Centro de Informatización Universitaria

Facultad 1

Universidad de las Ciencias Informáticas

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

Desarrollo de la técnica matemática escalamiento multidimensional para la evaluación del agrupamiento de tarjetas en el paquete Abad.

Autor: Orelsys Pérez Pérez

Tutor: Ing. Ingrid Tobio Pérez
Ing. Ivis Cañizares Rivera

Consultante: Ing. Yanicet Aveleira Rodríguez

La Habana, Junio del 2012

Declaración de autoría

Declaro que soy el único autor del trabajo titulado Desarrollo de la técnica matemática escalamiento multidimensional para la evaluación del agrupamiento de tarjetas en el paquete Abad y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Orelsys Pérez Pérez

Firma del Autor

_____ _____

Ing. Ingrid Tobio Pérez

Firma del Tutor

Ing. Ivis Cañizares Rivera

Firma del Tutor

Agradecimientos

A mi mamá, a mi papá, a mi hermanita y a toda mi familia en general por siempre estar junto a mí, aconsejarme y darme todo el amor y el cariño que me han dado.

A todos mis amigos y a todas las personas que siempre me ayudaron durante el transcurso de la carrera.

Y a todas las personas que me ayudaron, me aconsejaron y me dieron su apoyo durante la realización de este trabajo.

Dedicatoria

Este trabajo se lo dedico a las personas más importantes de mi vida, mi mamá, mi papá y mi hermanita, así como a mis primos, abuelas y abuelos.

También quisiera dedicárselo a mi bisabuelo Pepe, el que siempre fue un ejemplo a seguir en la familia, una persona de carácter fuerte, muy respetado y querido por todos.

Resumen

El presente trabajo tiene como propósito describir el desarrollo de un módulo para el proceso de evaluación mediante la técnica matemática escalamiento multidimensional (EMD) para la herramienta de agrupamiento de tarjetas en el paquete Abad. La herramienta se integró al *software* estadístico R para utilizar los algoritmos necesarios para realizar el EMD. El módulo brinda la posibilidad de representar los resultados en una gráfica de dos dimensiones los cuales muestran todas las relaciones entre los puntos contenidos en ella, facilitando así la interpretación de los resultados; también permite generar un informe en formato PDF con dicha gráfica y su leyenda. Para su desarrollo se utilizaron tecnologías libres cumpliendo con las políticas de migración tecnológica del país.

Palabras clave: agrupamiento de tarjetas, escalamiento multidimensional

Índice de contenido

Introducción **1**

Capítulo 1. Fundamentación teórica **5**

1.1 Arquitectura de la información **5**

1.2 Agrupamiento de tarjetas **5**

1.2.1 Análisis de las técnicas de evaluación 6

Análisis de *Cluster* 6

Pathfinder (PFNets) 6

Redes Neuronales Artificiales 7

Escalamiento multidimensional 7

Valoración de las técnicas de evaluación 7

1.3 Escalamiento multidimensional **8**

1.3.1 Modelo general de escalamiento multidimensional 8

1.3.2 Tipos de escalamiento multidimensional 10

1.4 Escalamiento multidimensional no métrico **10**

1.6 Análisis de soluciones existentes para el agrupamiento de tarjetas **11**

1.6.1 Herramientas de agrupamiento de tarjetas 11

Valoración de las herramientas de agrupamiento de tarjetas 12

1.7 Principales software estadísticos **12**

Statical Package for the Social Sciences (SPSS) 12

SAS/STAT 13

Software estadístico R 13

Comparación entre SPSS, SAS y R 13

Valoración de los programas estadísticos 15

1.6 Metodologías, herramientas y lenguajes **15**

1.6.1 Herramientas de modelado 18

Visual Paradigm 8.0 18

Evolus Pencil 1.3.4 19

Lenguaje unificado de modelado (UML) 2.1 19

1.6.2 Lenguaje de programación. 20

Java 20

1.6.3 Entorno integrado de desarrollo (IDE) 21

Netbeans 7.0 21

1.6.4 *Software* estadístico R 21

Paquete rJava 22

Librería JRI 22

Función isoMDS 23

Conclusiones parciales **23**

Capítulo 2. Propuesta de solución **24**

2.1 Objeto de automatización **24**

2.2 Propuesta del sistema **24**

2.3 El contexto del sistema. Modelo de dominio **25**

Índice de contenido

2.5 Especificación de los requisitos funcionales y no funcionales 27
2.5.1 Requisitos funcionales 27
2.5.2 Requisitos no funcionales 32
2.6 Estándares de codificación 34
2.7 Arquitectura y patrones de diseño 35
2.7.1 Patrones de diseño 35
2.7.2 Patrones GRASP 35
2.7.3 Descripción de la arquitectura 36
Conclusiones parciales 42
Capítulo 3. Validación de la solución 44
3.1 Validación de los requisitos 44
3.1.1 Métrica para evaluar los requisitos 44
Especificidad 44
Compleción 46
3.2 Pruebas 48
3.2.1 Objetivos de las pruebas de software 48
3.2.2 Métodos de pruebas 48
3.2.3 Prueba de integración 49
3.2.4 Prueba de aceptación 50
Resultados de las pruebas de aceptación 53
Conclusiones parciales: 53
Conclusiones Generales 55
Recomendaciones 56
Bibliografía referenciada 57
Bibliografía consultada 61
Anexos 63

Índice de tablas

Tabla 1. Interpretación de STRESS1 9

Tabla 2. Comparación de aspectos generales entre los programas estadísticos SPSS, SAS y R 14

Tabla 3. Especificación de requisitos Seleccionar algoritmo de evaluación 27

Tabla 4. Especificación de requisitos Obtener la matriz de coocurrencia 28

Tabla 5. Especificación de requisitos Obtener la matriz de distancias euclidianas 28

Tabla 6. Especificación de requisitos Obtener la matriz de coordenadas 28

Tabla 7. Especificación de requisitos Obtener una imagen de la gráfica de soluciones 29

Tabla 8. Especificación de requisitos Mostrar gráficamente las soluciones 29

Tabla 9. Especificación de requisitos Mostrar una leyenda de los puntos mostrados en la gráfica 30

Tabla 10. Especificación de requisitos Generar un documento en formato pdf la gráfica y su leyenda 31

Tabla 11. Requisitos no funcionales 32

Tabla 12. Ejemplo de los patrones 36

Tabla 13. Especialistas del proyecto 45

Tabla 14. Prueba de Integración: Software estadístico R 49

Tabla 15. Prueba de Integración: Escalamiento multidimensional no métrico 50

Tabla 16. Caso de prueba: Evaluar 51

Tabla 17. Caso de prueba: Actualizar resultados 52

Tabla 18. Caso de prueba: Exportar PDF 52

Índice de figuras

Figura 1. Algoritmo para el EMD-NM de Shepard-Kruskal 11

Figura 2. Propuesta de solución: Integración SORTAS - R 25

Figura 3. Diagrama del modelo de dominio 26

Figura 4. Arquitectura de la herramienta Agrupamiento de tarjetas y Análisis de secuencia 37

Figura 5. Capa presentación: Paquetes 38

Figura 6. Capa lógica del negocio proceso preparación 39

Figura 7. Capa lógica del negocio proceso ejecución 39

Figura 8. Capa lógica del negocio proceso evaluación 40

Figura 9. Capa lógica del negocio. Entidades 40

Figura 10. Capa lógica del negocio. Entrada y salida 41

Figura 11. Capa de acceso a datos 42

Figura 12. Resultado de la métrica para la calidad de la especificación. Especificidad 46

Figura 13. Resultado de la métrica para la calidad de la especificación. Compleción 47

Figura 14. Resultado final de la aplicación de las métricas 47

Figura 15. Resultados de la pruebas de aceptación por iteración 53

Figura 16. Crear tarjeta 63

Figura 17. Crear de categoría 64

Figura 18. Agrupar las tarjetas en las categorías 65

Figura 19. Seleccionar algoritmos de evaluación 66

Figura 20. Mostrar resultados 67

Figura 21. Informe PDF 68

Introducción

En la actualidad el uso de las tecnologías de la informática y las comunicaciones (TIC) se ha vuelto parte de la vida cotidiana de las personas y de la economía en general. Con el aumento en la interdependencia entre los países y sus ciudadanos, el uso de la Internet se ha convertido en una poderosa herramienta para llevar a cabo diversas actividades administrativas, productivas y comerciales por parte de cualquier empresa, de ahí que los principales cambios tecnológicos se centren en resolver los problemas de *Interacción Persona-Ordenador*[1]con el objetivo de crear productos más usables e interactivos.

Los sistemas interactivos se caracterizan por la importancia del diálogo con las personas donde la interfaz de usuario (IU) representa un papel fundamental en el proceso de desarrollo de cualquier aplicación. La IU es la parte accesible de un sistema computacional, tanto *hardware* como *software*, que permite al usuario interactuar con el sistema. El diseño de buenas IU es parte integral y relevante en el proceso de desarrollo de *software* y en muchos casos ocupa un porcentaje alto del costo de desarrollo de la aplicación(ACOSTA y ZAMBRANO, 2006).

Una mala IU significa una mala *experiencia de usuario*[2], lo que conlleva a que estos dejen de usar la aplicación o se sienta insatisfecho con la misma aunque esta funcione correctamente, pues el usuario no está interesado en la estructura interna de la aplicación, sino en cómo usarla y que la información que él desea obtener sea de fácil acceso, además de que la organización y estructuración de esta, le propicie un máximo de comprensión y asimilación. Uno de los principales objetivos en la construcción de IU es la producción de sistemas usables.

La interfaz de una aplicación es usable cuando las características del diseño y su funcionamiento garantizan su correcta operación, entendimiento y la satisfacción del usuario durante el proceso de interacción con el producto (MONTES DE OCA SÁNCHEZ DE BUSTAMANTE, 2004).La usabilidad de la aplicación depende no solo del diseño de la interfaz, sino también de que la información contenida en ella tenga una buena estructura y organización. La Arquitectura de la Información (AI) es un enfoque de diseño que ha cobrado especial relevancia estos últimos años por esta misma razón (HASSAN *et al.*, 2004).

[1] Es la disciplina relacionada con el diseño, evaluación e implementación de sistemas informáticos interactivos para el uso de seres humanos, y con el estudio de los fenómenos más importantes con los que está relacionado.

[2] Concepto que tiene su origen en el campo del Marketing, y que se puede definir como la sensación, sentimiento, respuesta emocional, valoración y satisfacción del usuario respecto a un producto, resultado del fenómeno de interacción con el producto y la interacción con su proveedor.

Surge como una disciplina para estudiar, analizar y organizar la disposición y estructuración de la información en cualquier medio, de tal forma que sea consultada por el usuario de manera intuitiva (GARRETT, 2002). Su objetivo radica en la creación de sistemas de organización de la información y etiquetados (términos que designan o describen una entidad) que realmente posean un significado para los usuarios pues muchas veces estos tienen que esforzarse durante la realización de actividades tan simples como pueden ser navegar ociosamente en Internet o ejecutar una búsqueda sencilla.

Esta disciplina cuenta con un gran número de técnicas por la cuales se apoyan los *arquitectos de información*[3] para guiar el proceso, las cuales se encuentran resumidas en el marco metodológico que lleva por nombre Diseño Centrado en el Usuario, el cual tiene como objetivo "*lograr la satisfacción de las necesidades de todos sus usuarios potenciales, adaptar la tecnología utilizada a sus expectativas y crear interfaces que faciliten la consecución de sus objetivos. Es un proceso cíclico en el que las decisiones de diseño están dirigidas por el usuario y los objetivos que pretende satisfacer el producto, y donde la usabilidad del diseño es evaluada de forma iterativa y mejorada incrementalmente"(HASSAN-MONTERO y ORTEGA-SANTAMARÍA, 2009)*.

Se relaciona con un heterogéneo conjunto de metodologías y técnicas que comparten un objetivo común: conocer y comprender las necesidades, limitaciones, comportamiento y características del usuario, involucrando en muchos casos a usuarios potenciales o reales en el proceso. Una de las técnicas más populares y eficaces para extraer la estructura semántica del conocimiento que los usuarios tienen sobre un dominio concreto, es el llamado agrupamiento de tarjetas o por sus siglas en inglés *cardsorting*.

El agrupamiento de tarjetas es una de las técnicas más comunes realizadas en la AI por quienes se encargan del diseño conceptual y estructural de un sitio web, lo que pretende esta prueba es establecer patrones o *modelos mentales*[4] de organización de contenidos por parte de los usuarios que realizan este tipo de prueba.

[3] El arquitecto de información es la persona que debe identificar la misión (los objetivos) y la visión (las expectativas de los usuarios) de la página web, determinar los contenidos y funcionalidades de la página, facilitar el acceso mediante sistemas de organización, etiquetado, navegación y búsqueda y planificar en previsión de futuras modificaciones y crecimiento de la página.

[4] Un modelo mental representa el proceso de pensamiento de una persona para saber cómo funciona algo (es decir, la comprensión de una persona del mundo que le rodea). Los modelos mentales se basan en hechos incompletos, experiencias pasadas, y las percepciones, incluso intuitivas. Todo ello ayuda a dar forma a las acciones y el comportamiento, influencia en que la gente preste atención a las situaciones complicadas, y definir cómo la gente se acerca y resolve problemas.

En la Universidad de las Ciencias Informáticas (UCI) específicamente en el Centro de Informatización Universitaria (CENIA) se creó un proyecto para apoyar el desarrollo de los procesos de diseño de *experiencia de usuario*[5], dentro de las iniciativas que proponen tienen desarrollada una herramienta para ejecutar los procesos de preparación, ejecución y evaluación de la técnica agrupamiento de tarjetas, para llevar a cabo el proceso de evaluación a partir de analizar las soluciones de un ejercicio, esta herramienta cuenta con los algoritmos de agrupamiento jerárquicos (en inglés, *hierarchical clustering*), cuya finalidad es agrupar los elementos (o variables) tratando de lograr la máxima homogeneidad en cada grupo y la mayor diferencia entre los grupos. Esta organización jerárquica es representada tradicionalmente por un árbol llamado dendrograma[6]. Estos algoritmos brindan elementos a los profesionales para la toma de decisiones, pero se dificulta la interpretación de los gráficos que generan, pues se pueden interpretar de diversas formas, llegando a una solución inadecuada según el objetivo deseado. Por otro lado, el método no permite que un elemento esté en dos racimos, lo que podría ser conveniente o necesario en algún caso.

Partiendo de la situación expuesta anteriormente surge el siguiente **problema de la investigación:** ¿Cómo analizar y representar los resultados de aplicar la técnica de agrupamiento de tarjetas de forma tal que el arquitecto de información tenga una idea global de los agrupamientos hechos por el usuario, facilitando la interpretación de los mismos?

Se define para la presente investigación como **objeto de estudio**: el análisis matemático para evaluar la aplicación de la técnica de agrupamiento de tarjeta.

El **campo de acción** estará enmarcado en la técnica de escalamiento multidimensional para evaluar la aplicación de la técnica de agrupamiento de tarjeta en la arquitectura de información.

Se ha identificado como **objetivo general** de la investigación: desarrollar un módulo para el paquete Abad que analice y represente los resultados de aplicar la técnica de agrupamiento de tarjetas mediante la técnica matemática escalamiento multidimensional, permitiendo que el arquitecto de

[5] Enfoque multidisciplinario que integra las diferentes disciplinas (usabilidad, arquitectura de información, diseño de interacción, etc.) y roles como un "paraguas" ofreciendo una perspectiva más amplia e integradora, que incluye también el comportamiento emocional del usuario y la importancia de la estética en ese comportamiento.

[6] Es una representación gráfica en forma de árbol que resume el proceso de agrupación en un análisis de *clusters*. Los objetos similares se conectan mediante enlaces cuya posición en el diagrama está determinada por el nivel de similitud/disimilitud entre los objetos.

información tenga una idea global de los grupos de contenidos hechos por el usuario facilitando la interpretación de estos resultados.

De donde se derivan los siguientes **objetivos específicos**:

- Identificar los principios teóricos de la investigación relacionados con el análisis matemático para evaluar la aplicación de la técnica de agrupamiento de tarjeta en la arquitectura de información.
- Definir el módulo técnica matemática escalamiento multidimensional para la evaluación del agrupamiento de tarjetas.
- Desarrollar el módulo técnica matemática escalamiento multidimensional para la evaluación del agrupamiento de tarjetas.
- Aplicar pruebas al módulo técnica matemática escalamiento multidimensional para la evaluación del agrupamiento de tarjetas.

Los métodos científicos teóricos que se emplearon para darle solución a las tareas propuestas fueron:

- Histórico – lógico: permitió realizar el estudio del estado del arte e investigar acerca de otras aplicaciones o soluciones similares, los lenguajes y el proceso de desarrollo de *software*, el marco de trabajo y las herramientas
- Analítico - sintético: se utilizó en el proceso de análisis de la bibliografía consultada, con el objetivo de realizar una síntesis de los elementos relevantes que más aporten a la propuesta de solución.

Los métodos científicos empíricos que se emplearon para darle solución a las tareas propuestas fueron:

- Observación: se utilizó para identificar la situación problemática que da paso al desarrollo de la investigación, así como para realizar una evaluación de los resultados obtenidos con los resultados esperados.

Justificación de la investigación

La presente investigación se realiza para mejorar el proceso de evaluación de la herramienta de agrupamiento de tarjetas. Actualmente esta herramienta utiliza algoritmos de agrupamientos jerárquicos los cuales son muy usados a nivel mundial pero los dendogramas que generan pueden ser difíciles de interpretar de acuerdo al número de subgrupos obtenidos, además este tipo de técnica solo encuentra relaciones locales entre subconjuntos de conceptos lo que puede traer como consecuencias una mala interpretación de los resultados.

El presente trabajo, está estructurado en tres capítulos, distribuidos de la siguiente manera:

Capítulo 1. Fundamentación teórica

Se desarrolla un estudio de los principales sistemas de agrupamiento de tarjetas y de escalamiento multidimensional existentes. Se desarrolla una valoración crítica del diseño propuesto por el analista, un análisis de posibles implementaciones ya existentes y que puedan ser reutilizadas. Además, se describen los lenguajes, las herramientas y metodologías a utilizar para el desarrollo del componente.

Capítulo 2. Propuesta de solución

Se describe la solución propuesta y los procesos básicos, se recoge la especificación de los requisitos funcionales y no funcionales y el modelo de historias de usuario del sistema.

