas las historias de usuario y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la realización de varias iteraciones al sistema antes de ser entregado al cliente.

2.4.3.1. Plan de Iteraciones.

Iteración 1:

Esta iteración tiene como objetivo la implementación de las historias de usuario con prioridad alta. Durante el transcurso de la misma se creará la base de la arquitectura del sistema. Al final de esta se contará con una primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Iteración 2:

El objetivo de esta es la implementación de las historias de usuario de prioridad media. Al finalizar se contará con una versión de prueba con las funcionalidades concernientes a visualizar los datos del experimento y realizar una comparación entre dos experimentos. Esta será mostrada al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

Iteración 3:

Durante el transcurso de esta se implementaron la historia de usuario de prioridad baja. Al finalizar la misma se constará de la versión 1.0 del producto final. Como resultado de esta, el sistema será puesto en funcionamiento durante un período de tiempo para evaluar su desempeño.

2.4.3.2. Plan de duración de las iteraciones.

El plan de duración de las iteraciones es el encargado de mostrar las historias de usuario que serán implementadas en cada una de las iteraciones, así como la duración estimada y el orden de implementación de cada una de ellas.

Iteración	Orden de la Historias de usuario a implementar.	Semanas
1	1. Gestionar plug-ins.	5
	2. Crear experimento.	
2	3. Ver experimento.	4
	4. Comparar experimento.	
3	5. Generar ayuda.	2

Tabla 4 Plan de duración de iteraciones.

2.4.4. Plan de entrega.

El plan de entregas es el compromiso final del equipo de desarrollo con los clientes. Es una cuestión de vital importancia para el negocio entre ambas partes, ya que la entrega tardía de la solución, repercute notablemente de manera negativa en el desarrollo del producto creando insatisfacción en el cliente.

Historias de Usuario.	Iteración 1 Marzo	Iteración 2 Abril	Iteración 3 Mayo
Gestionar plug-ins.	0.1		1.0
Crear experimento.	0.1		1.0
Ver experimento.		0.2	1.0
Comparar experimento.		0.2	1.0
Generar ayuda.			1.0

Tabla 5 Plan de entregas.

2.5. Diseño del sistema.

2.5.1. Patrones de Diseño.

Los patrones de diseño brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular, los mismos son los encargados de identificar Clases, Instancias, Roles, Colaboraciones y la distribución de Responsabilidades (36). El conocimiento de los patrones de diseño es vital para entender el funcionamiento del mismo y por ende lograr los objetivos trazados.

2.5.1.1. Patrones para Asignar Responsabilidades (GRASP[10]).

Los patrones GRASP se consideran que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Son a su vez principios o técnicas de trabajo utilizadas para mejorar los diseños orientados a objetos, como el polimorfismo y la alta cohesión. Para el desarrollo del diseño del sistema se definen los siguientes patrones:

Experto: El patrón experto es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

[10] GRASP: Siglas del inglés: Object-oriented design General Responsibility Assignment Software Patterns

Se evidencia en las clases: ClaseSerializar, ClaseLeerXML y ClaseGuardarDato pues estas cuentan con la información necesaria para cumplir responsabilidades específicas.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador.

Se evidencia en la clase: ClasePeticiones pues tiene la responsabilidad de crear una instancia de otras clases como: ClaseGuardarDato, ClaseLeerXML, ClasePlugin y ClaseSerializar.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Se evidencia en la clase: ClasePeticiones pues es la clase que se encuentra en la capa Intermedia (Capa de Negocio) encargándose de recibir y enviar los datos a cada una de las clases adyacentes con las que se encuentre relacionada.

Bajo acoplamiento: El acoplamiento mide la fuerza con que una clase está conectada a otra, de esta forma una clase con bajo acoplamiento debe tener un número mínimo de dependencia con otras clases. Este patrón se tuvo presente debido a la importancia que se le atribuye a realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez permitan la reutilización, evidenciándose en ClasePeticiones

Alta cohesión: Se aplica para realizar un diseño que evite contener clases con un alto grado de abstracción, que asuman responsabilidades que podían haber delegado a otros objetos o que tengan responsabilidades muy complejas. Se tienen las clases controladoras que se encargan de ejecutar acciones de acuerdo a las peticiones que le llegan y las clases de persistencia que interactúan de forma tal que se elimina la sobrecarga de funcionalidades, evidenciándose en ClasePeticiones.

2.5.2. Tarjetas de Clase, Responsabilidad y Colaboración.

Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC.

Esta técnica se usa para guiar el sistema a través de análisis guiados por la responsabilidad donde las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema. El nombre de la clase será el título en la tarjeta, las responsabilidades a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento

correspondiente (37). Fueron identificadas veintiuna (21) tarjetas CRC, a continuación se muestra una, pueden verse las restantes Anexo 2**.**

2.5.2.1. Tarjeta C.R.C.

Tarjeta CRC	
Clase: PluginInterface	
Descripción: Define las funcionalidades que deben ser implementadas para generar un plug-in.	
Responsabilidades: 1. Obtener y modificar la instancia de la aplicación que cargara el plug-in. 2. Obtener el nombre del plug-in. 3. Obtener la descripción del plug-in. 4. Obtener la versión del plug-in. 5. Obtener el autor del plug-in. 6. Obtener la interfaz de usuario del plug-in en caso que la tenga. 7. Inicializar la instancia del plug-in. 8. Destruir la instancia del plug-in. 9. Calcular matriz. 10. Ejecutar agrupamiento.	**Colaboraciones:** 1. Idisposable.

Tabla 6 Descripción Tarjeta CRC PluginInterface.

2.6. Conclusiones Parciales.

En este capítulo se han descrito las principales características del sistema, en función de realizar una aplicación capacitada para cumplir las expectativas del cliente. A favor de dicho sistema se muestra una propuesta de los procesos que serán objetos de automatización y la planificación que se trazó el equipo de desarrollo, como también se abordó todo lo referente a la fase diseño del proyecto, haciendo una descripción de cada uno de los artefactos generados en esta fase.

CAPÍTULO 3

IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA.

3.1. Introducción.

En este capítulo se hace mención a la fase implementación y prueba, propias de la metodología de desarrollo utilizada. Se exponen además los artefactos generados durante el transcurso de la misma.

3.2. Implementación del sistema.

La metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para retroalimentar a los desarrolladores con la opinión de este. A continuación se detallan las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada historia de usuario (37).

3.2.1. Iteraciones.

3.2.1.1. Iteración 1.

Durante esta iteración se trataron las tareas correspondientes a las historias de usuarios de prioridad alta, con el fin de lograr una rápida retroalimentación con el cliente.

Tareas críticas	Tiempo estimado	Tiempo real
Implementar las funcionalidades que permitan gestionar los plug-ins.	2	2
Implementar las funcionalidades que permitan crear un nuevo experimento.	3	3
Total:	5	5

Tabla 7 Tiempo de las tareas abordadas en la iteración 1.

Tareas abordadas en la Iteración 1:

Tarea de ingeniería.	
Número de la tarea: 1	**Número de historia de usuario:**1
Nombre de la tarea: Implementar las funcionalidades que permitan gestionar los plug-ins.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2

Fecha de inicio: 04/02/2013	**Fecha de fin:** 15/02/2013
Programador responsable: Yordany Bécquer del Toro y Yadira Ramón Galán	
Descripción: Se crea un formulario para adicionar, listar y eliminar los plug-ins. Para adicionar un plug-ins se selecciona el tipo de plug-in que se desea añadir, se realiza una búsqueda, se validan los datos del plug-in a adicionar y se listan luego los plug-ins adicionados. Para eliminar un plug-ins se selecciona uno del listado de plug-ins disponibles, de encontrarse en uso el plug-in que se desea eliminar el sistema muestra un mensaje de información, de encontrarse ya eliminado el sistema muestra un mensaje de error.	

Tabla 8 Descripción de la tarea de la HU Gestionar plug-ins.

Tarea de ingeniería.	
Número de la tarea: 2	**Número de historia de usuario:** 2
Nombre de la tarea: Implementar las funcionalidades que permitan crear un nuevo experimento.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 3
Fecha de inicio: 18/02/2013	**Fecha de fin:** 08/03/2013
Programador responsable: Yordany Bécquer del Toro y Yadira Ramón Galán	
Descripción: Se crea un formulario que contiene un Tabcontrol con pestañas que serán llenadas de manera consecuente, si alguna de ellas se encuentra incompleta o posee datos incorrectos no se pasará a la siguiente. En cada una de las pestañas se realizarán las siguientes funcionalidades: Se introduce un nombre para el experimento y se selecciona una dirección donde pueda almacenado. Se seleccionan además el dominio de datos realizando una búsqueda, la medida de distancia, el algoritmo de agrupamiento que muestra una interfaz la cual puede solicitar parámetros o no, la o las medidas de calidad y se introduce una breve descripción del experimento. Por último se genera de manera automática un resumen de los datos más relevantes del experimento para ser mostrado en la opción: “Cargar”, de la pestaña Principal.	

Tabla 9 Descripción de la tarea de la HU Crear experimento.

3.2.1.2. Iteración 2.

Durante esta iteración se trataron las tareas correspondientes a las historias de usuarios de prioridad media, esta será mostrada al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

Tareas Medias	Tiempo estimado	Tiempo real
Implementar las funcionalidades que permitan mostrar los datos de un experimento realizado.	2	2
Implementar las funcionalidades que permitan comparar dos experimentos en función de un determinado grupo de criterios.	2	2
Total:	4	4

Tabla 10 Tiempo de las tareas abordadas en la iteración 2.

Para un mejor entendimiento acerca de la descripción de las tareas de implementación para esta iteración ver Anexo 3.

3.2.1.3. Iteración 3.

Durante esta iteración se trataron las tareas correspondientes a las historias de usuarios de prioridad baja. Al culminar esta, se consta de un producto listo para su puesta en funcionamiento.

Tareas Bajas	Tiempo estimado	Tiempo real
Implementar las funcionalidades que permitan generar la ayuda del sistema.	2	2
Total:	2	2

Tabla 11 Tiempo de las tareas abordadas en la iteración 3.

Para un mejor entendimiento acerca de la descripción de las tareas de implementación para esta iteración ver Anexo 3.

3.3. Manual de Usuario.

Cómo implementar un plug-in:

Para que el usuario realice un experimento, es necesario que el sistema tenga adicionado al menos dos plug-ins: uno que contenga una medida de distancia para calcular cuan semejantes son los vectores de características atendiendo un criterio de cercanía y otro que contenga un algoritmo de agrupamiento que es el que permite finalizar el experimento obteniendo los vectores agrupados. Para implementar un plug-in el usuario deberá crear un proyecto de tipo class Library y añadir como referencia un fichero denominado PluginInterface, este funciona como conexión física y funcional entre el sistema y los algoritmos, dando una comunicación entre ellos. Luego el usuario deberá implementar esta interfaz la cual contiene los siguientes atributos:

1. Host: Especifica la clase que instanció el plug-in.
2. Nombre: Se encarga de obtener el nombre del plug-in.

3. Descripción: Se encarga de obtener una descripción para el plug-in.
4. Autor: Se encarga de obtener el autor del plug-in.
5. Versión: Se encarga de obtener la versión del plug-in.
6. MainInterface: Este atributo se inicializa cuando se implementa un algoritmo de agrupamiento que requiera de una interfaz para el plug-in, de lo contrario se inicializa en null. Se hace necesario para la creación de esta interfaz el uso de un componente denominado UserControl que permite embeberse dentro de otros componentes.

La interfaz a implementar contiene las siguientes funcionalidades:

1. Initialize: Se encarga de inicializar los datos del plug-in.
2. Dispose: Se encarga de destruir los datos del plug-in.
3. Calcular: Se implementa específicamente para el plug-in de medida de distancia y se encarga de calcular la distancia entre cada elemento de la matriz de dominio de datos.
4. Ejecutar: Se implementa específicamente para el plug-in de algoritmo de agrupamiento y se encarga de ejecutar la funcionalidad que realiza el agrupamiento de los vectores de características del experimento.

Se lanza una excepción(throw new NotImplementedException()) para la funcionalidad que no haya sido implementada, es decir cuando el usuario implemente un plug-in para una medida de distancia de lanza una excepción dentro de la funcionalidad “Ejecutar” y cuando el usuario implemente una funcionalidad para un algoritmo de agrupamiento se lanza una excepción dentro de la funcionalidad “Calcular”.