Capítulo 3. Validación de la propuesta de solución

Contiene una descripción de todas las pruebas realizadas para validar el buen funcionamiento de la aplicación.

Además, contiene las Conclusiones, Recomendaciones, Referencias y Anexos.

Capítulo 1. Fundamentación teórica

En este capítulo se exponen los aspectos generales sobre el agrupamiento de tarjetas, centrándose en el escalamiento multidimensional como proceso de evaluación. Se aborda sobre los sistemas que realizan funciones similares a las que se presentan en la solución propuesta. Además, se realiza una valoración de las características fundamentales de las tecnologías, metodología y herramientas a utilizar en la implementación de la técnica.

1.1 Arquitectura de la información

"Se puede definir la arquitectura de información como la actividad y resultado de organizar, clasificar, ordenar, estructurar y describir los contenidos de un espacio, con el fin de que sus usuarios puedan satisfacer sus necesidades informativas con el menor esfuerzo posible"(HASSAN-MONTERO y ORTEGA-SANTAMARÍA, 2009). "*Esta disciplina por su carácter multidisciplinar se nutre de técnicas, metodologías y teorías de una gran variedad de áreas de conocimiento (diseño gráfico, psicología cognitiva, ciencias de la documentación), así como de prácticas profesionales y estudios de casos reales"(HASSAN-MONTERO et al., 2004)*. Una de las principales técnicas usadas en la AI es el agrupamiento de tarjetas la cual permite, desde las primeras etapas del proceso de diseño, anticipar cuál será la organización de categorías o menús de navegación que mejor se adapte al modelo mental de los usuarios.

1.2 Agrupamiento de tarjetas

Esta técnica consiste en solicitar a un grupo de participantes que agrupen los conceptos representados en cada tarjeta por su similitud semántica. El objetivo es, por tanto, identificar qué conceptos, de los representados en cada tarjeta, tienen relación semántica entre sí, e incluso cuál es el grado de esa relación. Se puede diferenciar entre dos tipos de agrupamiento.

- Abierto: el usuario puede agrupar las categorías libremente en el número de conjuntos que crea necesario. Tiene el objetivo de descubrir qué tipo de clasificación de categorías sería más correcto utilizar
- Cerrado: los grupos o conjuntos están predefinidos y etiquetados y el usuario únicamente deberá colocar cada categoría en el grupo que crea corresponda. Está recomendado para verificar si una clasificación de información es familiar y comprensible para el usuario (HASSAN-MONTERO y MARTÍN-FERNÁNDEZ, 2004).

El abierto cumple la función de ayudar a la toma de decisiones organizativas, y el cerrado cumple la función de evaluar esas decisiones. De hecho, ya ambos tipos de agrupamiento tienen propósitos diferentes y complementarios, su utilización combinada puede ofrecer una imagen más fiel del modelo mental del usuario (HASSAN-MONTERO y ORTEGA-SANTAMARÍA, 2009). De manera general, el agrupamiento de tarjetas consta de tres fases: preparación, ejecución y evaluación (ver anexo).

- Preparación: se selecciona el tipo de agrupamiento que se va a realizar, se crean las tarjetas, instrucciones y categorías, que serán usadas en la fase de ejecución.
- Ejecución: los participantes agrupan las tarjetas en las categorías según su criterio, en el caso del agrupamiento abierto los participantes crean sus categorías.
- Evaluación: se construye una matriz cuadrada donde se contabilizan, las coocurrencias entre tarjetas (el número de veces que cada par de tarjetas fue colocada en un mismo grupo por los usuarios) y se le aplica los algoritmos de agrupamientos con el objetivo de encontrar relaciones entre los términos representados por las tarjetas permitiendo así obtener recomendaciones sobre la organización de la información de acuerdo con los patrones de los participantes (MARNET, 2008).

1.2.1 Análisis de las técnicas de evaluación

Para evaluar ejercicios de agrupamiento de tarjetas se pueden utilizar diferentes tipos de técnicas, dentro de las que sobresalen:

Análisis de *Cluster*

El Análisis de *Clusters* (o Análisis de conglomerados) es una técnica de Análisis Exploratorio de Datos para resolver problemas de clasificación. Su objeto consiste en ordenar objetos en grupos (conglomerados o *clusters*) de forma que el grado de asociación/similitud entre miembros del mismo *cluster* sea más fuerte que el grado de asociación/similitud entre miembros de diferentes *clusters*. Cada *cluster* se describe como la clase a la que sus miembros pertenecen (VICENTE-VILLARDÓN, 2006).

Pathfinder (PFNets)

La representación del conocimiento se estructura en redes semánticas, está basado en la teoría matemática de grafos. En la solución que ofrece, las relaciones entre conceptos que se representan, son las que el experto ha considerado importantes. Si la relación entre dos conceptos es muy importante en la representación mental de una persona, esta relación se verá reflejada en un puntero directo entre ellos en el grafo. Por el contrario, si la relación no es importante y puede ser deducida a

partir de las relaciones con otros conceptos, no existirá un puntero directo entre ellos, y la importancia relativa de esta relación se inferirá del número de punteros y nodos que hay que recorrer entre los dos conceptos en el grafo (ECURED, 2011).

Redes Neuronales Artificiales

"*Una red neuronal solo puede aplicarse a agrupamientos cerrados, ya que este método clasifica elementos en grupos previamente configurados. Antes de poder utilizarse, una red neuronal esta necesita una fase de aprendizaje, que puede ser costosa en tiempo. En esta fase la neurona "aprende" que elementos pertenecen a un grupo o a otro" (LAFUENTE et al., 2006)*.

Escalamiento multidimensional

"*El objetivo fundamental de este procedimiento es representar un conjunto de conceptos en un espacio multidimensional, de tal manera que la distancia euclidiana [7]entre dos objetos en ese espacio se corresponda lo más fielmente posible con el juicio de proximidad conceptual asignado por una persona o un conjunto de personas a ese par de conceptos. Este tipo de análisis está estrechamente ligado a las teorías de representación dimensionales, y ofrece una forma de obtener una representación empírica de las dimensiones que relacionan los distintos conceptos de un área de conocimiento determinada"(CAÑAS et al., 2002)*. El EMD es más complejo en cálculo que otros métodos, pero para dar una idea general de la agrupación es el más adecuado, ya que ofrece las dimensiones globales que relacionan todos los conceptos con todos los demás.

Valoración de las técnicas de evaluación

Una vez analizados las posibles técnicas para evaluar ejercicios de agrupamiento de tarjetas se llega a la conclusión de que el más factible es el EMD para dar respuesta al problema existente pues la técnica *Pathfinder* no tiene en cuenta todas las relaciones y las redes neuronales se le aplican solo a agrupamientos cerrados; además el EMD brinda una visión global de los agrupamientos realizados por los usuarios siendo un buen complemento para el análisis de *cluster* con el que cuenta la aplicación.

[7] En un espacio cartesiano, la distancia entre dos puntos *i* y *j* es el segmento que conecta ambos puntos y que viene determinado por la fórmula:$d_{ij}(X) = \left(X_{i1} = X_{j1}\right)^2 + \left(X_{i2} = X_{j2}\right)^2$. Así que:$d_{ij}(X)$es la raíz cuadrada de la suma de la diferencias entre las coordenadaspara cada dimensión del espacio, o simplemente la aplicación delteorema de Pitágoras para la hipotenusa del ángulo recto que se forma con los puntos.

1.3 Escalamiento multidimensional

El escalamiento multidimensional (EMD), más conocido en inglés como *Multidimensional Scaling* (MDS), tiene como objetivo representar las proximidades entre un conjunto de elementos como distancias en un espacio de un número reducido de dimensiones, donde las distancias que median entre los puntos se corresponden con las proximidades entre los objetos por medio de una función de ajuste resultante de un proceso iterativo de optimización, pudiéndose describir las relaciones entre los objetos sobre la base de las proximidades observadas(LÓPEZ-GONZÁLEZ y HIDALGO SÁNCHEZ, 2010).

1.3.1 Modelo general de escalamiento multidimensional

El EMD toma como entrada una matriz de proximidades $\Delta \in M_{nxn}$, donde n es el número de estímulos y cada elemento δ_{ij} de Δ representa la proximidad entre el estímulo i y el estímulo j.

$$\Delta = \begin{pmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & \cdots & \delta_{nn} \end{pmatrix}$$

También se fija el número de dimensiones, p, para hacer el gráfico de los objetos en una solución particular. Generalmente el camino que se sigue es(LINARES, 2001):

1. Arreglar los n objetos en una configuración inicial en p dimensiones, esto es, suponer para cada objeto las coordenadas (x1, x2,..., xp) en el espacio de p dimensiones.
2. Calcular las distancias euclidianas entre los objetos de esa configuración, esto es, calcular las d_{ij}, que son las distancias entre el objeto i y el objeto j.

$$d_{ij} = \left[\sum_{t=1}^{m} \left| X_{it} - X_{jt} \right|^2 \right]^{\frac{1}{2}}$$

3. Hacer una regresión de d_{ij} sobre δ_{ij}. Esta regresión puede ser lineal, polinomial o monótona.

Por ejemplo, si se considera lineal se tiene el modelo

$$d_{ij} = a + b\delta_{ij} + \varepsilon$$

y utilizando el método de los mínimos cuadrados se obtienen estimaciones de los coeficientes a y b, y de ahí puede obtenerse lo que genéricamente se conoce como una “disparidad”

$$\hat{d}_{ij} = \hat{a} + \hat{b}\delta_{ij}$$

Si se supone una regresión monótona, no se ajusta una relación exacta entre d_{ij} y δ_{ij}, sino se supone simplemente que si δ_{ij} crece, entonces d_{ij} crece o se mantiene constante.

4. Medir la bondad de ajuste entre las distancias de la configuración y las disparidades. Existen diferentes definiciones de este estadístico, pero la mayoría surge de la definición del llamado índice de esfuerzo (en inglés: STRESS).

$$\text{STRESS1} = \sqrt{\frac{\sum\sum\left(d_{ij} - \hat{d}_{ij}\right)^2}{\sum\sum d_{ij}^2}}$$

$$\text{SSTRESS1} = \sqrt{\frac{\sum\sum\left(d_{ij}^2 - \hat{d}_{ij}^2\right)^2}{\sum\sum d_{ij}^4}}$$

Todas las sumatorias sobre i y j van de 1 a p y las disparidades dependen del tipo de regresión utilizado en el tercer paso del procedimiento.

El STRESS1 es la fórmula introducida por *Kruskal* quien ofreció la siguiente guía para su interpretación:

Tabla 1. Interpretación de STRESS1

Valor del STRESS1	Interpretación
0.2	Pobre
0.1	Regular
0.05	Bueno
0.025	Excelente
0.00	Perfecto

5. Las coordenadas (x1, x2,..., xt) de cada objeto se cambian ligeramente de tal manera que la medida de ajuste se reduzca.

Los pasos del 2 al 5 se repiten hasta que al parecer la medida de ajuste entre las disparidades y las distancias de configuración no pueda seguir reduciéndose. El resultado final del análisis es entonces las coordenadas de los n objetos en las p dimensiones. Estas coordenadas pueden usarse para

elaborar un gráfico que muestre cómo están relacionados los objetos.

1.3.2 Tipos de escalamiento multidimensional

Existen dos modelos básicos de EMD que son: el modelo de escalamiento métrico (EMD-M) y el modelo de escalamiento no métrico (EMD-NM). La elección entre uno u otro depende especialmente de la calidad métrica de los datos de partida.

- EMD-M: se recomienda cuando las proximidades son valores numéricos bien determinados, donde las desemejanzas son magnitudes muy cercanas a las distancias reales (LÓPEZ-GONZÁLEZ y HIDALGO SÁNCHEZ, 2010).
- EMD-NM: en ámbitos donde intervienen datos de preferencia y juicios de apreciación humana muy probablemente sujetos a error, las desemejanzas deben ser interpretadas en sentido ordinal, y se asume que la función que relaciona proximidades y distancias ha de ser monótona, por lo que para evaluar técnicas de agrupamiento de tarjetas se recomiende este modelo (LÓPEZ-GONZÁLEZ y HIDALGO SÁNCHEZ, 2010).

1.4 Escalamiento multidimensional no métrico

El EMD-NM establece una relación monótona creciente entre las proximidades y las distancias, es decir, sí $\delta_{ij} < \delta_{kl} \Rightarrow d_{ij} \leq d_{kl}$. El procedimiento se basa en los siguientes apartados(CASAS y HURTADO, 2002):

1. Transformación de la matriz de proximidades en una matriz de rangos, asignando a la desemejanza menor el rango 1; a la siguiente el 2, hasta llegar a la mayor que tendrá el rango n (n-1)/2.
2. Se sitúan los ítems en un espacio de m dimensiones, lo que supone determinar la matriz de configuración inicial, $X \in M_{nxm}$ de coordenadas aleatorias, que nos da la distancia entre los estímulos.
3. Comparación de las proximidades con las distancias, obteniéndose las disparidades $(\hat{d}_{ij})$
4. Definición del Stress.
5. Minimización del Stress.

Existen otros algoritmos que permiten realizar el EMD-NM pero no se tendrán en cuenta, pues para implementar un algoritmo de EMD que analice los datos obtenidos tras aplicar la técnica de agrupamiento de tarjetas se debe elegir el escalamiento multidimensional clásico no métrico, que sigue el modelo de *Kruskal* (ver figura 1).

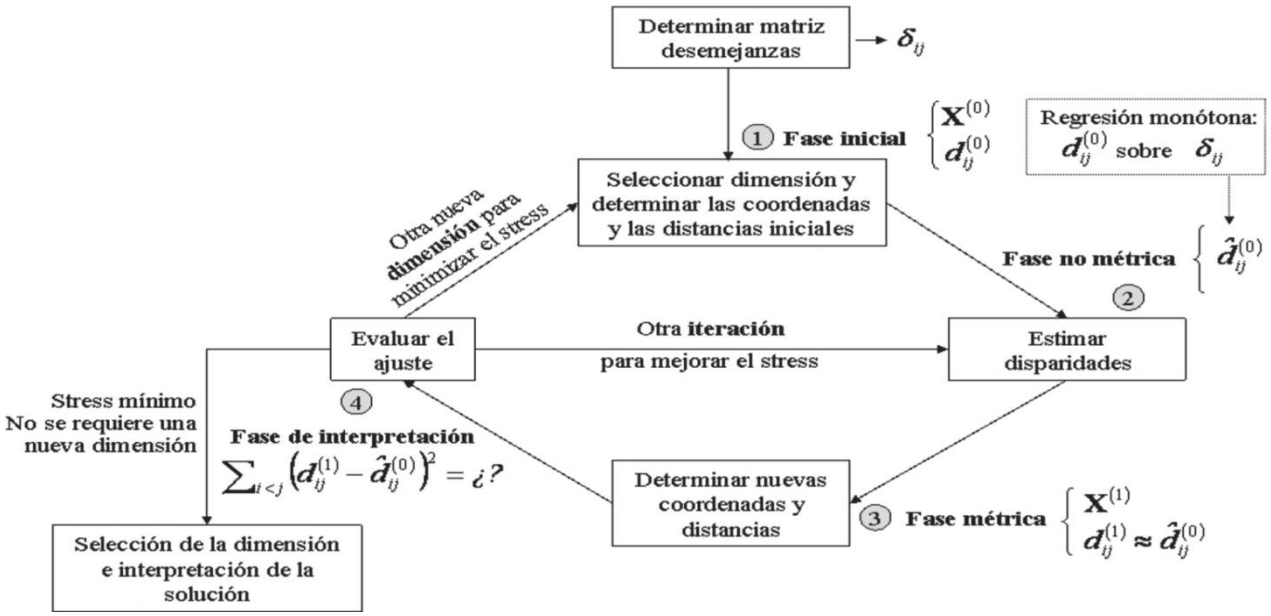


Figura 1. Algoritmo para el EMD-NM de Shepard-Kruskal

1.6 Análisis de soluciones existentes para el agrupamiento de tarjetas

En la presente investigación se analizaron un grupo de aplicaciones destinadas al agrupamiento de tarjetas con el fin de conocer cuáles de estas utilizaban el EMD como método de evaluación, también se analizaron un conjunto de herramientas estadísticas que tienen incorporados este método para identificar posibles características que se le puedan incorporar a la solución que se propone.

1.6.1 Herramientas de agrupamiento de tarjetas

Dentro de las aplicaciones que ejecutan algunos o casi todos los procesos del agrupamiento de tarjetas se analizaron variantes como:

- Web Sort: utilizado por Yahoo, Paypal, Oracle (LIME y CHILE, 2009)
- OptimalSort*:* desarrollada por *OptimalWorkshop* que desarrollaron también soluciones remotas de experiencia de usuario. Tiene un valor total de ciento nueve dólares por mes con estudios ilimitados durante ese tiempo (WORKSHOP, 2009)
- Xsort: diseñada para Mac OS X (XSORT, 2005)
- Uxsort: Creado por los investigadores de experiencia de usuario de Microsoft (USABILITYCENTRIC, 2004)
- uzCardSort: licencia MPL de herramientas, basado en el navegador web Mozilla (UZCARDSORT, 2004)
- CardSword: licencia GPL (CARDSWORD, 2005)
- aCaSo: desarrollada en el Laboratorio Aragonés de Usabilidad (LAFUENTE *et al.*, 2006).

Valoración de las herramientas de agrupamiento de tarjetas

De manera general todas o casi todas las aplicaciones existentes que se tienen en cuenta, el método de evaluación que utilizan el *clustering* jerárquico para tratar sus datos. Además, para jerarquizar un grupo es el método más conveniente. El EMD que es el más adecuado para obtener grupos generales, un primer nivel de agrupación y el más fácil de interpretar se identificó solo en el desarrollo aCaSo una iniciativa propia del laboratorio de usabilidad en España. Estas aplicaciones además de contar con los métodos de *clustering,* generan un fichero con la matriz de coocurrencias para que el arquitecto de información, en caso de que se quiera apoyar en otro tipo de evaluación como el EMD, evalúe el mismo haciendo uso de *software* estadísticos.