Cómo Incluir un plug-in:

Después de implementado los plug-ins, para hacer uso de los mismos, el usuario puede cargarlo en el sistema de varias maneras distintas, como se explica a continuación:

1. El instalador del programa crea un ejecutable del sistema y dos directorios que serán los encargados de almacenar los plug-ins de medida de distancia y de algoritmo de agrupamiento, con solo adicionar los plug-ins a cada uno de los directorios correspondientes el sistema podrá emplear los nuevos plug-ins para realizar un nuevo experimento.
2. Con solo ejecutar el sistema después de su correcta instalación, oprimir el botón “Gestionar”, seleccionar que tipo de plug-ins desea añadir, realizar una búsqueda para la adición del nuevo fichero y validar que sus datos sean correctos podrá emplear los nuevos plug-ins para realizar un nuevo experimento.
3. El sistema cuenta con otra variante para añadir los plug-ins al sistema muy similar que la anterior pero la que se explica a continuación es mucho más rápida y directa: con solo ejecutar

el sistema oprimir el botón “Instalar” realizar una búsqueda para la adición del nuevo fichero y validar que sus datos sean correctos podrá emplear los nuevos plug-ins para realizar un nuevo experimento.

3.4. Validación.

Las pruebas son una de las prácticas fundamentales en las cuales se basa la metodología XP. Esta actividad se realiza en forma continua a lo largo del proyecto. Mediante estas se reduce el número de errores no detectados así como el tiempo entre la introducción de estos en el sistema y su detección. Esto contribuye a una mejor calidad de los productos desarrollados y la seguridad de los programadores a la hora de implantar cambios y modificaciones. Existen dos tipos de pruebas, las unitarias y las de aceptación (37).

3.4.1. Tipos de pruebas.

Pruebas unitarias:

Una prueba unitaria es la verificación de un módulo (unidad de código) determinado dentro de un sistema. El concepto de “módulo” varía de acuerdo al lenguaje de programación que haya sido utilizado. Las pruebas unitarias aseguran que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema.

Los programadores realizan estas pruebas cuando: la interfaz de un método no es clara, la implementación es complicada, para probar entradas y condiciones inusuales, luego de modificar algo. Éstas deben contemplar cada módulo del sistema que pueda generar fallas. Para poder integrar el código realizado al ya existente, el mismo debe aprobar satisfactoriamente todos los casos de prueba definidos.

En la metodología XP los programadores deben escribir las pruebas unitarias para cada módulo antes de escribir el código. No es necesario escribir casos de prueba para todos los módulos, sólo para aquellos en que exista la posibilidad de que puedan fallar (37).

Pruebas de aceptación:

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en la metodología XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención.

Las pruebas de aceptación permiten al cliente saber cuándo el sistema funciona, y que los programadores conozcan que es lo que resta por hacer.

El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Los clientes son responsables de verificar que los resultados de las pruebas sean correctos. Asimismo, en caso de que fallen, deben indicar el orden de prioridad de resolución.

Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información (37).

3.4.2. Pruebas realizadas al sistema.

La realización de los casos de pruebas por historia de usuario, se llevan a cabo a través del análisis de escenarios (posibles eventos que ocurrirán sobre el sistema), variables (todas los datos que introduce el usuario en el sistema) y realizando una descripción de lo que ocurre en el sistema ante estos eventos, a continuación se presentan los casos de pruebas para las historias de usuario con prioridad alta, si el cliente desea ver los casos de pruebas de las historias restantes ver Anexo 4.

Caso de Prueba:	
Número de Caso de Prueba: 1	**Número de Historia de Usuario:** 1
Nombre del probador: Yadira Ramón Galán y Yordany Bécquer del Toro.	
Nombre del Caso de Prueba: Adicionar plug-ins.	
Descripción: Adicionar un plug-ins para comprobar si es adicionado correctamente.	
Condiciones de ejecución: La aplicación debe estar disponible.	
Entradas: El usuario accede a la opción: “Gestionar”, oprime el botón” Adicionar”, realiza una selección del tipo de plug-in que desea añadir y realiza una búsqueda, valida los datos del plug-in a adicionar, oprime el botón “Siguiente” y el botón “Salir”. El usuario accede a la opción: “Instalar”, realiza una búsqueda, valida los datos del plug-in a adicionar, oprime el botón “Siguiente” y el botón “Salir”.	
Resultado esperado: El sistema verifica que el plug-in seleccionado sea válido y realiza una notificación de la adición.	
Evaluación: Satisfactoria.	

Tabla 12 Descripción del Caso de Prueba Adicionar plug-ins.

Caso de Prueba:	
Número de Caso de Prueba: 2	**Número de Historia de Usuario:** 1
Nombre del probador: Yadira Ramón Galán y Yordany Bécquer del Toro.	
Nombre del Caso de Prueba: Eliminar plug-ins.	
Descripción: Eliminar un plug-ins para comprobar si es eliminado correctamente.	
Condiciones de ejecución: La aplicación debe estar disponible.	
Entradas: El usuario accede a la opción:"Plug-ins", selecciona un plug-ins del listado de plug-ins disponibles, oprime el botón "Eliminar".	
Resultado esperado: El sistema verifica que el plug-in seleccionado sea válido para la eliminación, que no se encuentre en uso por el sistema o que no haya sido eliminado con anterioridad y realiza una notificación después de ejecutada la operación.	
Evaluación: Satisfactoria.	

Tabla 13 Descripción del Caso de Prueba Eliminar plug-ins.

Caso de Prueba:	
Número de Caso de Prueba: 3	**Número de Historia de Usuario:** 2
Nombre del probador: Yadira Ramón Galán y Yordany Bécquer del Toro.	
Nombre del Caso de Prueba: Crear experimento.	
Descripción: Se crea un nuevo experimento con el fin de comprobar que la creación de un experimento se realice correctamente.	
Condiciones de ejecución: La aplicación debe estar disponible.	
Entradas: El usuario accede a la opción: "Nuevo" y aparece un formulario con pestañas que contienen datos a llenar. El usuario introduce el nombre del experimento, oprime el botón "Examinar" para realiza una búsqueda que le permita seleccionar una ubicación para el mismo y oprime el botón "Siguiente". El usuario oprime el botón "Examinar" para realiza una búsqueda que le permita seleccionar un dominio de datos para el nuevo experimento y oprime el botón "Siguiente"; para volver a la pestaña anterior oprime el botón "Atrás". El usuario selecciona una medida de distancia del listado que aparece en la pestaña y oprime el botón "Siguiente", para volver a la pestaña anterior oprime el botón "Atrás", de no aparecer ningún elemento en el listado el usuario deberá acceder a las opciones "Gestionar" ó "Instalar" para adicionar un plug-in y oprimir el botón "Actualizar". El usuario selecciona un algoritmo de agrupamiento del listado que aparece en la pestaña y oprime el botón "Siguiente", para volver a la pestaña anterior oprime el	

botón “Atrás”, de no aparecer ningún elemento en el listado el usuario deberá acceder a las opciones “Gestionar” ó ”Instalar” para adicionar un plug-in y oprimir el botón “Actualizar”. Si el algoritmo de agrupamiento requiere de parámetros, el usuario debe introducir la información solicitada. El usuario selecciona la o las medidas de calidad que evaluarán el resultado arrojado por el agrupamiento y oprime el botón “Siguiente”, de seleccionar la medida de calidad Índice de Jaccard el usuario deberá realizar una búsqueda para cargar un fichero con un agrupamiento manual, para volver a la pestaña anterior oprime el botón “Atrás”. El usuario introduce una breve descripción del experimento y oprime el botón “Siguiente”, para volver a la pestaña anterior oprime el botón “Atrás”. El sistema muestra un resumen con las propiedades del experimento y el usuario oprime el botón “Realizar experimento”, para volver a la pestaña anterior oprime el botón “Atrás”.
Resultado esperado: El sistema verifica que el nombre y la descripción introducida, la ubicación, la medida de distancia, el algoritmo de agrupamiento y la medida de calidad seleccionadas sean válidas para realizar el nuevo experimento.
Evaluación: Satisfactoria.

Tabla 14 Descripción del Caso de Prueba Crear experimento.

3.5. Conclusiones Parciales.

Como parte del presente capítulo se abordó todo lo referente a la fase de implementación del proyecto, haciendo una descripción de cada uno de los artefactos generados durante el transcurso del mismo. Además en este capítulo se hizo mención a los tipos de pruebas propuestas por XP y se efectuó paso a paso la descripción de los casos de pruebas realizados al sistema.

CONCLUSIONES GENERALES.

Cuando surgió la ciencia de la computación, quizás nunca se pensó en el alcance que esta iba a llegar a tener, probablemente solo se pensó en hacer algo simple que permitiera compartir información rápidamente. El surgimiento de esta investigación ha permitido: desarrollar un sistema informático amigable y flexible que facilita la evaluación de algoritmos de agrupamiento para el reconocimiento de patrones, permitiendo la persistencia y comparación de los experimentos realizados, quedando como colofón del mismo:

- El análisis realizado de los procesos para la evaluación de los algoritmos de agrupamiento para el reconocimiento de patrones y los sistemas que existen actualmente así como la especificación de las tecnologías y herramientas que se utilizaron para el desarrollo de la solución.
- La descripción de los aspectos más significativos de la solución y la validación de la solución propuesta mediante el diseño y aplicación de las pruebas de aceptación, permitiendo validar el correcto funcionamiento de las funcionalidades implementadas.

Por todo lo anteriormente expresado se puede concluir que se cumplió con el objetivo general propuesto: Desarrollar una aplicación informática para la evaluación de algoritmos de agrupamiento para reconocimiento de patrones, de manera satisfactoria.

RECOMENDACIONES.

Existen una serie de factores que permitirán la mejora de los procesos para la evaluación de los algoritmos de agrupamiento para el reconocimiento de patrones y que deben tenerse en consideración para versiones posteriores de este producto, entre ellos están:

- Incorporar otras medidas de calidad para la evaluación del agrupamiento.
- Permitir seleccionar a la vez varios experimentos durante la búsqueda que se realiza para seleccionar los experimentos a comparar.
- Comparar más de dos experimento a la vez.

REFERENCIAS BIBLIOGRÁFICAS.

1. **Duda, R.O. and Hart.** Pattern Classification, and Scene Analysis. New York : John Wiley & Sons, 1973.

2. **González, Damaris Pascual.** Algoritmos de Agrupamiento basados en densidad y Validación de clusters. 2010.

3. **Jain, A. K., Murty, M. N. and Flynn, P. J.** Data Clustering: A Review. ACM Computing Surveys, Vol.31, No. 3, 264 - 323 . (1999).

4. **Diego A. Ingaramo, Marcelo L. Errecalde,Paolo Rosso.** Medidas internas y externas en el agrupamiento de resúmenes científicos de dominios reducidos. 2007.

5. **Witten, Ian H. y Frank, Eibe.** «Data Mining: Practical machine learning tools and techniques 2nd Edition». , . San Francisco : Morgan Kaufmann, 2005.

6. Técnicas de Análisis de Datos en WEKA.

7. **Goering, Richard.** "Matlab edges closer to electronic design automation world". 2004.

8. Caja de herramienta de agrupamiento. [En línea] http://mscerts.programming4.us/es/398166.aspx.

9. Tutorial de RapidMiner 5.0. [En línea] http://www.dataprix.com/ejemplo-9-validaci-n-cruzada-num-rica.

10. **Jackman, Simon.** R For the Political Methodologist. 2003.

11. **Kurt Hornik, Achim Zeileis, Torsten Hothorn and Christian Buchta. .** RWeka: An R Interface to Weka. R package version 0.3-17. [En línea] http://cran.r-project.org/web/packages/RWeka/index.html.

12. **Álvarez, Manuel Miguel Ramos.** DESCRIPCIÓN CON ANÁLISIS DE CLUSTER.

13. **Christopher D. Manning, Prabhakar Raghavan & Hinrich Schutze.** Introduction to Information Retrieval.

14. **Estivill-Castro, V.** "Why so many clustering algorithms". 2002.

15. **Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, Arthur Zimek.** "On Using Class-Labels in Evaluation of Clusterings". 2010.

16. **Rand, W. M.** "Objective criteria for the evaluation of clustering methods". 1971.