Posteriormente se realizó un estudio de programas estadísticos muy conocidos y utilizados a nivel mundial que traen paquetes incluidos para el EMD, con el objetivo de identificar posibles características que se puedan tener en cuenta en la solución que se propone. Las herramientas que se analizaron fueron SPSS, SAS y R, existen otras como *Stata* y *Systat*, pero no se tuvieron en cuenta por poseer estructuras similares a SPSS; solo se consideró SPPS por su mayor popularidad (SALAS, 2008).

1.7 Principales *software* estadísticos

***Statical Package for the Social Sciences* (SPSS)**

Es una potente aplicación de análisis estadísticos y gestión de datos, dotada de una intuitiva interfaz gráfica que utiliza menús descriptivos y cuadros de diálogo sencillos que realizan la mayor parte del trabajo resultando muy fácil de manejar. Es muy popular debido a la capacidad de trabajar con bases de datos de gran tamaño. El programa consiste en un módulo base y módulos anexos que se han ido actualizando constantemente con nuevos procedimientos estadísticos. Cada uno de estos módulos se compra por separado.

Es una tecnología que(PINTO MOLINA, 2011):

- automatiza el proceso de descubrimiento del conocimiento
- ayuda a centrarse en un área de interés
- permite predecir resultados
- permite encontrar patrones dentro de un fichero de datos
- amplía las capacidades ofrecidas por otras herramientas.

SAS/STAT

Software diseñado para necesidades analíticas, tanto empresariales como especializadas. Proporciona un conjunto completo de soluciones que pueden satisfacer las necesidades de análisis de datos de todas las áreas de la empresa (SAS/STAT®, 2011). SAS, a diferencia de SPSS, es un programa que requiere el ingreso de comandos para ejecutar gran parte de sus rutinas y opciones. Por lo tanto, necesita del conocimiento de la sintaxis antes de su uso (SALAS, 2008).

***Software* estadístico R**

Es un programa estadístico y un lenguaje de programación de uso libre, de distribución gratuita y de código abierto, desarrollado como un gran proyecto colaborativo de estadísticos de diversos países y disciplinas. También es un programa basado sobre comandos, en el que se puede acceder a todos los procedimientos y opciones a través de sintaxis computacional (SALAS, 2008).

Comparación entre SPSS, SAS y R

A continuación se muestran algunos de los resultados obtenidos en el trabajo titulado "¿Por qué comprar un programa estadístico si existe R?" en el cual se realizó una comparativa de los *software* estadísticos SPSS, SAS y R, los resultados arrojados fueron (SALAS, 2008):

Calidad de gráficos

SPSS ofrece una serie de gráficos tipo que si bien pueden modificarse en su formato (leyendas y color), son difíciles de personalizar. Aunque tanto SAS como R permiten el diseño personalizado de gráficos, SAS requiere del uso de diferentes rutinas (o paquetes). Las sintaxis de R son más sencillas (o "planas") y no requieren de una gran cantidad de paquetes, también ofrece una amplia gama de formatos en los cuales los gráficos pueden ser exportados, sin necesidad de mayor sintaxis. Finalmente, a título subjetivo, la calidad visual de un gráfico en R parecería ser superior a la de SAS y a la de SPPS.

Control de procesos

Los procedimientos estadísticos usan una serie de algoritmos que poseen diferentes variantes. Cuando un usuario no los conoce con profundidad, normalmente, el programa usa variantes predefinidas de estos algoritmos. En todos los programas, estas definiciones pueden ser especificadas. En SPSS, sin embargo, resulta complejo cambiarlas (solo están disponibles algunas básicas). Dado que SAS y R requieren sintaxis, también permiten un mayor control de los procedimientos estadísticos a ser ejecutados. R es más flexible por ser de código abierto; un usuario

puede usar las funciones programadas en el *software* como también escribir funciones propias de manera sencilla.

Sistemas operativos

Tanto SPSS como SAS pueden funcionar en el S.O. Linux, pero su configuración es compleja. SPPS también puede ejecutarse en Macintosh. R es el único que funciona de manera estable e íntegra en los tres sistemas operativos de mayor uso. La versatilidad de plataformas donde R puede ser instalado ofrece una ventaja para los diferentes usuarios en distintas disciplinas.

Integración con otros sistemas

R a través de la librería JRI (*Java/R Interface)*, la cual es parte del paquete rJava, permite crear un puente de comunicación de Java a R vía JNI[8] (*Java Native Interface*), mediante el cual puedes acceder y utilizar las funciones del R (LANDA-TORRES *et al.*, 2011). En cambio, tanto SPSS como SAS no cuentan con este tipo de ventajas por lo que su integración con otros sistemas para utilizar sus funcionalidades se hace complejo.

Tabla 2. Comparación de aspectos generales entre los programas estadísticos SPSS, SAS y R

Aspectos	**Programa estadístico**		
	SPSS	SAS	R
Calidad de gráficos	Regular	Buena-Excelente	Excelente
Control de procesos	Baja	Excelente	Excelente
Licencia	Propietaria	Propietaria	GPL
Código fuente disponible	No	No	Sí
Tipos de EMD	Ambos	Ambos	Ambos
Multiplataforma	Sí	Sí	Sí
Integración con otros sistemas	No	No	Sí

[8] La interfaz nativa de Java (JNI) permite la integración de código escrito en el lenguaje de programación Java con código escrito en otros lenguajes como C y C + +. Se permite a los programadores aprovechar al máximo la plataforma Java sin tener que abandonar su inversión en el código heredado.

Valoración de los programas estadísticos

Tras el análisis realizado anteriormente de las herramientas estadísticas se identificaron un conjunto de características que se pueden tener en cuenta para dar respuesta al problema existente, pues el *software* estadístico R cuenta con los elementos necesarios para dar solución el problema planteado. Además de ser una herramienta libre y contar con los métodos necesarios para el EMD-NM, permite integrarse fácilmente a herramientas desarrolladas en la plataforma Java mediante la librería JRI del paquete rJava, e incluso ofrece la posibilidad de acceder a sus funcionalidades y adaptarlas según a la necesidad del cliente.

1.6 Metodologías, herramientas y lenguajes

Metodologías ágiles de desarrollo

Las Metodologías Ágiles o "Ligeras" constituyen un nuevo enfoque en el desarrollo de *software*, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales, debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

El término ágil nace en febrero del 2001 en reunión celebrada en Utah-EEUU. Participaron un grupo de expertos de la industria del *software*, incluyendo algunos creadores y promotores de las metodologías de desarrollo. La reunión se realizó con el objetivo de proponer alternativas a los procesos de *software* tradicionales que se caracterizan por ser poco flexibles y estar dirigidos por una gran cantidad de documentación que se genera en las distintas etapas del desarrollo del *software*. Como base para este nuevo enfoque de metodología se crea El Manifiesto Ágil, documento que resume la filosofía ágil. En él se valoran cuatro aspectos fundamentales que resumen los doce principios que guían a este tipo de metodología.

Aspectos valorados en EL Manifiesto Ágil (CANÓS *et al.*, 2003):

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas:** Al recurso más importante de cualquier proyecto es el personal con el que dispone para construir el *software*. El recurso humano calificado con capacidades técnicas adecuadas, facilidades para adaptarse al entorno, trabajar en equipo e interactuar convenientemente con el usuario, da mayor garantía de éxito que contar con herramientas y procesos rigurosos.

- **Desarrollar *software* que funciona más que conseguir una buena documentación:** Se reconoce la importancia de mantener una documentación completa y actualizada, pero con la misma claridad se hace énfasis en que se deben producir los documentos estrictamente

necesarios; los documentos deben ser cortos y limitarse a lo fundamental, dando una mayor prioridad a la construcción de un *software* que funcione correctamente y que cumpla con las expectativas del cliente.

- **La colaboración con el cliente más que la negociación de un contrato:** Generalmente es el cliente que indica o solicita que debe hacer el *software* y espera por el mismo durante el plazo establecido, de acuerdo con sus expectativas. Cambiando este enfoque y complementando la importancia que adquiere el equipo de desarrollo, se incluye el cliente como parte del equipo de trabajo, su participación debe estar presente desde el inicio del proceso de desarrollo del *software* hasta su culminación. Se busca la colaboración y el beneficio común sin evadir las responsabilidades de ambas partes.

- **Responder a los cambios más que seguir estrictamente un plan:** Un producto de *software* se enfrenta constantemente a cambios durante su proceso de desarrollo. Por lo que se propone que la planificación no sea estricta sino que sea adaptable y abierta ante nuevos cambios en los requisitos, tecnologías en el equipo y otros componentes del proyecto.

Modelo de calidad

Un modelo de calidad es, por lo tanto, un conjunto de prácticas vinculadas a los procesos de gestión y el desarrollo de proyectos. Este modelo supone una planificación para alcanzar un impacto estratégico, cumpliendo con los objetivos fijados en lo referente a la calidad del producto o servicio (DEFINICION.DE, 2008-2012).

Es un conjunto de buenas prácticas que pueden aplicarse durante el ciclo de vida del *software*. Están enfocadas fundamentalmente en los procesos de gestión y desarrollo de proyecto. Indican qué se debe hacer para aumentar la calidad del producto, no como hacerlo, debido a que se depende de la metodología de desarrollo usada y los objetivos específicos del negocio. Guían a las organizaciones a la mejora continua y a la competitividad, dándoles especificaciones de qué tipo de requisitos o actividades deben de desarrollar para poder brindar productos y servicios de alto nivel.

Modelo de Capacidad y Madurez Integrado (CMMI)

CMMI es un modelo de referencia para el crecimiento de capacidades y madurez que proporciona a las organizaciones los elementos esenciales para procesos eficaces y se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y *Software*. Fue creado por el SEI (*Software Engineer Institute*) y está compuesto por veinte dos Áreas de proceso (AP) las que se distribuyen en 5 niveles según las representaciones del modelo (GARCÍA, 2005).

Niveles de capacidad de los procesos

Los 6 niveles definidos en CMMI para medir la capacidad de los procesos son (CHRISSIS *et al.*, 2009):

0. Incompleto: el proceso no se realiza, o no se consiguen sus objetivos.
1. Ejecutado: el proceso se ejecuta y se logra su objetivo.
2. Gestionado: además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumple los requisitos.
3. Definido: además de ser un proceso gestionado se ajusta a la política de procesos que existe en la organización, alineada con las directivas de la empresa.
4. Cuantitativamente gestionado: además de ser un proceso definido se controla utilizando técnicas cuantitativas.
5. Optimizante: además de ser un proceso cuantitativamente gestionado, de forma sistemática se revisa y modifica o cambia para adaptarlo a los objetivos del negocio. Mejora continua.

Proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI

La UCI actualmente se encuentra inmersa en un proceso de mejora de los procesos que se desarrollan como parte de la construcción del *software* basado en el modelo de CMMI. Este proceso de mejora tiene como objetivo que la universidad alcance el nivel dos del modelo CMMI, lo que la convertiría en la primera institución en el país en alcanzar este nivel. El objetivo es asegurar que la organización está basada en procesos y con un programa de mejora continua alineado con sus objetivos de negocio. Ayudar la UCI a establecer las bases y fundamentos para seguir mejorando sus procesos y fortalecer su cultura de calidad en el desarrollo de *software*. Alinear los procesos de desarrollo de *software* con los principios y requisitos del modelo CMMI, estableciendo planes de mejora con los que la organización oriente sus procesos hacia la consecución de sus metas (CALISOFT, 2009).

El proceso de desarrollo ágil con segundo nivel de CMMI tiene definido el siguiente ciclo de vida:

- **Estudio preliminar:** se realiza un estudio profundo de la organización cliente que posibilita obtener la información requerida para determinar el alcance del proyecto, así como la estimación del costo, tiempo y el esfuerzo.
- **Modelado de negocio:** se comprende el negocio de la entidad con el objetivo de que el *software* a desarrollar cumpla con lo que realmente quiere el cliente. Se pueden utilizar técnicas para la descripción del modelado del negocio como la Notación de Modelado de Proceso de Negocio.
- **Requisitos:** el objetivo fundamental es desarrollar el modelo del sistema, identificando los requisitos funcionales y no funcionales con las descripciones correspondientes en cada caso.

- **Análisis y diseño:** se realiza el análisis y el modelado del sistema a partir de los requisitos definidos previamente.
- **Implementación:** a partir de los artefactos obtenidos durante el análisis y diseño se procede a realizar la implementación del *software* en términos de componentes de implementación.
- **Pruebas internas:** se realizan las pruebas internas con el equipo del proyecto en cada una de las iteraciones o versiones finales próximas a ser liberadas, según lo defina el proyecto. Se desarrollan artefactos de pruebas como: diseños de casos de prueba y listas de chequeo.
- **Pruebas de liberación:** pruebas realizadas por parte de la oficina o institución encargada de la calidad y de la certificación del proyecto a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- **Despliegue:** se realiza la entrega de la aplicación al cliente, así como la configuración y prueba en el ámbito del cliente. Las pruebas realizadas durante esta fase incluyen pruebas de aceptación y pruebas piloto. Se debe realizar además capacitaciones a los trabajadores del sistema.
- **Soporte:** por un tiempo limitado el proyecto ofrecerá un servicio para resolver conflictos y problemas de usabilidad y rendimiento del *software* entregado al cliente, suministrándole actualizaciones y parches a errores.

1.6.1 Herramientas de modelado

Las herramientas de modelado, permiten crear un "simulacro" del sistema, a bajo costo y riesgo mínimo. A bajo costo porque, al fin y al cabo, es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real (el cual es más costoso). Además, minimizan los riesgos, porque los cambios que se deban realizar (por errores o cambios en los requisitos), se pueden realizar más fácil y rápidamente sobre el modelo y sobre el sistema ya implementado (SCRIBD, 2011).

Visual Paradigm 8.0

Es una herramienta profesional muy potente que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Diseñado para varios tipos de usuarios, incluyendo Ingenieros de *Software*, Analistas de Sistemas, Analistas de Negocio y Arquitectos de Sistema. Permite realizar Diagramas de procesos de negocios, Modelado UML, Modelos de casos de usos, Modelos de actividad, de interacción, de bases de datos, de entidad-relación. Además posee integración para varios Entornos Integrados de Desarrollo (IDE), puede realizar Ingeniería de Código y también generar documentación, entre otras cosas; todo bajo un modelo colaborativo. Soporta todas las necesidades de diseño y modelado a lo largo del ciclo de vida

de desarrollo de *software*, es una herramienta que ayuda a construir aplicaciones de calidad, de manera más rápida, óptima y más barata (GRUPO-SATÉLITE-SA, 2011).

Es posible generar código desde Visual Paradigm para plataformas como .NET, Java y PHP, así como obtener diagramas a partir del código, esto es de gran utilidad pues ahorra tiempo a los desarrolladores y reduce las posibilidades de cometer errores. Brinda la posibilidad de obtener una base de datos relacional y el código necesario para acceder a esta a partir de un Diagrama Entidad Relación, además se conecta fácilmente a varios servidores de base de datos. Establece interoperabilidad con otras aplicaciones como el *Microsoft Visio* y el *Rational Rose* y permite documentar todo el trabajo y especificaciones de Casos de Usos sin necesidad de utilizar herramientas externas, por ejemplo editores de texto, utilizando plantillas que se encuentran o que pueden ser creadas por los usuarios. Disponible en múltiples lenguajes y plataformas: *Microsoft Windows* (98, 2000, XP, o Vista), GNU Linux, Mac OS X, Solaris o Java (VISUAL-PARADIGM, 2010).

Evolus Pencil 1.3.4

Es una extensión de Firefox que se utiliza para el diseño de los prototipos de las interfaces de usuario, se caracteriza por (PROJECT, 2010):

- Brindar un conjunto de componentes como: entradas de texto, íconos y botones.
- Ser multi-página, creación simultánea de varios documentos.
- Edición en pantalla de los elementos de texto.
- Permitir exportar imágenes al formato PNG, HTML o PDF.
- Permitir operaciones estándar de dibujo: alineado, escalado, rotación, entre otras.
- Ser Multi-plataforma.
- Posibilitar, a través de las propiedades de los componentes, cambiar el estilo al diseño.

Lenguaje unificado de modelado (UML) 2.1

UML (Lenguaje Unificado para la Construcción de Modelos) se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de *software*. Es un sistema notacional (que, entre otras cosas, incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Este lenguaje permite que personas con poco conocimiento de programación puedan participar en el análisis y diseño de un sistema (LARMAN, 2004).

Se seleccionó el lenguaje UML para modelar los artefactos generados en el proceso de desarrollo de *software*, ya que el uso de lenguajes visuales facilita el entendimiento por parte del equipo de

desarrollo sobre el sistema que se modela. Además, proporciona una forma estándar de modelado, cubriendo todo lo relacionado a los procesos del negocio.

1.6.2 Lenguaje de programación.

Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems[9] a inicios de los años noventa. Es el lenguaje más revolucionario de los últimos años, cuya característica principal es su capacidad para ejecutarse en plataformas como Unix, Windows y OS/2. Entre sus principales características se encuentran (QUAST, 2011):

- **Orientado a objeto:** trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.
- **Distribuido:** proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir, establecer y aceptar conexiones con servidores o clientes remotos, facilitando la creación de aplicaciones distribuidas.
- **Robusto:** fue diseñado para crear *software* altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores, porque se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.
- **Seguro:** una de las características más importantes de Java es su seguridad. No permite acceso directo a memoria y el manejo y aritmética de punteros. Además, está prohibido el acceso a archivos locales por parte de los navegadores de Internet.
- **Arquitectura neutral:** el código generado por el compilador Java es independiente de la arquitectura: podría ejecutarse en un entorno UNIX, Mac o Windows. El motivo de esto es que el que realmente ejecuta el código generado por el compilador no es el procesador del ordenador directamente, sino que este se ejecuta mediante una máquina virtual. Esto permite que todos los programas que se desarrollen en Java puedan ejecutarse en cualquier máquina que se conecte a ella independientemente del sistema operativo que emplee siempre y cuando el ordenador en cuestión tenga instalada una máquina virtual de Java.

[9] Sun Microsystems es una empresa informática comprada por la Corporación Oracle antes era parte de Silicon Valley, fabricante de semiconductores y *software*.

- **Portable:** al ser de arquitectura neutral es altamente portable, pero esta característica puede verse de otra manera: los tipos estándares están igualmente implementados en todas las máquinas por lo que las operaciones aritméticas funcionaran igual en todas las máquinas. Sus programas son iguales en cualquiera de las plataformas, este lenguaje especifica tamaños básicos, esto se conoce como la máquina virtual de Java.
- **Multithread:** soporta de modo nativo los *threads*, sin necesidad del uso de librerías específicas, esto le permite además que cada *thread* de una aplicación Java pueda correr en una CPU distinta, si la aplicación se ejecuta en una máquina que posee varias CPU.