17. **Mallows, E. B. Fowlkes & C. L.** "A Method for Comparing Two Hierarchical Clusterings", . 1983.

18. **Arabie., L. Hubert et P.** Comparing partitions. J. of Classification. 1985.

19. Glosario. [En línea] http://www.glosarium.com/.

20. The cold sun. [En línea] http://www.thecoldsun.com.

21. **. Rivas, Lornel A., y otros.** Herramientas de Desarrollo de Software: Hacia la construcción de una Ontología. . 2007.

22. Visual Paradigm for UML. [En línea] http://www.freedownloadmanager.org.

23. PROGRAMACIÓN JAVA: Entorno de desarrollo integrado (IDE). [En línea] http://programacion-laura.blogspot.com .

24. [En línea] http://msdn.microsoft.com/netframework/ .

25. **Millán, José Antonio.** Los términos informáticos. [En línea] http://jamillan.com.

26. Orallo., Enrique Hernández. El Lenguaje Unificado de Modelado (UML). [En línea] http://www.acta.es/articulos_mf/26067.pdf..

27. **H. Canós, José y Letelier, Patricio y Penadés, Mª Carmen.** Métodologías Ágiles en el Desarrollo de Software. [En linea]. [En línea] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

28. **Escribano, Gerardo Fernández.** Introducción a Extreme Programming. 2002.

29. **Penadés, Patricio Letelier y Mª Carmen.** Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).

30. **Autores, Colectivo de.** Introducción al proceso de desarrollo de software. Ciudad de la Habana : Universidad de las Ciencias Informáticas : s.n., 2007.

31. **Perera Morales, José Raúl.** Arquitectura de Software para Sistema Gestion de Inventario. Ciudad de la Habana : s.n., 2007.

32. **César de la Torre Lloente, Unai Zorrila Castro, Miguel Angel Ramos Barroso, Javier Calvarro Barroso.** Guía de Arquitectura N-Capas orientada al Dominio con .Net 4.0.

33. **Pressman, Roger S.** Ingeniería del Software. Un enfoque práctico. .

34. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. s.l. : Pearson Educación S.A.

35. **E. Gamma, R. Helm, R. Johnson, and J. Vlissides.** Desing Patterns. . 1995.

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

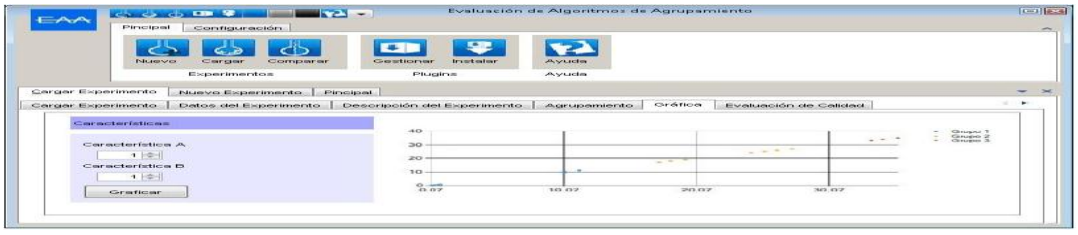
37. **Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler.** Rapid Prototyping for Complex Data Mining Tasks.

38. **Carpi, Elena.** Lenguaje informático y lengua española. Università di Pisa : s.n.

ANEXOS.

Anexo 1 Descripción de Historias de Usuario (HU):

Historia de Usuario	
Número: 3	**Nombre de Historia:** Ver experimento.
Usuario: Usuario Común.	**Iteración Asignada:** 2
Prioridad en Negocio: Media.	**Riesgo en Desarrollo:** Medio.
Puntos Estimados: 2	**Puntos Reales:** 2
Programador Responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	
Descripción: El sistema debe permitir cargar un experimento realizando una búsqueda, mostrar sus datos y la descripción, el agrupamiento realizado, la gráfica del resultado generado y la evaluación de la medida de calidad arrojada.	
Observaciones: • Para ver un experimento debe haberse creado alguno con anterioridad. • De no existir un experimento creado debe lanzarse un mensaje de alerta. • La grafica debe permitir cambiar la perspectiva de visualización del experimento.	
Prototipo de Interfaz:  	



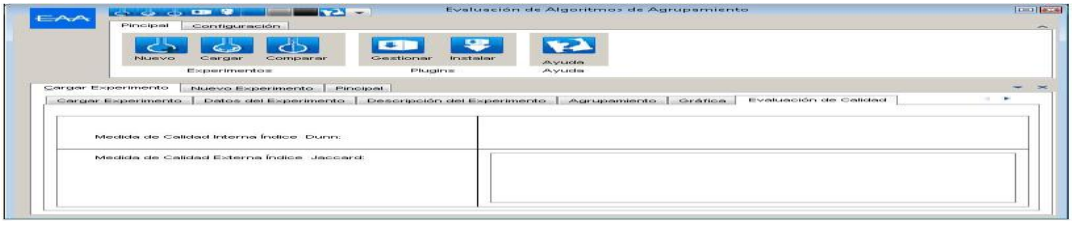


Tabla 15 Descripción HU Ver experimento.

Historia de Usuario	
Número: 4	**Nombre de Historia:** Comparar experimento.
Usuario: Usuario Común.	**Iteración Asignada:** 2
Prioridad en Negocio: Media.	**Riesgo en Desarrollo:** Medio.
Puntos Estimados: 2	**Puntos Reales:** 2
Programador Responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	

Descripción: El sistema permite cargar varios experimentos para luego incluir solo dos en el listado de experimentos a comparar y realizar una selección de los criterios de comparación. Además podrá excluir un experimento incluido o excluir todos los experimentos que una vez fueron incluidos. A su vez si el usuario así lo desea podrá almacenar el resultado de la comparación.