1.6.3 Entorno integrado de desarrollo (IDE)

Un entorno de desarrollo integrado (en inglés *Integrated Development Environment* o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos(EUTIO, 2009).

Las herramientas que normalmente componen un IDE son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones. Proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual Basic entre otros). Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación.

Netbeans 7.0

NetBeans es un entorno de desarrollo integrado (IDE), una herramienta que les permite a los programadores escribir, compilar, depurar y ejecutar programas. Puede utilizarse para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso.

La Plataforma NetBeans es una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de *software*, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones(NETBEANS, 2006).

1.6.4 *Software* estadístico R

R es un lenguaje y un entorno para el cálculo estadístico y el dibujo de gráficas. Es un proyecto GNU

similar al lenguaje y entorno comercial S desarrollado por los laboratorios Bell. R se puede considerar como una implementación distinta de S. Hay importantes diferencias, pero la mayoría del código escrito para S funciona sin apenas alteración en R. Proporciona una extensa variedad de técnicas y gráficas estadísticas (modelado lineal y no lineal, *tests* de estadística clásica, análisis de series temporales, clasificación y *clustering*), y es muy extensible.

Uno de los puntos fuertes de R es la facilidad para producir o diseñar gráficas de gran calidad de diseño, incluyendo los símbolos y fórmulas matemáticas necesarias. Está disponible como *software* libre bajo el contrato GNU de la *Free Software Foundation*. Se puede compilar y ejecutar en una gran variedad de sistemas UNIX (como FreeBSD y Linux). También en Windows y MacOS. R es una solución integrada para la manipulación de datos, cálculo y generación de gráficas la cual incluye (CICA, 2007):

- un manejo efectivo de datos y facilidad para el almacenamiento de los mismos
- conjunto de operadores para el cálculo sobre *arrays*, en concreto sobre matrices
- coherente e integrada colección de herramientas intermedias para el análisis de datos
- facilidades gráficas para el análisis de datos y su visualización
- un lenguaje de programación bien diseñado, efectivo y sencillo que incluye estructuras de control condicionales, bucles, funciones recursivas y facilidades para la entrada/salida.

Paquete rJava

rJava es una herramienta que es posible utilizarse con Java, la cual es una interfaz que habilita un puente de comunicación entre Java y R, creando objetos y llamando métodos adecuados para su comunicación vía JNI (forma de comunicación que se realiza sobre la base del lenguaje C). Existen dos vías de comunicación entre los dos lenguajes mencionados, la primera es con JRI (Java R Interface) que es la librería que logra la comunicación de Java a R, mientras que rJava lo hace en sentido contrario, pero las dos interfaces se encuentra incluidas en el proyecto que lleva por nombre rJava (LANDA-TORRES *et al.*, 2011).

Librería JRI

JRI es una interfaz de Java / R, que permite ejecutar aplicaciones Java en el interior R como un solo hilo. Básicamente se carga a R como una biblioteca dinámica en Java y proporciona una API[10] de Java para la funcionalidad R. Es compatible tanto con simples llamadas a las funciones de investigación y

[10] Interfaz de programación de aplicaciones API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

un REPL[11] en plena ejecución (R-PROJECT, 2005).

Función isoMDS

Es la implementación del método de *Kruskal* de EMD-NM y esta se encuentra en el paquete MASS. A partir de la matriz de distancias euclidianas se obtiene una matriz de coordenadas la cual puede ser representada gráficamente mediante la función *plot*. Trabaja por defecto con dos dimensiones, pero dándole valores a la variable *k* se obtienen resultados en el número de dimensiones especificadas, también se le puede especificar el máximo de iteraciones asignándole un valor a la variable *maxit (RIPLEY, 2011)*.

Conclusiones parciales

Después de haber realizado una profunda investigación sobre las técnicas que pueden ser utilizadas para evaluar ejercicios de agrupamiento de tarjetas, se identificó que el EMD era el más indicado para dar respuesta al problema planteado. De las herramientas homólogas analizadas se identificó que solo una de ellas utiliza el EMD entre sus técnicas de evaluación, la cual no se puede tener en cuenta por ser un desarrollo propio del Laboratorio Aragonés de Usabilidad.

Por otra parte, el análisis a los programas estadísticos, permitió encontrar en uno de ellos, el R, una solución sencilla para un proceso tan complejo. Además, se realizó un estudio de varias herramientas y tecnologías algunas de las cuales están propuestas por el centro, con la finalidad de determinar cuáles de ellas eran las adecuadas para el correcto desarrollo del módulo.

[11] Por sus siglas en inglés *read-eval-print loop*. Se refiere al ciclo de leer, evaluar e imprimir, La función de lectura acepta una expresión por parte del usuario, y lo convierte en una estructura de datos en la memoria. La función evaluación toma esta estructura de datos interna y la evalúa. La función de impresión toma el resultado obtenido de la evaluación, y lo imprime para el usuario.

Capítulo 2. Propuesta de solución

En el presente capítulo se describe la solución propuesta. Se define el estado actual de los procesos involucrados en el campo de acción que son objeto de informatización. Se especifican los requisitos funcionales y no funcionales que el módulo debe de cumplir, los cuales permiten obtener una concepción general del mismo. Se definen además los patrones a utilizar y se presenta una propuesta de la herramienta.

2.1 Objeto de automatización

Con el presente trabajo se pretende desarrollar un módulo capaz de mejorar el proceso de evaluación de la herramienta de agrupamiento de tarjetas. El proceso que será objeto de automatización es el EMD-NM, el cual permitirá a los arquitectos de información tener una mejor visión e interpretación de los resultados obtenidos, también servirá de complemento para los métodos de análisis de *clustering* con los que cuenta la herramienta actualmente.

2.2 Propuesta del sistema

El módulo que se desarrollará tiene como objetivo mejorar el proceso de evaluación de la herramienta de agrupamiento de tarjetas y servir de complemento a los métodos de evaluación con que cuenta la misma. La solución consiste en la integración de herramienta de agrupamiento de tarjetas con el *software* estadístico R mediante la librería JRI la cual es parte del paquete rJava. Esta integración será muy beneficiosa debido la que el *software* R cuenta con algoritmos necesarios para llevar a cabo la técnica EMD-NM, así como las funciones necesarias para visualizar los resultados en una gráfica de dispersión.

El punto de partida para realizar el EMD-NM será la matriz de coocurrencias obtenida tras realizar un ejercicio de agrupamiento de tarjetas, haciendo uso de la función *dist*[12] se obtendrá la matriz de distancia euclídea y con la función *isoMDS* se obtiene la matriz con las coordenadas de cada tarjeta, la cual será representada en una gráfica de puntos. Para hacer más visible el gráfico resultante, se le asignará un número a cada tarjeta, dicha relación, *-número – tarjeta-* se mostró al lado de la gráfica en una leyenda (ver figura 2). También se utilizarán otras funcionalidades del R como:

- rbind: esta función permite para crear la matriz de coocurrencias en el R
- dimnanes: se utilizará para asignarle un número a cada tarjeta.

[12] En la instrucción dist (x, *method=euclidean*), '*method'* puede ser algunos de los siguientes términos: *euclidean*, *maximum*, *manhattan*, *canberra*, *binary* o *minkowski*.

- library: se utilizará para llamar a la librería MASS la cual cuenta con el algoritmo de EMD-NM
- plot: permitirá representar gráficamente los resultados obtenidos tras aplicar el EMD-NM
- jpeg: se utilizará para visualizar la gráfica obtenida tras utilizar la función plot.

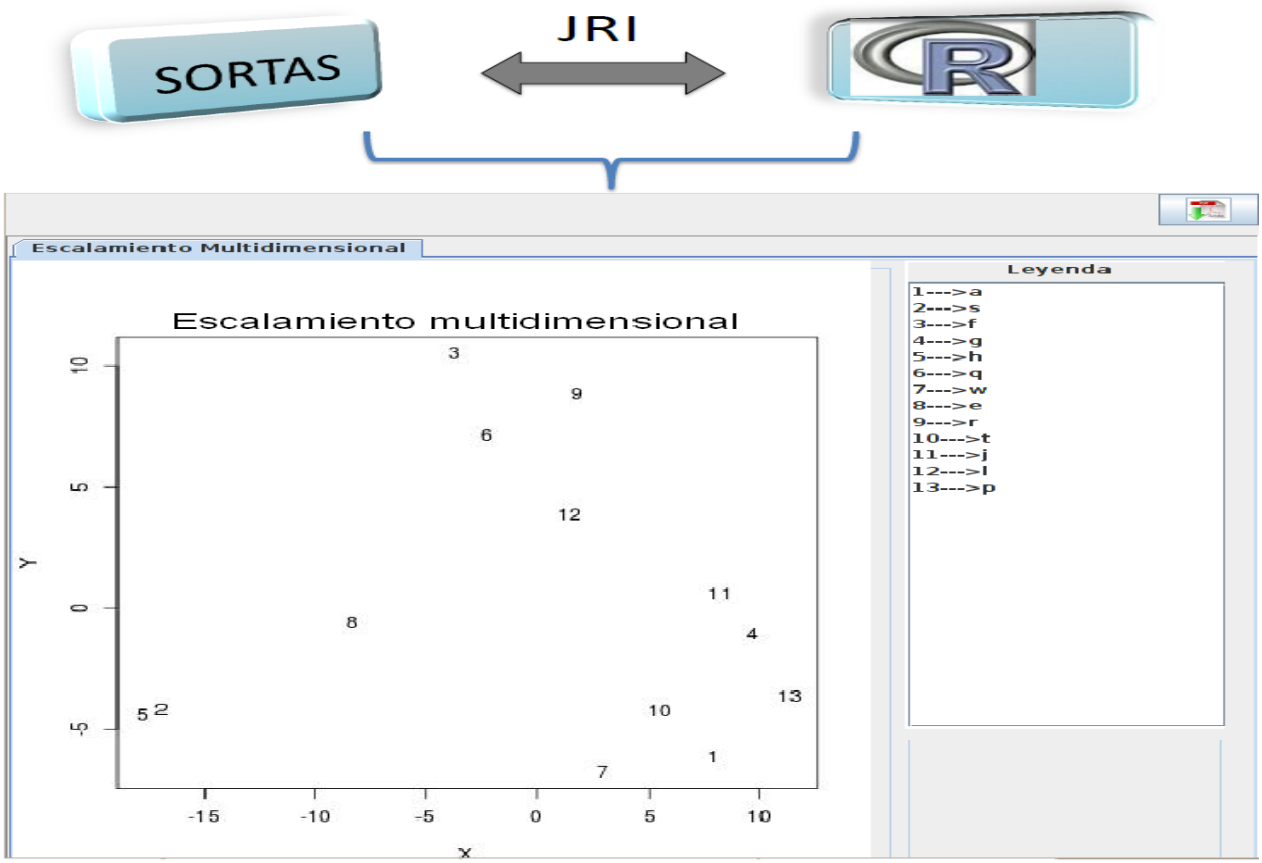


Figura 2. Propuesta de solución: Integración SORTAS - R

2.3 El contexto del sistema. Modelo de dominio

Modelo de Dominio o Conceptual

"*Es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física"* (JACOBSON *et al.*, 2004).

Este modelo permitirá mostrar de manera visual los principales conceptos que se manejan, ayudando a los usuarios, desarrolladores e interesados; a utilizar un vocabulario común para poder entender el

contexto en que se desarrolla el sistema. Además, contribuirá a identificar personas, eventos, transacciones y objetos involucrados en el sistema.

Entidades

- **AI:** Arquitecto de información es la persona a la que destinada la herramienta y la cual interactúa con ella
- **Fichero xml:** entidad encargada de almacenar los resultados obtenidos en cada de fase de un ejercicio de agrupamiento de tarjetas
- **Evaluación:** entidad encargada de mostrar los algoritmos que pueden ser utilizados para evaluar un ejercicio
- **algoritmos de evaluación:** son los diferentes métodos que puede utilizar el AI para evaluar un ejercicio
- **Resultados:** entidad encargada de mostrar los resultados obtenidos tras aplicar los algoritmos de evaluación.

A continuación se muestra el modelo de dominio perteneciente a la herramienta de agrupamiento de tarjetas, el cual está compuesto por 5 clases conceptuales (Ver figura 3).

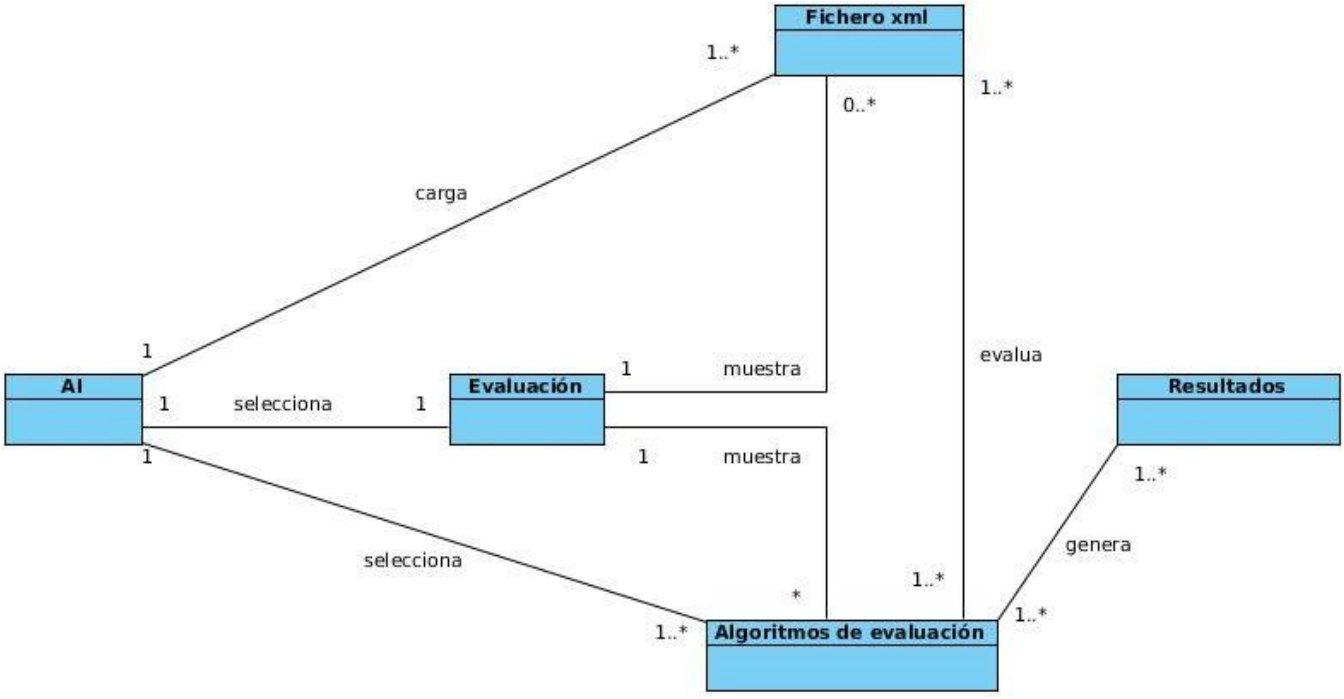


Figura 3. Diagrama del modelo de dominio

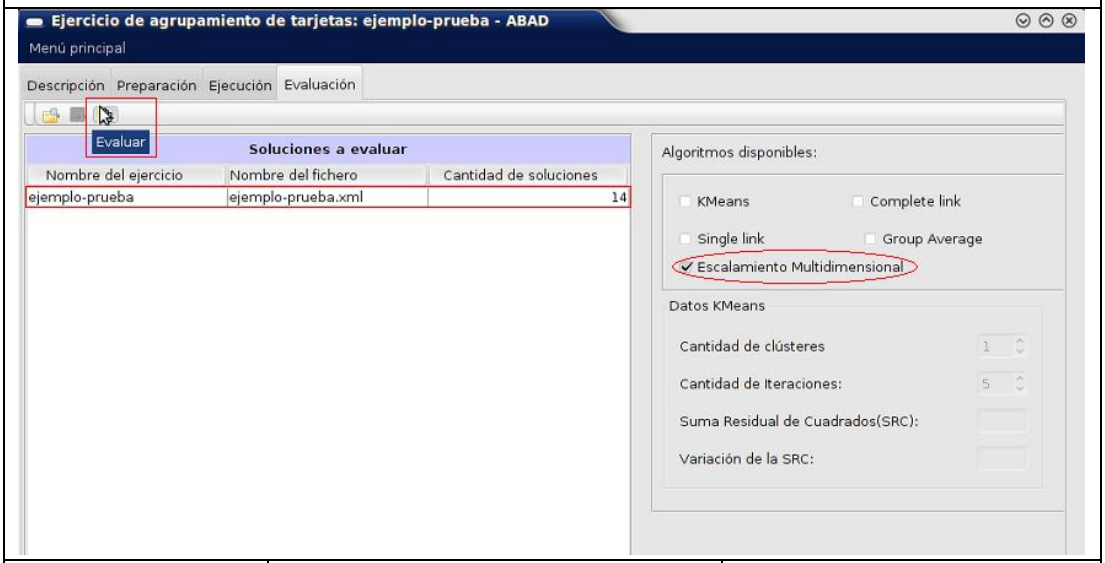
Para realizar la evaluación de un ejercicio el **AI** puede cargar un **Fichero xml** donde se encuentre recogido todo el proceso de agrupamiento, una vez cargado este fichero selecciona la ventana **Evaluación** y esta muestra los diferentes Algoritmos de evaluación con los que cuenta la herramienta, los cuales evalúan los grupos contenidos en el **Fichero xml** mostrando los **Resultados**, las gráficas correspondientes a cada algoritmo empleado.