Observaciones:

- El sistema debe tener creado al menos dos experimentos con anterioridad.
- De no existir un experimento creado debe lanzarse un mensaje de alerta.
- Cuando se ha cargado un experimento a comparar debe habilitarse el botón: “Incluir”.
- Cuando se ha incluido un experimento se habilitan los botones: “Excluir” y “Excluir todos”.
- Cuando se han incluido dos experimentos se inhabilita el botón: “Incluir”.
- Cuando se han incluido dos experimentos se habilita el botón: “Siguiente”.
- Si se excluye un experimento y queda otro incluido se inhabilita el botón: “Siguiente”.
- Si se excluye un experimento y queda otro incluido se habilita el botón: “Incluir”.
- Si se excluye un experimento y no queda otro por ser excluido se inhabilita los botones: “Excluir” y “Excluir todos”.
- Al excluirse todos los experimentos se inhabilitan los botones: “Excluir” y “Excluir todos”.
- Al excluirse todos los experimentos se habilita el botón: “Incluir”.
- Para realizar la comparación el usuario debe seleccionar al menos tres criterios de comparación, de no ser así el sistema debe mostrar un mensaje de alerta.
- Guardar la comparación debe ser opcional para el usuario.
- Si el usuario no introduce donde desea guardar el experimento, debe mostrarse un mensaje de alerta.
- Si dos experimentos a comparar no están evaluados por la misma medida de calidad el sistema debe mostrar un mensaje y no se mostrarán los datos de las evaluaciones arrojadas por cada una de las medidas.

Prototipo de Interfaz:

Tabla 16 Descripción HU Comparar experimento.

Historia de Usuario	
Número: 5	**Nombre de Historia:** Generar ayuda.
Usuario: Usuario Común.	**Iteración Asignada:** 3
Prioridad en Negocio: Baja.	**Riesgo en Desarrollo:** Bajo.
Puntos Estimados: 2	**Puntos Reales:** 2
Programador Responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	
Descripción: El sistema debe mostrar una ayuda como guía para todos los usuarios que deseen utilizar la aplicación.	
Observaciones:	
Prototipo de Interfaz: No aplica.	

Tabla 17 Descripción HU Generar ayuda.

Anexo 2 Tarjetas de clase, responsabilidad y colaboración (CRC):

Tarjeta CRC	
Clase: ClaseAgrupamiento.	
Descripción: Se encarga de guardar todos los datos del agrupamiento.	
Responsabilidades: 1. Obtener y modificar el total de vectores del agrupamiento. 2. Obtener y modificar el total de vectores que fueron agrupados. 3. Obtener y modificar los outlieres. 4. Obtener y modificar el porciento de vectores agrupados. 5. Obtener y modificar el porciento de los outliers. 6. Obtener y modificar la cantidad de grupos que fueron creados. 7. Obtener y modificar el listado con la posición de los vectores por grupos. 8. Obtener y modificar el listado de vectores por grupo. 9. Obtener y modificar la calidad del agrupamiento interno. 10. Obtener y modificar la calidad del agrupamiento externo.	**Colaboraciones:**

Tabla 18 Descripción Tarjeta CRC ClaseAgrupamiento.

Tarjeta CRC	
Clase: ClaseDatosExperimento.	
Descripción: Clase serializable, que se encarga de guardar todos los datos del experimento.	
Responsabilidades:	**Colaboraciones:**

1. Obtener y modificar el nombre del experimento. 2. Obtener y modificar el dominio de datos. 3. Obtener y modificar la medida de distancia utilizada para el experimento. 4. Obtener y modificar el algoritmo de agrupamiento. 5. Obtener y modificar la descripción del experimento. 6. Obtener y modificar la fecha de creación de los experimentos. 7. Obtener y modificar la lista con las filas de la matriz de dominio de datos. 8. Obtener y modificar el listado del resultado de ejecución del algoritmo de agrupamiento. 9. Obtener y modificar el agrupamiento. 10. Obtener y modificar la dirección del experimento. 11. Obtener y modificar la medida de calidad. 12. Obtener y modificar el tiempo de ejecución del experimento.	

Tabla 19 Descripción Tarjeta CRC ClaseDatosExperimento.

Tarjeta CRC
Clase: ClaseDatosGraficar.
Descripción: Grafica los vectores de características agrupados formando puntos (x,y) por cada vector que pertenece al grupo.

Responsabilidades:	**Colaboraciones:**
1. Listar todos los grupos resultante del agrupamiento con las características de cada vector que forma parte de un grupo. 2. Listar todos los grupos del agrupamiento con sus vectores correspondientes. 3. Determinar cuál es la cantidad de vectores agrupados. 4. Listar todas las características a graficar de un grupo formando puntos (x,y) por cada vector que pertenece al grupo. 5. Listar los grupos que contienen la posición del vector que le pertenece.	

Tabla 20 Descripción Tarjeta CRC ClaseDatosGraficar.

Tarjeta CRC	
Clase: ClaseMedidaDeCalidad.	
Descripción: Evalúa la calidad del agrupamiento a partir de un tipo de validación de cluster en específico	
Responsabilidades:	**Colaboraciones:**
1. Calcular la medida de calidad a partir del índice Dunn sobre el resultado de un agrupamiento. 2. Calcular el método Centroide para cada distancia intragrupal. 3. Calcular distancia entre grupos. 4. Calcular la medida de calidad a partir del índice Jaccard sobre el resultado de un agrupamiento. 5. Calcular la función Jaccard entre dos grupos.	

Tabla 21 Descripción Tarjeta CRC ClaseMedidaDeCalidad.

Tarjeta CRC	
Clase: ClaseGenerarMatriz.	
Descripción: Crea la matriz de distancia.	
Responsabilidades: 1. Devolver una lista en donde cada posición de la misma es una fila de la matriz de dominio de datos. 2. Generar la matriz de distancia.	**Colaboraciones:**

Tabla 22 Descripción Tarjeta CRC ClaseNuevoExperimento.

Tarjeta CRC	
Clase: ClasePeticiones.	
Descripción: Clase mediadora, para acceder a las funcionalidades de las clases creando objetos de cada una de ellas.	
Responsabilidades: 1. Validar Dominio de datos. 2. Leer XML. 3. Guardar en Carpeta. 4. Copiar plug-in en carpeta. 5. Eliminar plug-in. 6. Serializar. 7. Deserializar. 8. Guardar comparación. 9. Leer fichero de Jaccard.	**Colaboraciones:**

Tabla 23 Descripción Tarjeta CRC ClasePeticiones.

Tarjeta CRC	
Clase: ClasePunto.	
Descripción: Se encarga de guardar los datos para formar los puntos que permiten graficar el agrupamiento.	
Responsabilidades:	**Colaboraciones:**

1. Obtener y modificar los datos de un punto (x, y).	

Tabla 24 Descripción Tarjeta CRC ClasePunto.