2.5 Especificación de los requisitos funcionales y no funcionales

2.5.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, no alteran la funcionalidad del producto, por lo que se mantienen invariables sin importarle con que propiedades o cualidades se relacionan (SOMMERVILLE, 2006)

Tabla 3. Especificación de requisitos Seleccionar algoritmo de evaluación

<table>
<tr><th>Nº</th><th>Nombre</th><th>Descripción</th><th>Complejidad</th><th>Prioridad para cliente</th></tr>
<tr><td>[RF1.]</td><td>Seleccionar algoritmo de evaluación</td><td>Se selecciona el algoritmo escalamiento multidimensional en la pestaña evaluación, la solución que desea evaluar y se presiona el botón evaluar.</td><td>Alta</td><td>Baja</td></tr>
<tr><td></td><td colspan="4">Prototipo
 
</td></tr>
<tr><td></td><td>Campos</td><td>Tipos de Datos</td><td colspan="2">Reglas o Restricciones</td></tr>
</table>

		• *No procede*	• *No procede*

Tabla 4. Especificación de requisitos Obtener la matriz de coocurrencia

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
[RF2.]	Obtener la matriz de coocurrencia.	Dada las ocurrencias entre las tarjetas se construye la matriz de coocurrencias.	*Alta*	*Baja*
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
		• *No procede*	• *No procede*	

Tabla 5. Especificación de requisitos Obtener la matriz de distancias euclidianas

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
[RF3.]	Obtener la matriz de distancias euclidianas.	Se le calcula la matriz de distancias euclidianas a partir de la matriz de coocurrencias	*Alta*	*Baja*
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
		• *No procede*	• *No procede*	

Tabla 6. Especificación de requisitos Obtener la matriz de coordenadas

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
[RF4.]	Obtener la matriz de coordenadas	Dada la matriz de distancia euclidianas se realiza el escalamiento multidimensional no métrico y se obtiene una matriz con las coordenadas de cada tarjeta.	*Alta*	*Baja*
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
		• *No procede*	• *No procede*	

Tabla 7. Especificación de requisitos Obtener una imagen de la gráfica de soluciones

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
[RF5.]	Obtener una imagen de la gráfica de soluciones	Se genera una imagen de la gráfica resultante para ser mostrada como resultado así como para ser utilizada en el informe en el informe PDF.	*Alta*	*Baja*
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
		• *No procede*	• *No procede*	

Tabla 8. Especificación de requisitos Mostrar gráficamente las soluciones

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
[RF6.]	Mostrar gráficamente las soluciones	Se muestra una gráfica con los resultados obtenidos tras evaluar determinado ejercicio mediante la técnica de EMD	*Alta*	*Alta*
	Prototipo			

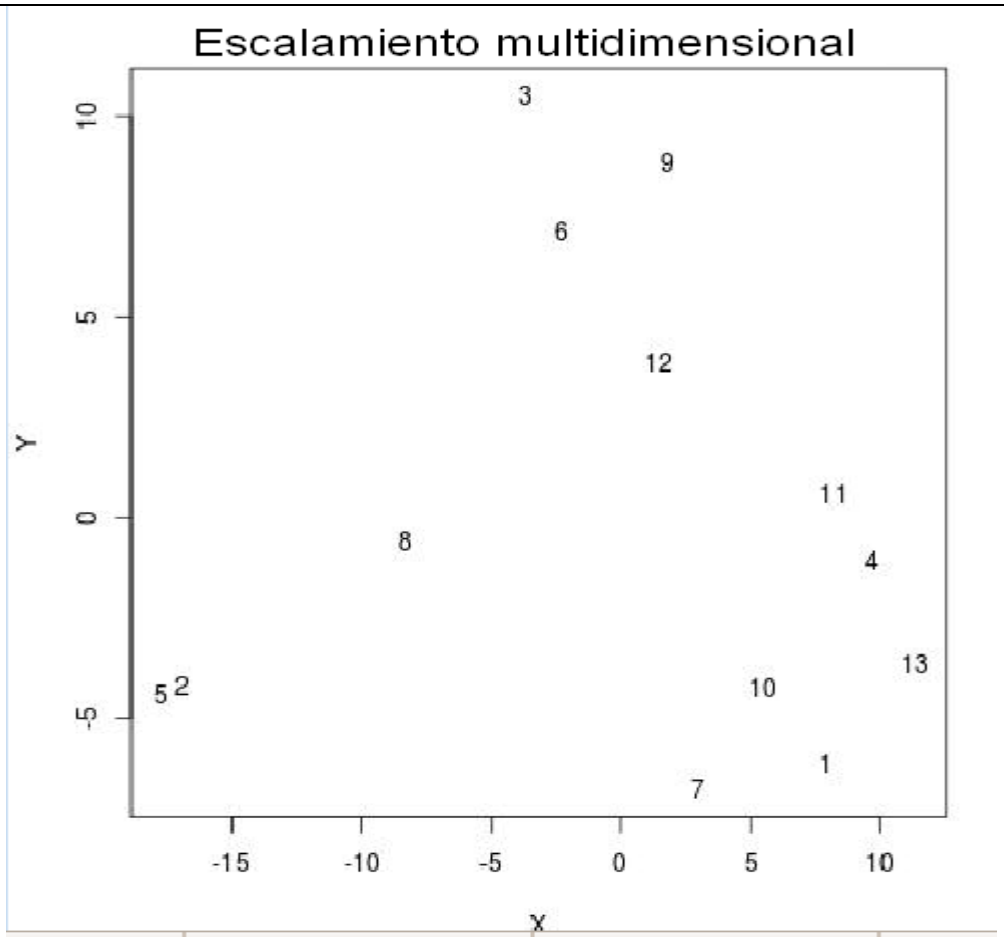
			
	Campos	**Tipos de Datos**	**Reglas o Restricciones**
		• *No procede*	• *No procede*

Tabla 9. Especificación de requisitos Mostrar una leyenda de los puntos mostrados en la gráfica

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
[RF7.]	Mostrar una leyenda de los puntos mostrados en la gráfica	Cada tarjeta es representada en la gráfica por un número y dicha relación (número - nombre de la tarjeta) es visualizada en la leyenda.	*Alta*	*Alta*
	Prototipo			

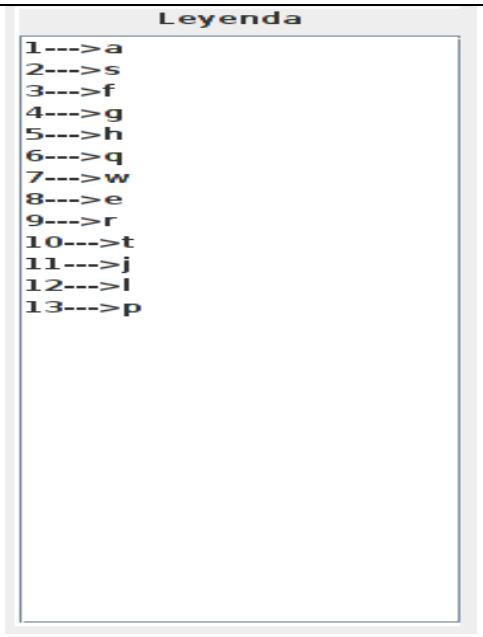
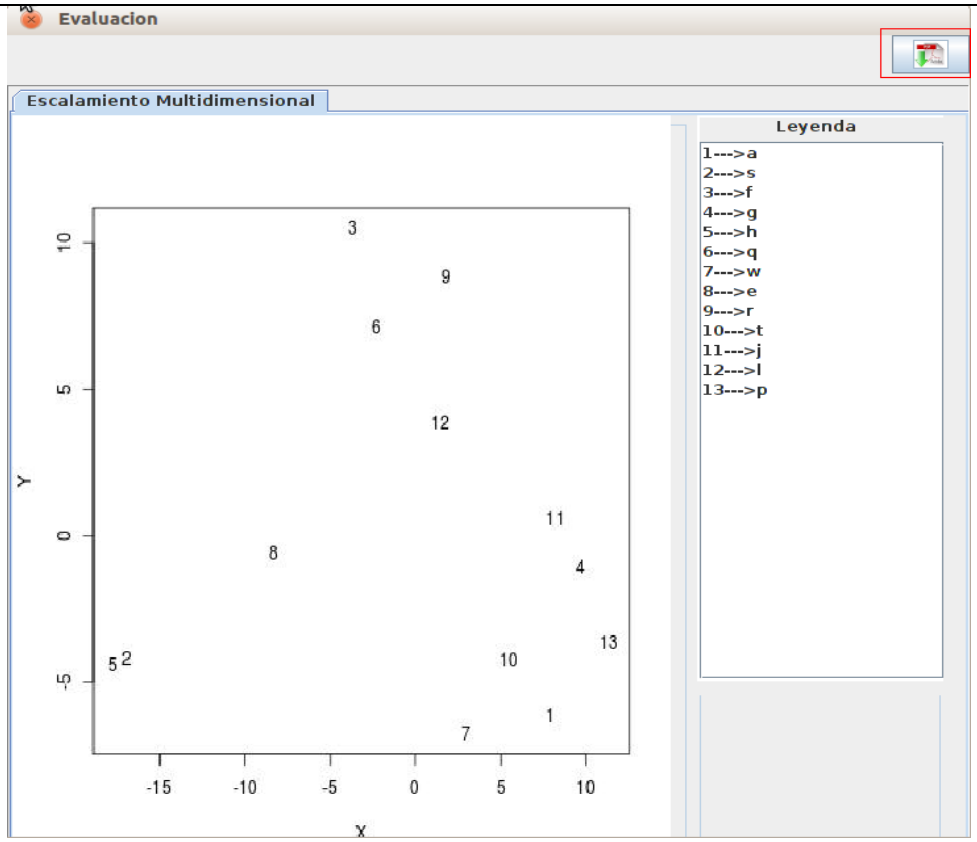
			
	Campos	**Tipos de Datos**	**Reglas o Restricciones**
		• *No procede*	• *No procede*

Tabla 10. Especificación de requisitos Generar un documento en formato pdf la gráfica y su leyenda

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
[RF8.]	Generar un documento en formato pdf la gráfica y su leyenda.	El arquitecto de información presiona el botón exportar a PDF y selecciona la dirección donde desea guardar dicho documento.	*Alta*	*Alta*
	Prototipo			



	Campos	Tipos de Datos	Reglas o Restricciones
		• *No procede*	• *No procede*

2.5.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características que hacen al producto atractivo, usable, rápido o confiable. Son fundamentales en el éxito del producto y normalmente están vinculados a requisitos funcionales (JACOBSON *et al.*, 2004).

Tabla 11. Requisitos no funcionales

Requisitos no funcionales	
Usabilidad	
1	Utilizar el idioma español para los mensajes y textos de la interfaz.

2	El sistema podrá ser utilizado por cualquier persona que posea conocimientos básicos de arquitectura de información
3	Diseño sencillo, que propicie el desarrollo de habilidades en el uso del sistema sin mucha inversión de tiempo de entrenamiento.
4	Menús: el módulo debe presentar una serie de menús, que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras
Escalabilidad	
5	El módulo debe estar en capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades después de su construcción y puesta en marcha inicial.
Restricciones de diseño	
6	El módulo se desarrollará utilizando "*Java"* como lenguaje de programación.
7	Como herramienta de desarrollo se utilizara el IDE NetBeans 7.0
Portabilidad	
8	El sistema será portable. Al menos podrá ser ejecutado tanto en Microsoft Windows y GNU - Linux.
Soporte	
9	La herramienta contará con un grupo de soporte y asesoría al cliente.
Legales	
10	Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de *software* libre.
Interfaz	
11	Interfaz externa: la interfaz deberá ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo de su empleo
12	El sistema debe contar con las interfaces de usuario diseñadas de la manera más intuitiva posible.
13	Los colores que deben predominar en el sistema deben tener un tono claro evitando el uso de colores fuertes que perjudiquen la visión de los usuarios.

14	Utilizará para los textos el tipo de fuente Arial con tamaño mínimo 8 y máximo 14.
15	Cada tarjeta será representada en la gráfica por un número y su correspondencia será vista en una leyenda
16	Debe predominar el uso de los íconos para la interacción con el usuario.
Hardware	
17	El sistema debe ejecutarse en un ordenador que tenga 256 MB de RAM como mínimo.
18	El sistema debe ejecutarse en un ordenador que tenga 100 MB libres en el disco duro como mínimo.
Software	
19	Debe estar instalada la Máquina Virtual de Java 1.5 o superior.
20	Debe estar instalado el *software* estadístico R
21	Debe estar instalado en el *software* estadístico R el paquete rJava

2.6 Estándares de codificación

En el caso de las declaraciones, para declarar una clase se utilizó la notación CamelCase o como también se le conoce UpperCamelCase o PascalCase consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en minúsculas. Ejemplo: ClaseNombreCompuesto.

Los métodos se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.

Las variables se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Ejemplo: variableNombreCompuesto.

Los comentarios se definen comenzando con los caracteres /* y terminando con ***/** Ejemplo: /* esto es un comentario */, siempre que se quiera comentar grandes instrucciones de códigos. Si solo se desea comentar una sola instrucción se utiliza //.

La identación[13] es a cuatro espacios en blanco. Entre los operadores aritméticos y sus operandos así como entre los operadores lógicos aritméticos y sus operandos se dejó un espacio en blanco. Las llaves para abrir y cerrar una clase, un método o un bloque de control de flujo se sitúan en una nueva línea y al mismo nivel del bloque al que pertenecen, ejemplo:

```
if.(TipoDato variableCondicion1 || TipoDato variableCondicion2)
{
.//BI
}
```

2.7 Arquitectura y patrones de diseño

"Una arquitectura es el sistema de decisiones significativas sobre la organización de un sistema de *software*, la selección de los elementos estructurales y de sus interfaces por los cuales el sistema es compuesto, junto con su comportamiento según lo especificado en las colaboraciones entre estos elementos, la composición de estos elementos estructurales y del comportamiento en subsistemas progresivamente más grandes y el estilo arquitectónico que dirigen esta organización, los elementos y sus interfaces, sus colaboraciones y su composición" (JACOBSON *et al.*, 2004).

2.7.1 Patrones de diseño

Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que brindan una estructura conocida para todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos, permiten tener una estructura de código común, ahorran grandes cantidades de tiempo en la construcción de un software, no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños son mucho más flexibles, modulares y reutilizables, además de dar una mejor imagen de profesionalidad y calidad. Existen varios tipos de patrones entre los que se encuentran los patrones estructurales, patrones de creación y los patrones de asignación de responsabilidad (LARMAN, 2004).

2.7.2 Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*, en español Patrones Generales de *Software* para la Asignación de Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones utilizados en la solución fueron:

[13] Mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente

- **Experto:** Se tiene en consideración a qué clase debe pertenecer un método, este principio sugiere que se asigne a la clase que más sepa del método (es decir al experto). Esto es una consecuencia del principio de alta cohesión, ya que si se asignan los métodos a las clases que tienen la información necesaria para ejecutarlos, se están creando clases altamente cohesionadas(LARMAN, 2004).
- **Bajo Acoplamiento:** Uno de los principios para proteger al *software* frente al cambio es mantener bajo el acoplamiento entre clases. El acoplamiento de una clase es el conjunto de dependencias que tiene con otras clases, cuanto menor sea el acoplamiento entre clases, menor influencia tendrán los cambios (LARMAN, 2004). La idea es tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- **Alta Cohesión:** Es el encargado de asignar una responsabilidad de modo que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (LARMAN, 2004).
- **Controlador:** ayuda a decidir quién se encarga de administrar un evento del sistema.
- **Creador:** trata el problema de qué clase debe ser la encargada de instanciar a otra.

Tabla 12. Ejemplo de los patrones

Patrón	Ejemplo de uso
Experto	La clase RJava es la encargada de realizar las evaluaciones de ejercicios de agrupamiento de tarjetas.
Creador	La clase EOT contiene a la clase Sol, es una agregación, la almacena, tiene sus datos de inicialización y la usa.
Bajo Acoplamiento	Se utiliza una sola clase para los algoritmos de evaluación para así reducir las dependencias entre clases
Alta Cohesión	La clase RJava realiza una labor única dentro del sistema, y ninguna otra clase realiza sus funciones.
Controlador	El proceso de evaluación se centraliza en la clase ControladorEval

2.7.3 Descripción de la arquitectura

La herramienta de agrupamiento de tarjetas a la cual se integra esta propuesta de solución está basada en la arquitectura tres capas como se muestra en la figura 4:

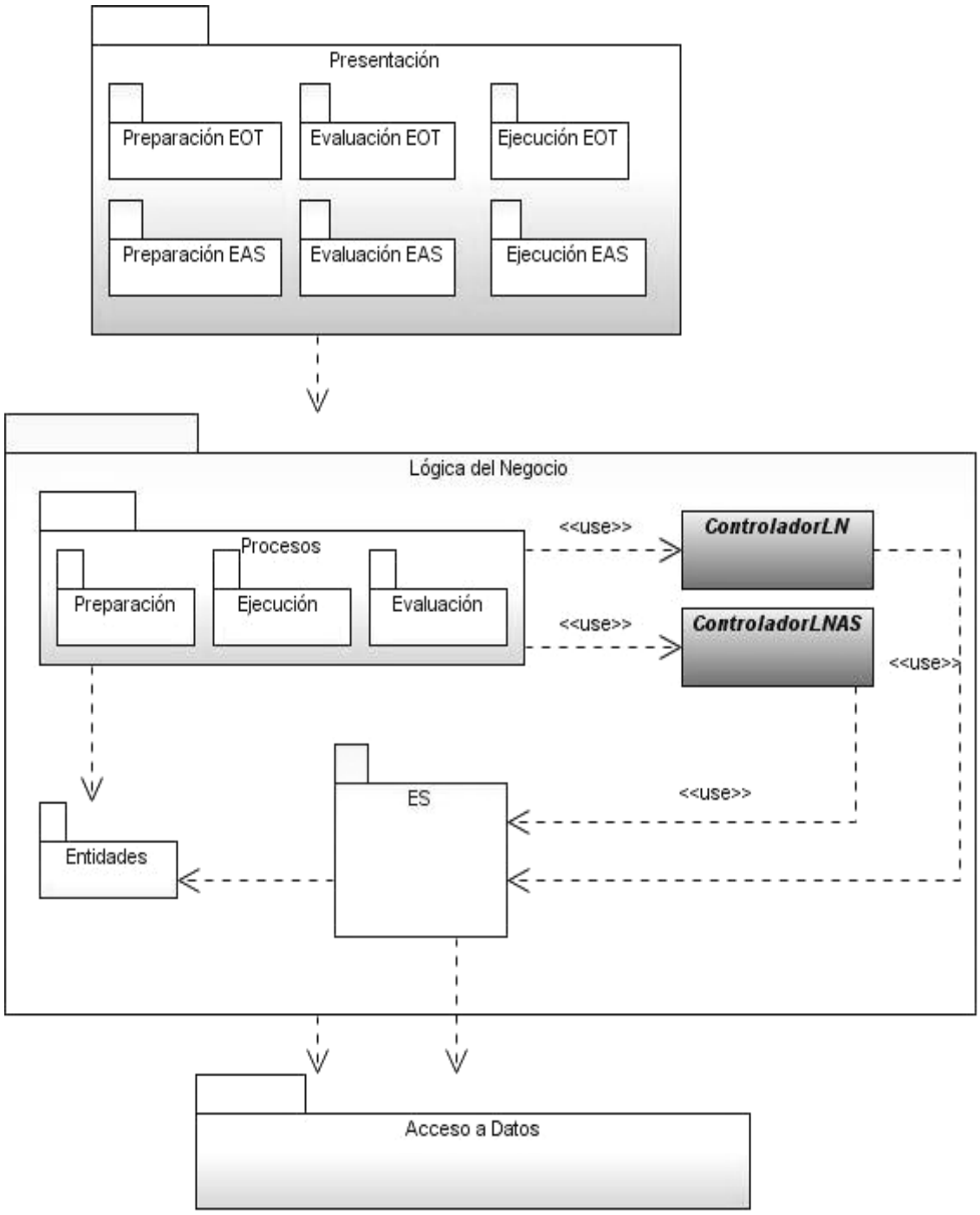


Figura 4. Arquitectura de la herramienta Agrupamiento de tarjetas y Análisis de secuencia

El módulo propuesto se desarrolla bajo este mismo concepto, a continuación se describe cada capa:

Capa presentación: la capa de presentación es la encargada de lograr la comunicación directa entre el sistema y el usuario final a través de una interfaz gráfica. Una interfaz gráfica de usuario consiste en

un conjunto de componentes empleados por usuarios para comunicarse con los sistemas informáticos. Los usuarios dirigen el funcionamiento del sistema mediante la generación de eventos. En la herramienta de agrupamiento de tarjetas, la capa presentación está dividida en paquetes según las funcionalidades que brinde cada uno para lograr una mayor organización. Los paquetes empleados son PreparaciónEOT, EjecuciónEOT, EvaluaciónEOT, PreparaciónAS, EjecuciónAS, EvaluaciónAS y el paquete Elementos Genéricos (figura 5).