Tarjeta CRC	
Clase: ClaseTemporizador.	
Descripción: Calcular el rendimiento del sistema para un experimento.	
Responsabilidades: 1. Almacenar la frecuencia de alto rendimiento. 2. Almacenar el valor del conteo inicial. 3. Almacenar el valor del conteo final. 4. Indicar si ya se ha inicializado el timer. 5. Iniciar el conteo de temporizador. 6. Detener el conteo del temporizador. 7. Retornar la cantidad de nanosegundos contados. 8. Retornar la cantidad de milisegundos contados. 9. Retornar la cantidad de segundo contados.	**Colaboraciones:**

Tabla 25 Descripción Tarjeta CRC ClaseTemporizador.

Tarjeta CRC	
Clase: TipoPlugins.	
Descripción: Clase enumerativa para los plug-ins.	
Responsabilidades: 1. Obtener los tipos de plug-ins para el sistema (PluginsAlgoritmoAgrupamiento, PluginsMedidaDistancia).	**Colaboraciones:**

Tabla 26 Descripción Tarjeta CRC TipoPlug-ins.

Tarjeta CRC	
Clase: ClaseGuardarDatos.	
Descripción: Guarda los datos del experimento en la carpeta creada por el usuario.	
Responsabilidades: 1. Copiar el XML con el domino de datos cargado en la carpeta creada por el usuario 2. Crear un fichero para la descripción del experimento y se adiciona en él dicha descripción. 3. Crear un fichero para la matriz de distancia (plug-ins 1) y se adiciona en él dicha matriz. 4. Crear un fichero para los algoritmos de agrupamiento (plug-ins 2) y se adiciona en él dicho algoritmo. 5. Crear un fichero para almacenar las propiedades del experimento. 6. Crear un fichero para guardar la comparación realizada entre dos experimentos.	**Colaboraciones:**

Tabla 27 Descripción Tarjeta CRC ClaseGuardarDatos.

Tarjeta CRC	
Clase: ClaseFicheroDeJaccard.	
Descripción: Valida si el fichero introducido por el usuario es correcto.	
Responsabilidades: 1. Almacena en una lista todas las listas de grupos contenidos dentro del fichero cargado por el usuario. 2. Leer el fichero. 3. Devolver una lista que contiene todas las	**Colaboraciones:**

líneas del fichero de Jaccard donde cada línea representa un grupo. 4. Validar si cada elemento de la lista pasada por parámetro es un grupo. 5. Validar si el elemento pasado cumple con los requisitos para ser un grupo. 6. Devolver todos los elementos del vector en forma de cadena en una lista eliminando los corchetes"[ ]" y las comillas (" ").	

Tabla 28 Descripción Tarjeta CRC ClaseFicheroDeJaccard

Tarjeta CRC	
Clase: ClaseLeerXML.	
Descripción: Valida la correcta lectura del XML que contiene el dominio de datos para el experimento.	
Responsabilidades: 1. Validar si el dominio de datos cargado es correcto para realizar el experimento. 2. Leer el XML que se encuentra en la dirección dada. 3. Comprobar si la cantidad de datos es correcta. 4. Comprobar si la cantidad de filas es igual a la de columnas.	**Colaboraciones:**

Tabla 29 Descripción Tarjeta CRC ClaseLeerXML.

Tarjeta CRC	
Clase: ClasePlugin.	
Descripción: Se encarga de la inserción y la eliminación de los plug-ins.	
Responsabilidades: 1. Copiar el plug-ins seleccionado en la carpeta correspondiente.	**Colaboraciones:** 1. TipoPlugin.

2. Verificar que la dirección pasada por parámetro no se encuentra en el fichero de los plug-ins eliminados, de no estar el fichero lo crea y adiciona la dirección. 3. Copiar todas las direcciones del directorio pasado por parámetro en una lista. 4. Devolver todas las direcciones del directorio pasado por parámetro en una lista. 5. Elimina el plug-ins seleccionado.	

Tabla 30 Descripción Tarjeta CRC ClasePlugin.

Tarjeta CRC	
Clase: ClaseSerializar.	
Descripción: Codifica un objeto de la clase ClaseDatosExperimento en un archivo de texto para posteriormente mostrar los datos cuando se cargue el experimento en la función Ver Experimento.	
Responsabilidades 1. Serializar la clase pasada por parámetro. 2. Deserializar la clase que se encuentra en una dirección dada.	**Colaboraciones:**

Tabla 31 Descripción Tarjeta CRC ClaseSerializar.

Tarjeta CRC	
Clase: Global.	
Descripción: Se encarga de crear una instancia de los servicios plug-ins.	
Responsabilidades: 1. Crea instancias de ServicioPluginsMedidaDistancia y de ServicioPluginAlgoritmos.	**Colaboraciones:**

Tabla 32 Descripción Tarjeta CRC Global.

Tarjeta CRC	
Clase: PluginDisponible.	
Descripción: Clase de datos para los plug-ins disponibles. Guarda e instancia el plug-in cargado así como la dirección de ensamblado del plug-in.	
Responsabilidades: 1. Obtener y modificar la dirección de ensamblado.	**Colaboraciones:**

Tabla 33 Descripción Tarjeta CRC PluginDisponible

Tarjeta CRC	
Clase: PluginsDisponibles.	
Descripción: Gestionar la colección de plug-ins disponibles.	
Responsabilidades: 1. Adicionar un plug-in a la dirección de plug-ins disponibles. 2. Elimina un plug-in de la dirección de plug-ins disponibles. 3. Buscar un plug-in de la dirección de plug-ins disponibles.	**Colaboraciones:**

Tabla 34 Descripción Tarjeta CRC PluginsDisponibles

Tarjeta CRC	
Clase: ServiciosPlugins.	
1. **Descripción:** Clase padre para listar una colección de todos los plug-ins encontrados cargados por el método BuscarPlugins	
Responsabilidades: 1. Obtener y modificar el listado de plug-ins disponibles. 2. Buscar plug-ins. 3. Buscar plug-ins mediante un directorio.	**Colaboraciones:**

4. Adicionar plug-ins. 5. Cierra todas las instancias de los plug-ins. 6. Adicionar un plug-in a la lista de plug-ins disponibles.	