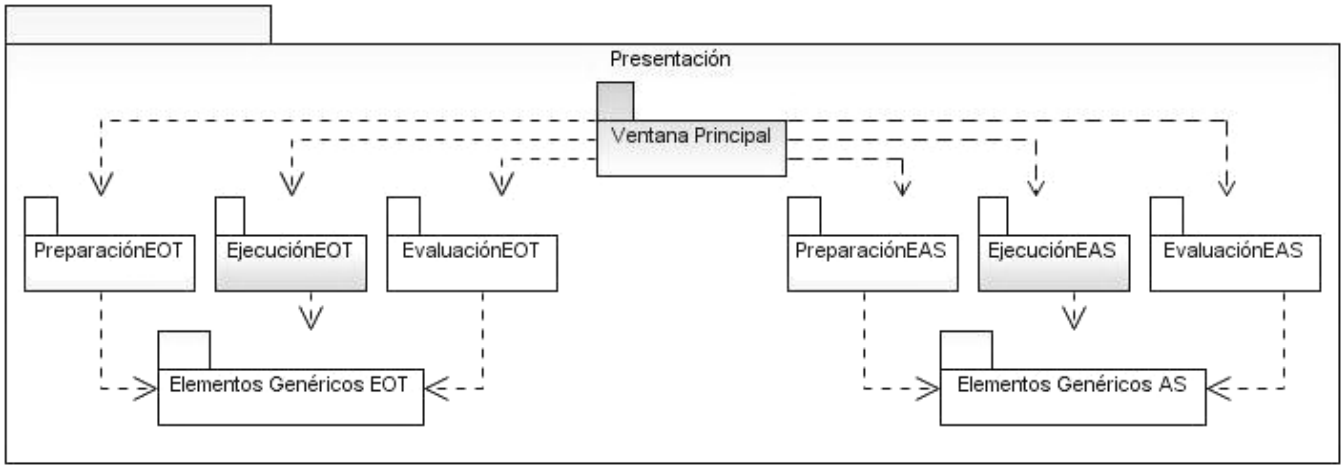


Figura 5. Capa presentación: Paquetes

En el caso de módulo desarrollado se operó sobre el paquete EvaluaciónEOT, que es el encargado de gestionar el proceso de evaluación de un ejercicio de agrupamiento de tarjetas, este módulo permite:

- seleccionar el tipo de evaluación que el arquitecto desea utilizar
- visualizar los resultados obtenidos en una gráfica así como la leyenda de los puntos representados en la misma
- generar un informe en formato PDF con la gráfica obtenida, así como su leyenda (ver anexo).

Esta capa presentación se comunica únicamente con la capa lógica del negocio.

Capa lógica del negocio: La capa lógica del negocio, llamada también como lógica de la aplicación, se encarga de realizar las tareas para las cuales está concebido el sistema. Es implementada utilizando un modelo orientado a objetos del dominio de la aplicación. Se encarga de controlar las operaciones de acuerdo con las reglas del negocio.

En esta herramienta el paquete de lógica del negocio está dividida en tres paquetes: Procesos (figura 6) (figura 7) (figura 8), Entidades (figura 9) y Entrada Salida (ES) (figura 10).

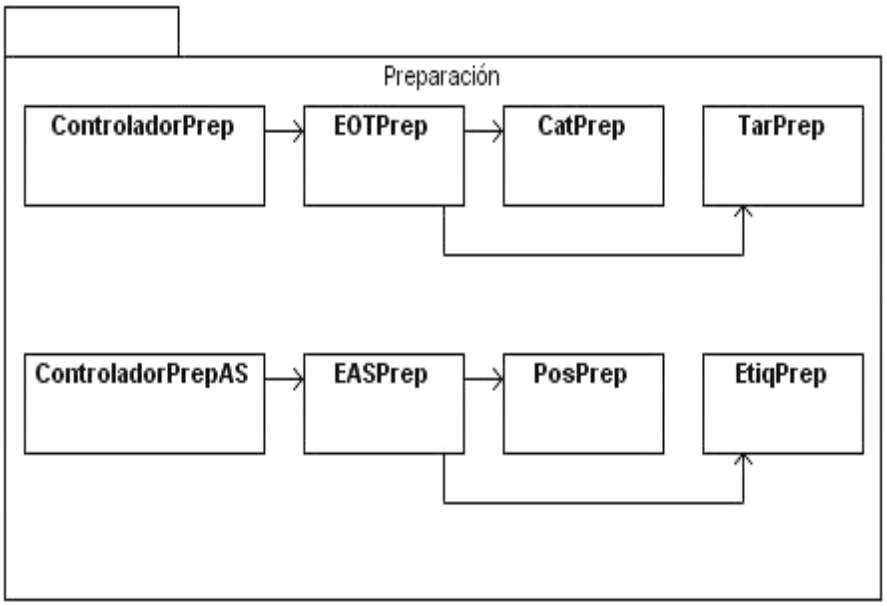


Figura 6. Capa lógica del negocio proceso preparación

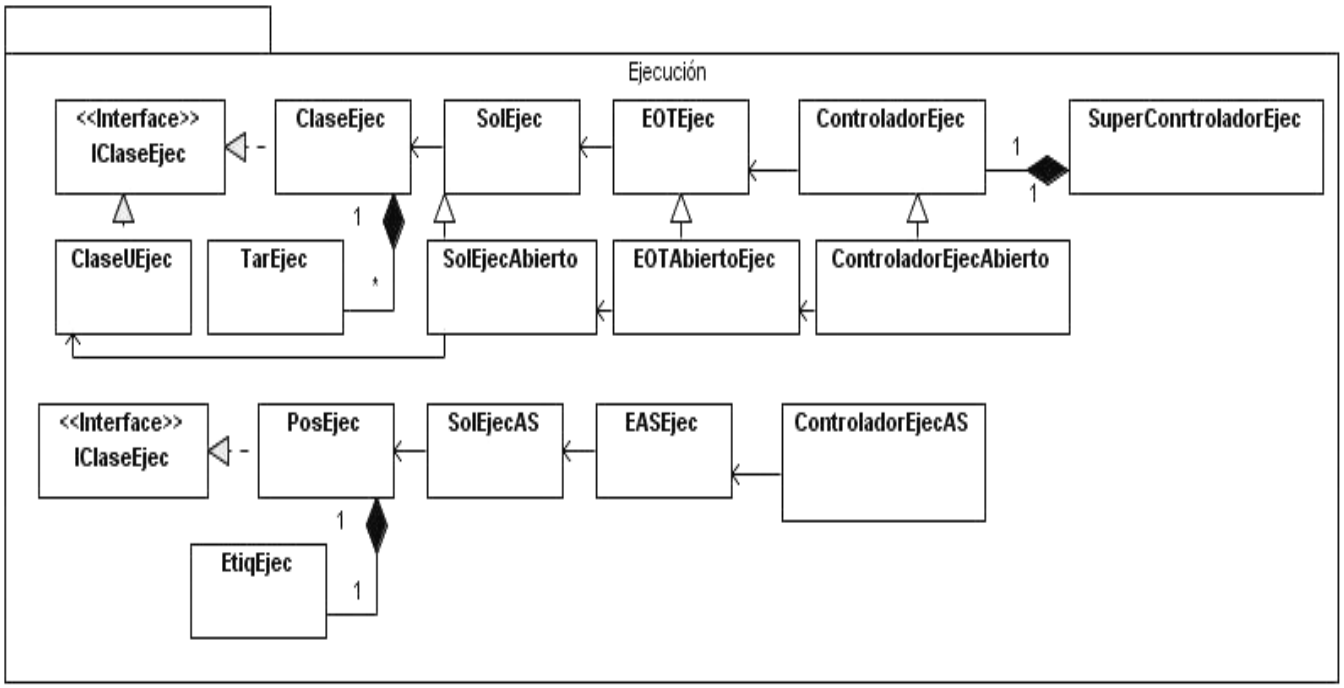


Figura 7. Capa lógica del negocio proceso ejecución

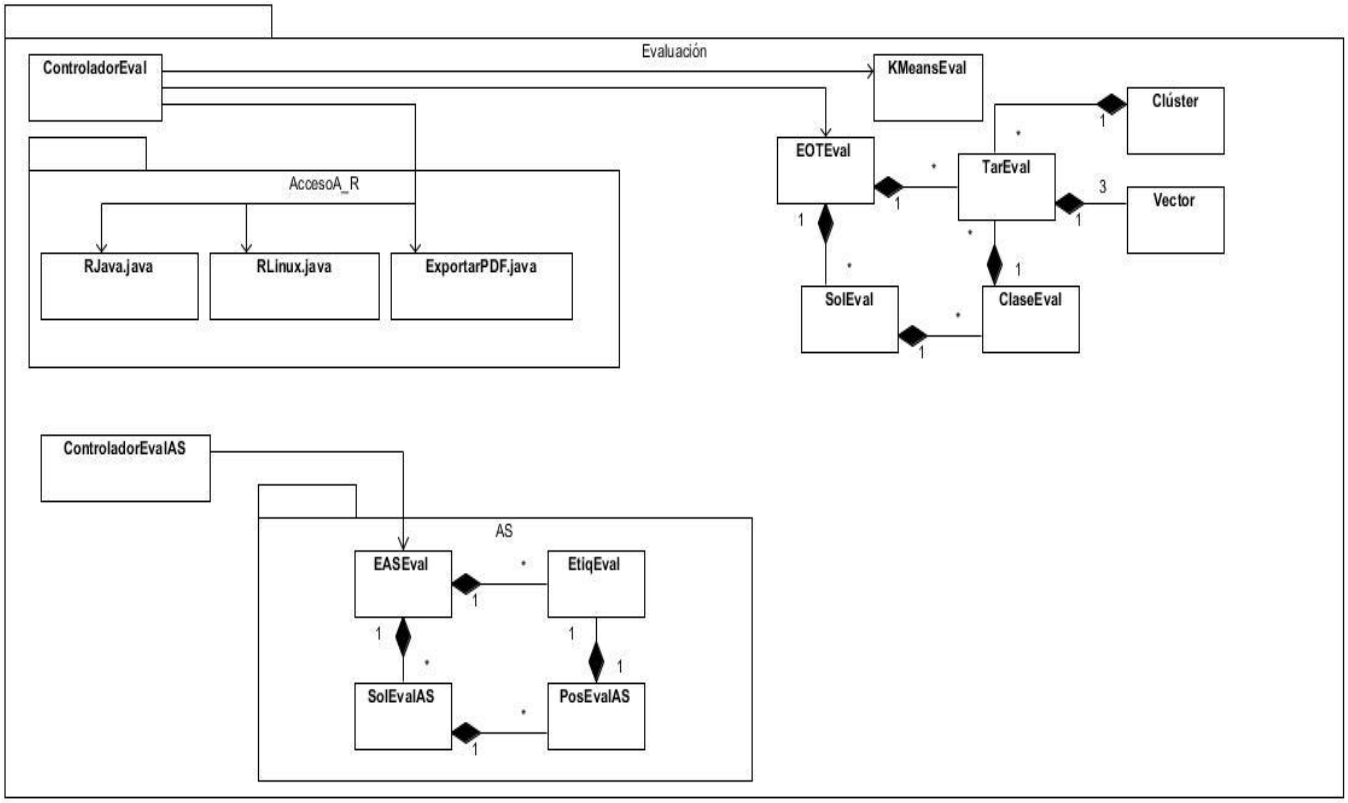


Figura 8. Capa lógica del negocio proceso evaluación

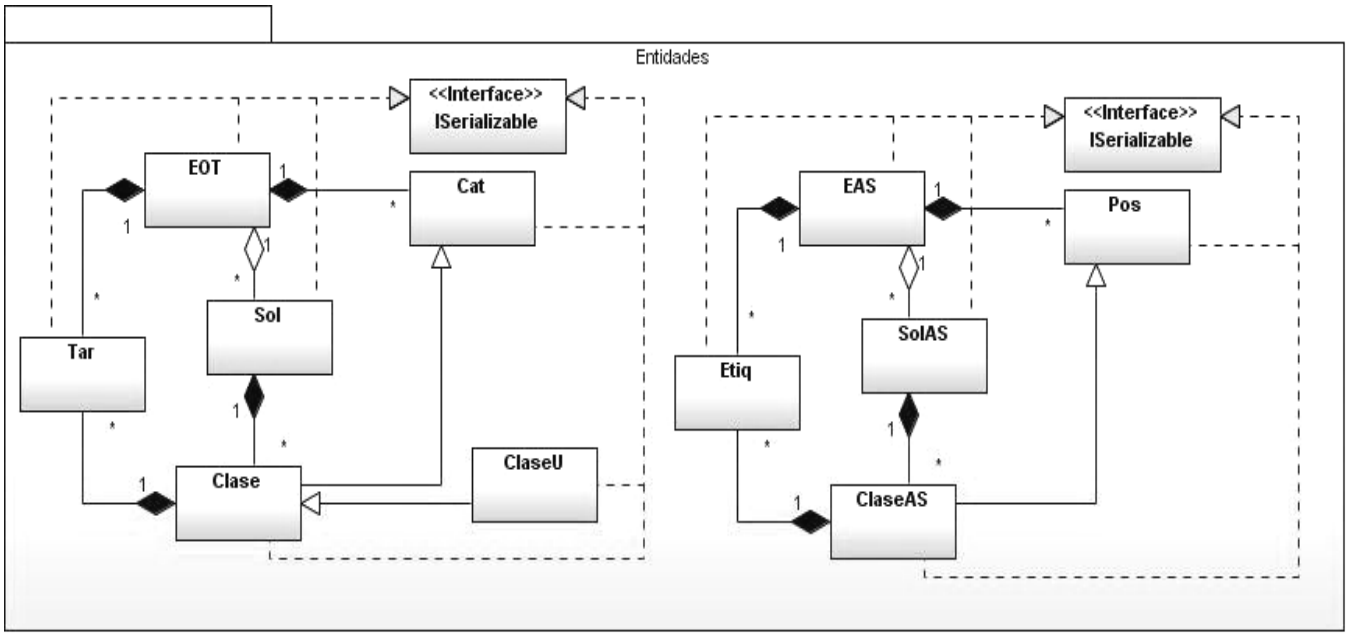


Figura 9. Capa lógica del negocio. Entidades

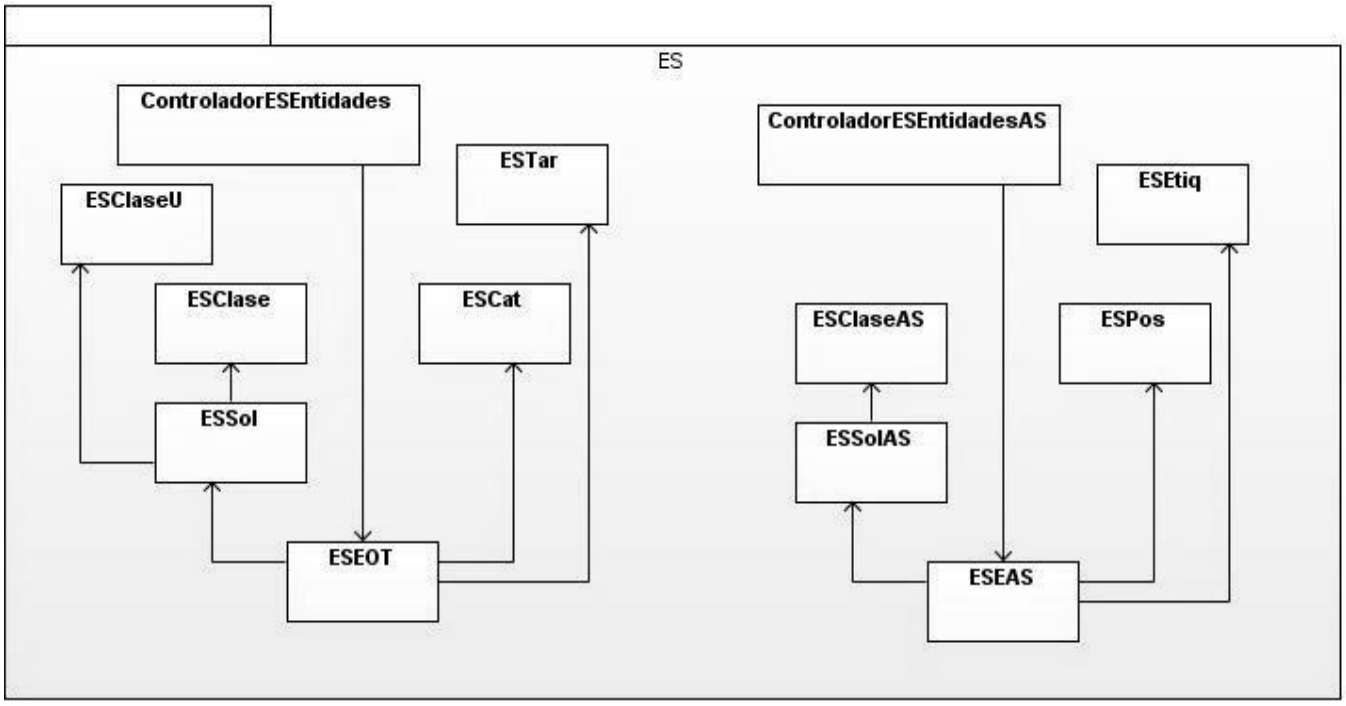


Figura 10. Capa lógica del negocio. Entrada y salida

Para realizar el proceso de evaluación ya sea para la técnica propuesta o para aplicar cualquier otro algoritmo de evaluación con los que cuenta la herramienta, esta capa se comunica con la capa acceso a datos solicitando la matriz de coocurrencias entre tarjetas para aplicarle las técnicas de evaluación.