Tabla 35 Descripción Tarjeta CRC ServiciosPlugins.

Tarjeta CRC	
Clase: ServicioPluginsAlgoritmos.	
Descripción: Clase hija que gestiona las funcionalidades para los plug-ins disponibles de algoritmos de agrupamiento.	
Responsabilidades: 1. Buscar plug-ins.	**Colaboraciones:** 1. ServiciosPlugins.

Tabla 36 Descripción Tarjeta CRC ServicioplugínsAlgoritmos.

Tarjeta CRC	
Clase: ServicioPluginsMedidaDistancia.	
Descripción: Clase hija que gestiona las funcionalidades relacionadas con los plug-ins disponibles para las medidas de distancias.	
Responsabilidades: 1. Buscar plug-ins.	**Colaboraciones:** 1. ServiciosPlugins

Tabla 37 Descripción Tarjeta CRC ServicioPluginsMedidaDistancia.

Anexo 3 Descripción de las tareas abordadas en la iteración 3.

Tarea de ingeniería.	
Número de la tarea: 3	**Número de historia de usuario:** 3
Nombre de la tarea: Implementar las funcionalidades que permitan ver los datos de un experimento realizado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2
Fecha de inicio: 18/03/2013	**Fecha de fin:** 29/03/2013
Programador responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	
Descripción: Se crea un formulario que contiene un Tabcontrol con pestañas que mostrarán de manera consecuente los datos del experimento realizado. Si se desea mostrar los datos de otro experimento se realiza una búsqueda.	

Tabla 38 Descripción de la tarea de la HU Ver experimento.

Tarea de ingeniería.	
Número de la tarea: 4	**Número de historia de usuario:** 4
Nombre de la tarea: Implementar las funcionalidades que permitan comparar dos experimentos en función de un determinado grupo de criterios.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2
Fecha de inicio: 01/04/2013	**Fecha de fin:** 12/04/2013
Programador responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	
Descripción: Se crea un formulario que contiene un Tabcontrol con pestañas que serán llenadas de manera consecuente, si alguna de ellas se encuentra incompleta o posee datos incorrectos no se pasará a la siguiente. En cada una de las pestañas se realizarán las siguientes funcionalidades: cargar varios experimentos para luego incluir solo dos en el listado de experimentos a comparar, realizar una selección de los criterios de comparación, excluir un experimento incluido o excluir todos los experimentos que una vez fueron incluido y si el usuario así lo desea podrá almacenar el resultado de la comparación.	

Tabla 39 Descripción de la tarea de la HU Comparar experimento.

Tarea de ingeniería.	
Número de la tarea: 5	**Número de historia de usuario:** 5
Nombre de la tarea: Implementar las funcionalidades que permitan mostrar la ayuda del sistema.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2
Fecha de inicio: 22/04/2013	**Fecha de fin:** 03/05/2013
Programador responsable: Yordany Bécquer del Toro y Yadira Ramón Galán.	
Descripción: Mostrar ayuda del sistema.	

Tabla 40 Descripción de la tarea de la HU Generar ayuda.

Anexos 4 Descripción de las Pruebas de Aceptación.

Caso de Prueba:	
Número de Caso de Prueba: 4	**Número de Historia de Usuario:** 3
Nombre del probador: Yadira Ramón Galán y Yordany Bécquer del Toro.	
Nombre del Caso de Prueba: Ver experimento.	
Descripción: Visualizar un experimento realizado con el fin de comprobar que todos los datos de un experimentos son mostrados correctamente.	
Condiciones de ejecución: La aplicación debe estar disponible.	
Entradas: El usuario accede a la opción: “Cargar” y aparece un formulario con pestañas con los datos del experimento, el usuario oprime el botón “Examinar” para realiza una búsqueda que le permita seleccionar un experimento ya realizado y oprime el botón “Siguiente”. El sistema muestra los datos del experimento, una descripción del experimento, el agrupamiento realizado, una gráfica con el resultado y la evaluación arrojada por la medida de calidad.	
Resultado esperado: El sistema muestra los datos del experimento realizado.	
Evaluación: Satisfactoria.	

Tabla 41 Descripción del Caso de Prueba Ver experimento.

Caso de Prueba:	
Número de Caso de Prueba: 5	**Número de Historia de Usuario:** 4
Nombre del probador: Yadira Ramón Galán y Yordany Bécquer del Toro.	
Nombre del Caso de Prueba: Comparar experimento.	
Descripción: Comparar dos experimentos realizado con el fin de comprobar que se realiza un comparación correctamente.	
Condiciones de ejecución: La aplicación debe estar disponible.	
Entradas: El usuario accede a la opción: “Comparar experimento” y aparece un formulario con pestañas. El usuario oprime el botón “Examinar” para realiza una búsqueda que le permita cargar los experimentos en el sistema. El usuario oprime el botón “Incluir”, esta operación debe ser realizada al a menos dos veces para oprimir el botón “Siguiente” y realizar una selección de los criterios a comparar. Si el usuario desea eliminar un experimento incluido oprime el botón “Excluir”, si desea eliminarlos todos oprime el botón “Excluir todos”. Después de seleccionar los criterios de comparación, el usuario marca la opción	

"Sí" si desea guardar la comparación y oprime el botón "Siguiente".
Resultado esperado: El sistema muestra una comparación de dos experimentos.
Evaluación: Satisfactoria.

Tabla 42 Descripción del Caso de Prueba Comparar experimento.

Caso de Prueba:	
Número de Caso de Prueba: 6	**Número de Historia de Usuario:** 5
Nombre del probador: Yadira Ramón Galán y Yordany Bécquer del Toro.	
Nombre del Caso de Prueba: Generar ayuda.	
Descripción: Generar la ayuda para comprobar si se esta se muestra correctamente.	
Condiciones de ejecución: La aplicación debe estar disponible.	
Entradas: El usuario accede a la opción: "Ayuda". El sistema muestra una ayuda que funciona como guía para todos los usuarios que deseen utilizar la aplicación.	
Resultado esperado: El sistema muestra una ayuda.	
Evaluación: Satisfactoria.	

Tabla 43 Descripción del Caso de Prueba Generar ayuda.