Capa acceso a datos: La capa de acceso a datos es la que gestiona el almacenamiento de los datos, ya sea en una base de datos o en un fichero, así como la consulta a los mismos. Es la encargada de la persistencia y recuperación de objetos, específicamente de la interacción de la aplicación con los ficheros de almacenamiento de ejercicios de ordenamiento de tarjetas y del análisis de secuencia. Estos ficheros pueden contener ejercicios de agrupamiento de tarjetas y ejercicios de análisis de secuencia en forma de objeto serializado o de documento XML[14]. El paquete de acceso a datos mantiene un bajo acoplamiento con las entidades del negocio. Las clases de este paquete desconocen a las entidades del negocio y solo se centran en la entrada y salida de datos como se muestra en la (figura 11).

[14] XML: (de sus siglas en inglés de *e**X**tensible **M**arkup **L**anguage):* Lenguaje de marcas extensibles, es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C), el cual brinda soporte a bases de datos, siendo útil cuándo varias aplicaciones se deben comunicar entre sí o integrar información.

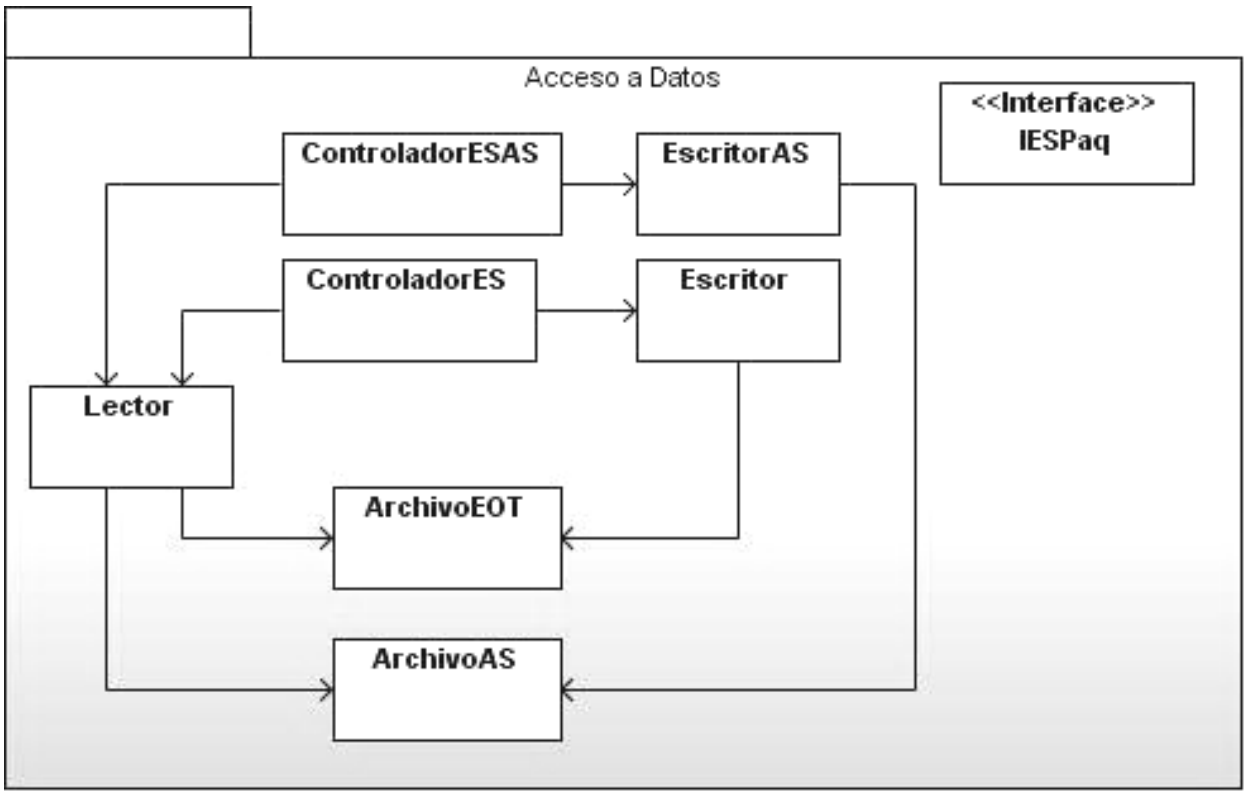


Figura 11. Capa de acceso a datos

Conclusiones parciales

En este capítulo se desarrolló la propuesta de solución para el proceso de evaluación de la herramienta de agrupamiento de tarjetas atendiendo a las características que debe tener el mismo. A partir del modelo del dominio propuesto y las propias necesidades del cliente, se realizó la especificación de los requisitos funcionales y no funcionales. Se describe la arquitectura a utilizar, así como los patrones de diseño que serán empleados durante todo el desarrollo del módulo, cada uno de estos elementos darán paso y contribuirán a una correcta implementación del mismo. Cada uno de estos elementos contribuyó a una correcta implementación del módulo. La integración del mismo con el *software* estadístico R permitirá ahorrar tiempo en la implementación de la solución así como una mayor fiabilidad de los resultados. Luego de realizadas estas tareas se llega a la conclusión que el sistema se encuentra apto para que se le comiencen a realizar las pruebas.

Capítulo 3. Validación de la solución

El desarrollo de sistemas de *software* implica una serie de actividades de producción en las que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde el inicio del proceso, así como dentro los pasos del diseño y desarrollo posteriores. El desarrollo de *software* ha de ir acompañado de una actividad que garantice la calidad. La prueba del *software* es un elemento crítico para la garantía de calidad del *software* y representa una revisión final de las especificaciones, del diseño y de la codificación.En este capítulo se aborda el tema referente a la realización de las pruebas, se realiza un análisis del tipo de prueba a utilizar para validar la solución y el diseño de casos de prueba.

3.1 Validación de los requisitos

Un elemento clave de cualquier proceso de ingeniería es la medición. Se emplean medidas para entender mejor los atributos de los modelos que se crean. Pero, fundamentalmente, se emplean las medidas para valorar la calidad de los productos de ingeniería o de los sistemas que son construidos (PRESSMAN, 2005).Durante la investigación se obtuvieron numerosos artefactos, producto de la metodología de *software* usada. Para validar estos artefactos, se aplican distintas métricas en pos de verificar la calidad y efectividad de los mismos.

3.1.1 Métrica para evaluar los requisitos

Para realizar la validación de los requisitos existe toda una lista de características que sugieren el uso de una o varias métricas como son: especificidad (ausencia de ambigüedad), corrección, compleción, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización (PRESSMAN, 2005).

Es importante tener en cuenta que para medir las características de la especificación, es necesario conseguir profundizar cuantitativamente en la especificidad y en la completitud.

Especificidad

Para llevar a cabo este proceso se tiene que: $n_r$representa el número de requisitos del sistema.

$$n_r = n_f + n_{nf}$$

$$n_r = 8 + 21$$

$$n_r = 29$$

Donde n_f es el número de requisitos funcionales y n_{nf} es el número de requisitos no funcionales. Para determinar la especificidad (ausencia de ambigüedad) de los requisitos se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito.

$Q = n_{ui}/n_r$

Donde n_{ui} es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. El valor de Q a medida que se acerca a 1, se va disminuyendo la ambigüedad de la especificación.

Con el objetivo de obtener la menor ambigüedad posible, para que los requisitos tengan una mayor claridad de modo que satisfagan las necesidades de cliente, se llevaron a cabo 2 revisiones con 3 especialistas del proyecto.

A continuación se presentan los especialistas del proyecto que llevaron a cabo la inspección:

Tabla 13. Especialistas del proyecto

Nombre y apellidos	Rol desempeñado
Yanicet Aveleira Rodríguez	Jefe de proyecto
Sergio René Vázquez	Jefe de línea
Gleydis María Rodríguez Ramírez	Analista

En la primera revisión se identificaron algunos requisitos funcionales y no funcionales que presentaban problemas en la redacción y de ambigüedad. Para un total de 29 requisitos, los revisores tuvieron la misma interpretación para 22 de ellos.

$$Q = 22/29$$
$$Q = 0.75$$

En la segunda revisión, ya corregidos los errores identificados en la primera, los revisores tuvieron la misma interpretación para el total de requisitos.

$$Q = 29/29$$
$$Q = 1$$

Figura 12. Resultado de la métrica para la calidad de la especificación. Especificidad

Al concluir el estudio cuantitativo realizado a los resultados obtenidos en cada una de las revisiones, se evidencia que los requisitos presentan un grado bajo de ambigüedad.

Compleción

La aplicación de esta métrica siempre devuelve un resultado entre 0 y 1. Mientras más cercano a 1 se encuentre el resultado, indica un alto nivel de completitud en la definición de los requisitos. Este valor se calcula de la siguiente forma:

$$Q_1 = n_a/(n_a + n_b)$$

n_a: número de requisitos completos,

n_b: número de requisitos pobremente especificados.

Para la aplicación de esta métrica se procedió de igual forma que la anterior, arrojando los siguientes resultados:

En la primera revisión:

$Q_1 = 20/29$

$Q_1 = 0.68$

En la segunda revisión:

$Q_1 = 27/29$

$Q_1 = 0.93$

Figura 13. Resultado de la métrica para la calidad de la especificación. Compleción

Culminada las revisiones los datos arrojados muestran que la mayoría de los requisitos están completamente especificados.

Según los resultados obtenidos con la aplicación de las métricas de Especificidad y Compleción para evaluar la calidad de la Especificación de Requisitos, se evidencia que los mismos están completamente especificados y son de claro entendimiento por el lector.

La siguiente gráfica muestra lo mencionado anteriormente.

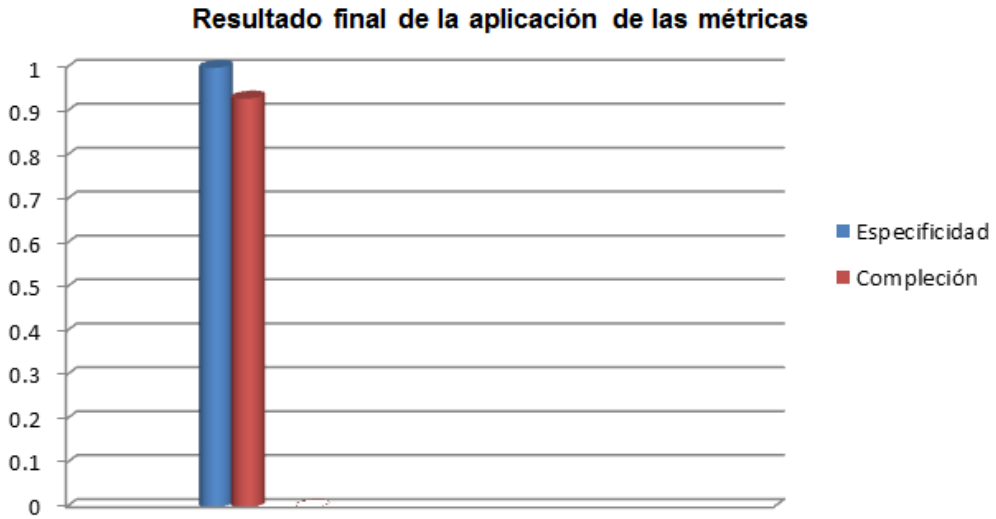


Figura 14. Resultado final de la aplicación de las métricas

3.2 Pruebas

Las pruebas de *software* consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. La prueba es un proceso que se enfoca sobre la lógica interna del *software* y las funciones externas. Es un proceso de ejecución de un programa con la intención de descubrir un error, no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el *software (SLIDESHARE, 2009)*.

3.2.1 Objetivos de las pruebas de software

La prueba de *software* es un elemento crítico para la garantía del correcto funcionamiento del *software*. Entre sus objetivos están:

1. Detectar defectos en el *software*.
2. Verificar la integración adecuada de los componentes.
3. Verificar que todos los requisitos se han implementado correctamente.
4. Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el *software* al cliente.
5. Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

3.2.2 Métodos de pruebas

- **Caja blanca:** La prueba de la caja blanca del *software* comprueba los caminos lógicos del *software* proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado (PRESSMAN, 2005).

- **Caja negra**: La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*. Este tipo de prueba se centra en los requisitos funcionales del *software* y permite obtener entradas que prueben todos los requisitos funcionales del programa.Los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (PRESSMAN, 2005).

3.2.3 Prueba de integración

"La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño" (PRESSMAN, 2005).

La necesidad de realizar las pruebas de integración viene dada por el hecho de que los módulos que forman un programa suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual. Con el uso de estas pruebas se consigue ir formando el programa global a medida que se comprueba como los distintos componentes interaccionan y se comunican libres de errores.

Su objetivo es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. Se diseñan para descubrir errores o completitud en las especificaciones de las interfaces.

En este nivel se asegura que las interfaces y ligas entre las partes del sistema trabajen apropiadamente. Antes de las pruebas de integración, los componentes tuvieron que haber pasado sus pruebas individuales, por lo que el enfoque ahora es sobre el flujo de control entre los módulos, y sobre los datos que son intercambiados entre ellos de manera independiente.

Las pruebas de integración se realizaron para garantizar que el módulo incluido en el proceso de evaluación funcionara correctamente cuando se combinara con la herramienta para ser ejecutado. Las pruebas de integración no arrojaron estados incompletos ni errores en las especificaciones de la interfaz del paquete.

Las tablas siguientes muestran estos resultados después de realizar una iteración de este tipo de prueba en interacción con la herramienta de agrupamiento de tarjetas:

Tabla 14. Prueba de Integración: *Software* estadístico R

Caso de Prueba
Número de caso de prueba: 1
Subsistema a integrar: *Software* estadístico R
Condiciones de ejecución: El *software* estadístico R esté instalado así como el paquete rJava.
Descripción de la prueba: Comprobar que el módulo Escalamiento multidimensional no métrico interactúa con el R.

Entradas/Pasos de ejecución: El módulo Escalamiento multidimensional no métrico introduce la matriz de coocurrencia al R y le realiza las siguientes peticiones: calcular la matriz de distancia euclidiana, llamar a la librería MASS, realizar el EMD-NM mediante la función isoMDS, pasándole la matriz de distancia euclidiana obtenida anteriormente y exportar en una imagen la gráfica con los resultados obtenidos.
Resultado esperado: Se obtiene una imagen con la gráfica de puntos.
Evaluación: Prueba satisfactoria.

Tabla 15. Prueba de Integración: Escalamiento multidimensional no métrico

Caso de Prueba
Número de caso de prueba: 2
Módulo a integrar: Escalamiento multidimensional no métrico
Condiciones de ejecución: El módulo Escalamiento multidimensional genere la imagen con la gráfica de puntos, dada una matriz de coocurrencias
Descripción de la prueba: Comprobar que la herramienta de agrupamiento de tarjetas muestre correctamente la imagen creada por el módulo Escalamiento multidimensional no métrico.
Entradas/Pasos de ejecución: El módulo Escalamiento multidimensional no métrico recibe la matriz de coocurrencias generada por la herramienta de agrupamiento de tarjetas y realiza el proceso de escalamiento multidimensional no métrico generando la imagen con la gráfica de puntos, la cual es mostrada por la herramienta de agrupamiento de tarjetas.
Resultado esperado: La herramienta de agrupamiento de tarjetas funciona correctamente
Evaluación: Prueba satisfactoria.

3.2.4 Prueba de aceptación

Las pruebas de aceptación constituyen pruebas de caja negra que se centran en el cumplimiento de los requisitos definidos para el sistema una vez concluido. El objetivo de las pruebas de aceptación es validar que un sistema cumpla con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento (PRESSMAN, 2005). Son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario. Es muy recomendable que las realicen en el entorno en que se va a explotar el sistema incluyendo el personal que lo va a manejar.

Para la realización de este tipo de prueba se escogieron usuarios con conocimientos básicos de arquitectura de información, los tipos de pruebas que se realizaron fueron:

- **Prueba Alfa:** Se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el *software* de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un

entorno controlado. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.

- **Prueba Beta:** Se llevan a cabo por los usuarios finales del *software* en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente, así la prueba beta es una aplicación en vivo del *software* en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de los problemas informados, el desarrollador del *software* lleva a cabo modificaciones y así prepara una versión del producto de *software* para toda la clase de clientes.

Casos de pruebas:

Tabla 16. Caso de prueba: Evaluar

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
El usuario estando en el escenario evaluación selecciona el algoritmo Escalamiento multidimensional, el fichero a evaluar y presiona el botón evaluar	Se muestra una ventana con la gráfica de puntos así como la leyenda	Positivo	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
El usuario estando en el escenario evaluación no selecciona ningún algoritmo de evaluación	El sistema muestra el mensaje “Debe seleccionar al menos un algoritmo de evaluación”	Positivo	
El usuario estando en el escenario evaluación selecciona el algoritmo de evaluación pero no hay	El sistema muestra el mensaje “No hay soluciones a evaluar”	Positivo	

fichero a evaluar			
Evaluación de la prueba	Satisfactoria		

Tabla 17. Caso de prueba: Actualizar resultados

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
El usuario estando en el escenario donde se visualizan los resultados, cierra el mismo, va al escenario de ejecución modifica los agrupamientos y vuelve a realizar el proceso de evaluación	Se muestra un gráfico con los nuevos resultados obtenidos así como leyenda	Positivo	
Evaluación de la prueba	Satisfactoria		

Tabla 18. Caso de prueba: Exportar PDF

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
El usuario estando en el escenario donde se visualizan los resultados presiona el botón exportar PDF, selecciona la ruta donde desea guardar el mismo, le pone un nombre y presiona el botón guardar	Se genera un informe en formato PDF con los gráficos contenidos en escenario de los resultados así como la leyenda correspondiente a cada gráfico	**Positivo**	

Evaluación de la prueba	Satisfactoria

Resultados de las pruebas de aceptación

Como resultado de las pruebas se obtuvieron un conjunto de no conformidades las cuales fueron solucionándose en cada iteración realizada, lográndose un producto que cumple con los requisitos definidos. En la figura se pueden apreciar los resultados obtenidos en cada iteración.

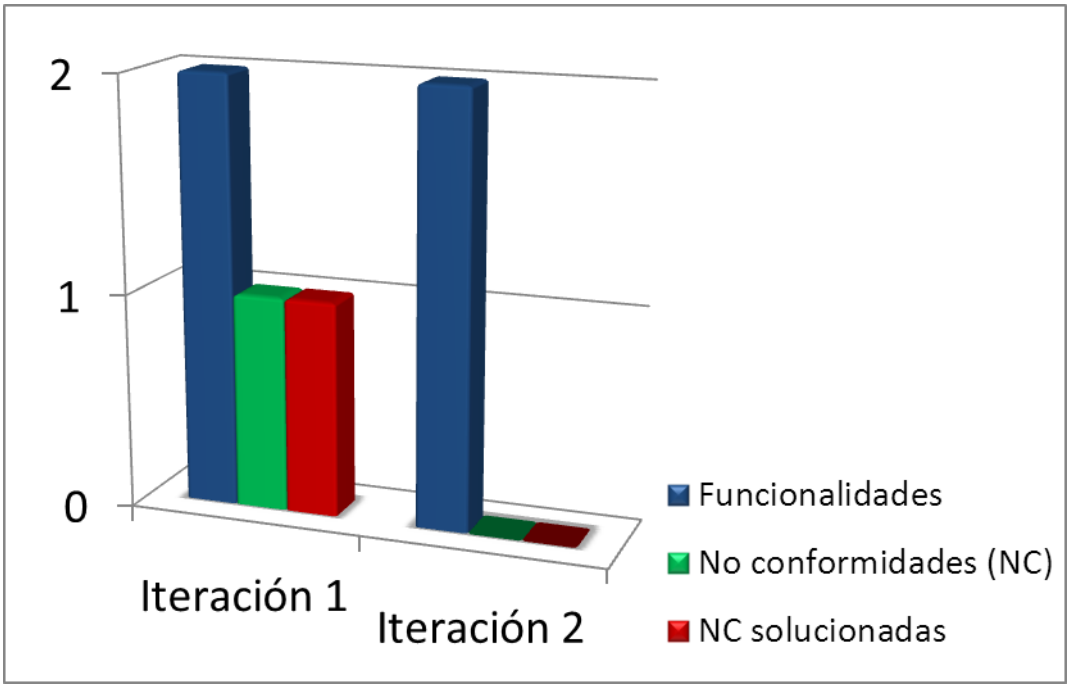


Figura 15. Resultados de la pruebas de aceptación por iteración

Se realizaron 2 iteraciones con el objetivo de identificar y solucionar los posibles errores existentes. En la primera iteración de un total de 2 funcionalidades analizadas, se detectó 1 no conformidad de funcionamiento interno las cual fue solucionada en un corto período de tiempo. En la segunda iteración se probaron nuevamente las 2 funcionalidades y no de detectaron no conformidades lo que demuestra que el módulo funciona según se previó.

Conclusiones parciales:

Con el uso de la métrica para la calidad de los requisitos y las técnicas de validación de los mismos se pudo demostrar que los requisitos definidos en el módulo están completos, son exactos y consistentes. Las pruebas de integración y aceptación fueron de gran importancia en la garantía de la calidad del

módulo, ya que permitieron identificar errores que atentaban contra el correcto funcionamiento de la herramienta de agrupamiento de tarjetas. Una vez realizadas las pruebas necesarias y solucionados los errores identificados se concluyó que el módulo desarrollado en la herramienta de agrupamiento de tarjetas así como la herramienta en su totalidad presenta buena calidad.

Conclusiones Generales

En el presente trabajo se arribó a las siguientes conclusiones:

- ✓ El estudio de soluciones homólogas demostró la necesidad de implementar un módulo que permitiera realizar evaluaciones por el método escalamiento multidimensional.
- ✓ Con el estudio de los *software* estadístico se logró dar solución al problema planteado.
- ✓ Por la complejidad matemática del escalamiento multidimensional se concluyó que era más factible la integración con el *software* R que su implementación.
- ✓ La solución implementada brinda una visión global de los resultados y que es un excelente complemento para los métodos de agrupamiento jerárquicos.
- ✓ La integración con el *software* estadístico R facilitó realizar el escalamiento multidimensional no métrico de forma rápida y eficiente, así como brindarle a los arquitectos de información otra vía de evaluación para representar los resultados.

Recomendaciones

Al finalizar la investigación y cumplir el objetivo general de la misma, con el fin de mejorar el proceso de evaluación de la herramienta de agrupamiento de tarjetas se recomienda:

- Implementar una funcionalidad que permita al arquitecto de información crear grupos manualmente en la gráfica generada por la técnica escalamiento multidimensional.
- Implementar una funcionalidad que agrupe de forma automática las tarjetas más semejantes representadas en la gráfica generada por el escalamiento multidimensional.

Bibliografía referenciada

1. ACOSTA, A. y ZAMBRANO, N. IMPORTANCIA, PROBLEMAS Y SOLUCIONES EN EL DISEÑO DE LA INTERFAZ DE USUARIO. 2006, vol. 18, nº p. 9. Disponible en: http://ri.biblioteca.udo.edu.ve/bitstream/123456789/1284/1/Importancia%2cproblemas%20y%20soluciones%20en%20el%20dise%C3%B1o%20de%20la%20int.pdf. ISSN 1316-6239.
2. CALISOFT. *Proceso de Mejora* Disponible en: http://calisoft.uci.cu/index.php/proceso-de-mejora.
3. CANÓS, J.; LETELIER, P.*, et al.* Metodologías Ágiles en el desarrollo de Software. *Universidad Politécnica de Valencia, Valencia*, 2003, nº Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm. ISSN ISSN 1666-1680.
4. CAÑAS, J. J.; ANTOLÍ, A.*, et al.* REPRESENTACIÓN MENTAL DE LOS CONCEPTOS, OBJETOS Y PERSONAS IMPLICADOS EN UNA TAREA REALIZADA EN UNA INTERFAZ Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, verano, año/vol. 6, número 016 Asociación Española para la Inteligencia Artificial. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 2002, nº 16, p. 107-113.
5. CARDSWORD. Disponible en: http://sourceforge.net/projects/cardsword/.
6. CASAS, F. M. G. y HURTADO, J. M. R. EL ANÁLISIS DE ESCALAMIENTO MULTIDIMENSIONAL: UNA ALTERNATIVA Y UN COMPLEMENTO A OTRAS TÉCNICAS MULTIVARIANTES. 2002, nº
7. CICA, C. I. C. D. A. *R* Disponible en: http://www.cica.es/Software/r.html.
8. CHRISSIS, M. B.; KONRAD, M.*, et al. CMMI: Guía para la integración de procesos y mantenimiento de productos.* Pearson, 2009. ISBN 8478290966.
9. DEFINICION.DE. *Definición de modelo de calidad* Disponible en: http://definicion.de/modelo-de-calidad/.
10. ECURED. *Card Sorting* Cuba: Disponible en: http://www.ecured.cu/index.php/Card_sorting.
11. EUTIO. *Entornos de Desarrollo Integrado* Disponible en: http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.
12. GARCÍA, J. *CMMI–CMMI nivel 2.* España, publicado el: 14 de Agosto de 2005 de 2005, última actualización: 14 de Agosto de 2005. Disponible en: http://www.ingenierosoftware.com/calidad/cmm-cmmi.php.
13. GARRETT, J. J. *The Elements of User Experience.* New York: New Riders Publishing, 2002, En New Riders Publishing.

14. GRUPO-SATÉLITE-SA. *Grupo Satélite sa* Disponible en: http://www.gruposatelite.net/index.php?option=com_content&view=article&id=66&Itemid=69.
15. HASSAN-MONTERO, Y. y MARTÍN-FERNÁNDEZ, F. J. *Card Sorting: Técnica de categorización de contenidos.* 2004, Disponible en: http://www.nosolousabilidad.com/articulos/cardsorting.htm.
16. HASSAN-MONTERO, Y.; MARTÍN-FERNÁNDEZ, F. J.*, et al. Arquitectura de la información en los entornos virtuales de aprendizaje. Aplicación de la técnica card sorting y análisis cuantitativo de los resultados.* 2004.
17. HASSAN-MONTERO, Y. y ORTEGA-SANTAMARÍA, S. *Informe APEI sobre usabilidad.* Editado por: Lavandera-Fernández, R. APEI, Asociación Profesional de Especialistas en Información, 2009.
18. HASSAN, Y.; MARTÍN FERNÁNDEZ, F. J.*, et al.* Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información. *Hipertext.net*, 2004, nº 2,
19. JACOBSON, I.; BOOCH, G.*, et al. El Proceso Unificado de Desarrollo de Software.* Madrid, España: Adison Wesley, 2004, Disponible en: http://es.scribd.com/doc/30251931/El-Proceso-Unificado-de-Desarrollo-de-Soft-Jacobson. ISBN 8478290362.
20. LAFUENTE, E.; LATORRE, P. M.*, et al. aCaSo: una herramienta para la gestión y análisis de Card Sorting.* 2006, Disponible en: http://www.laboratoriousabilidad.net/articulos.php. ISBN 84-690-1613-X.
21. LANDA-TORRES, F. J.; HERNÁNDEZ-GONZÁLEZ, S.*, et al.* Desarrollo de un módulo de pruebas de Estadıstica No Paramétrica con Java y R. 2011, nº Disponible en: www.sussex.ac.uk/Users/gr20/FNE(2011).pdf.
22. LARMAN, C. *UML Y PATRONES: Introducción al análisis y diseño orientado a objetos.* Editado por: Prentice Hall, H., S.A. . 2004, Disponible en: http://libropdf1.blogspot.com/2012/02/uml-y-patrones-introduccion-al-analisis.html. ISBN 9701702611.
23. LIME y CHILE. *Card Sorting with Results | WebSort.net* En linea. Disponible en: http://uxpunk.com/websort/.
24. LINARES, G. *Escalamiento multidimensional: conceptos y enfoques.* Ciudad de la Habana: Editorial Universitaria, 2001. ISBN 0257-4306.
25. LÓPEZ-GONZÁLEZ, E. y HIDALGO SÁNCHEZ, R. Escalamiento Multidimensional No Métrico. Un ejemplo con R empleando el algoritmo SMACOF. *ESE. Estudios sobre educación*, 2010, nº 18, p. 9-35. ISSN 1578-7001.

26. MARNET, I. *Analizar y representar gráficamente los resultados de un card sorting* Disponible en: http://www.deinterfaz.com/blog/analizar-y-representar-graficamente-los-resultados-de-un-card-sorting.
27. MONTES DE OCA SÁNCHEZ DE BUSTAMANTE, A. Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información. *ACIMED*, 2004, vol. 12, nº p. 1-1. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352004000600004&nrm=iso. ISSN 1024-9435.
28. NETBEANS. *¿Qué es NetBeans?* Disponible en: http://netbeans.org/index_es.html.
29. PINTO MOLINA, M. *Herramientas Estadísticas* Disponible en: http://www.mariapinto.es/e-coms/pa_es.htm.
30. PRESSMAN, R. S. *Ingeniería del "software", Un enfoque práctico.* 5ta ed. España: Imprenta Fareso S.A, 2005. ISBN 84-481-321-4-9.
31. PROJECT, P. *User Guides and Screencasts* Disponible en: http://pencil.evolus.vn/en-US/UserGuides.aspx.
32. QUAST, R. *Características del Java* Disponible en: http://es.scribd.com/doc/57226129/Caracteristica-Del-Java.
33. R-PROJECT. *JRI - Java/R Interface* Disponible en: http://www.rforge.net/JRI/.
34. RIPLEY, B. Package 'MASS'. 2010-01-03 2011, nº Disponible en: http://www.stats.ox.ac.uk/pub/MASS4/.
35. SALAS, C. ¿ Por qué comprar un programa estadístico si existe R? *Ecología austral*, 2008, vol. 18, nº 2, p. 223-231. ISSN 1667-782X.
36. SAS/STAT®, S. *Análisis Estadístico con SAS/STAT® Software* Disponible en: http://www.sas.com/offices/latinamerica/mexico/technologies/analytics/statistics/stat/index.html.
37. SCRIBD, I. *Herramientas de modelado* Disponible en: http://es.scribd.com/doc/73316306/DESARROLLO.
38. SLIDESHARE. *Pruebas de software* Disponible en: http://www.slideshare.net/aracelij/pruebas-de-software/.
39. SOMMERVILLE, I. *Software Engineering Eighth Edition.* Pearson Education Limited, China, 2006 ed. 2006. ISBN 978-0-321-31379-9.
40. USABILITYCENTRIC. *UXSort* Disponible en: http://www.uxsort.com/.
41. UZCARDSORT. Disponible en: http://uzilla.mozdev.org/cardsort.html.

42. VICENTE-VILLARDÓN, J. L. *INTRODUCCION AL ANÁLISIS DE CLUSTER.* Departamento de Estadística Universidad de Salamanca, 2006, Disponible en: http://biplot.usal.es/ALUMNOS/CIENCIAS/2ESTADISTICA/MULTIVAR/cluster.pdf.
43. VISUAL-PARADIGM. *Visual-Paradigm* Disponible en: http://www.visual-paradigm.com/product/vpuml/.
44. WORKSHOP, O. *OptimalSort* Disponible en: http://www.optimalworkshop.com/optimalsort.htm.
45. XSORT. Disponible en: http://www.xsortapp.com/.

Bibliografía consultada

1. ARCE, C.; DE FRANCISCO, C.*, et al.* Escalamiento multidimensional: Concepto y aplicaciones. *Papeles del Psicólogo*, 2010, nº 1, p. 46-56. ISSN 0214-7823.
2. BUZYDLOWSKI, J. W. *A comparison of self-organizing maps and pathfinder networks for the mapping of co-cited authors.* Drexel University, 2002, Disponible en: http://opim.wharton.upenn.edu/~sok/papers/b/janbuz-thesis.pdf. ISBN 0028-7806.
3. CARRERAS, O. *Usable & Accesible.* 2007, Disponible en: http://www.usableyaccesible.com/recurso_glosario.html.
4. CARRIÓN, J. J. S. Introducción al análisis multidimensional no-métrico. Reis, 1985, nº 29, p. 187-216. Disponible en: http://dialnet.unirioja.es/servlet/fichero_articulo?codigo=250540. ISSN 0210-5233.
5. DE CÁCERES, M. Ginkgo user's manual, versión 1.4. Unidad de Botánica, Universidad de Barcelona, Barcelona, 2005, nº Disponible en: http://biodiver.bio.ub.es/vegana/resources/help/ginkgo/Manual%20de%20GINKGO%201.4%20-%20ENG.pdf.
6. DE CÁCERES, M.; FONT, X.*, et al. VegAna, un paquete de programas para la gestión y análisis de datos ecológicos.* (Universidad de Barcelona). Avda. Diagonal 645, 08028 Barcelona, España: Departamento de Biología Vegetal Departamento de Estadística 2003, 1481-1497 p. Disponible en: http://biodiver.bio.ub.es/vegana/papers/AEET2003def.pdf.
7. GARRE, M.; CUADRADO, J. J.*, et al.* Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software. *REICIS Revista Española de Innovación, Calidad e Ingeniería del Software*, 2007, nº 001, p. 6-22. Disponible en: redalyc.uaemex.mx/redalyc/pdf/922/92230103.pdf.
8. GONZÁLEZ CARMONA, A.; ROMÁN MONTOYA, Y.*, et al.* Anidamiento en MDS mínimo cuadrático no métrico [MDS no métrico, Mínimos cuadrados, SSTRESS, Disimilaridad, Disparidad, ALSCAL, Propiedad de cascada]. *Estadística Española*, 1999, vol. 41, nº 144, p. 129-143. ISSN 0014-1151.
9. HERRERA-VILLAFRANCA, M.; GUERRA-BUSTILLO, C. W.*, et al.* Escalamiento Multidimensional y Mapas Auto organizados para visualizar el uso de los Métodos Estadísticos no paramétricos en la rama de las Ciencias Agraria y Biológica. *Ciencias de la Información*, 2012, vol. 43, nº 1, p. 51-56. Disponible en: http://cinfo.idict.cu/index.php/cinfo/article/view/382/pdf.
10. HOLLAND, S. *Non-metric multidimensional scaling (MDS).* University of Georgia: Department of Geology, 2008, Disponible en: http://strata.uga.edu/software/pdf/mdsTutorial.pdf.

11. KRUSKAL, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, 1964, vol. 29, nº 1, p. 1-27. Disponible en: http://www.springerlink.com/content/010q1x323915712x/. ISSN 0033-3123.
12. MARÍN, J. Tema 5: Análisis de Cluster y Multidimensional Scaling. 2006, Disponible en: http:// http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/AMult/tema5am.pdf
13. PARADIS, E. *R para Principiantes.* France: Universit Montpellier II,Institut des Sciences de l'E´volution, 2003, Disponible en: http://cran.r-project.org/doc/contrib/rdebuts_es.pdf.
14. PASCUAL, D.; PLA, F., et al. Algoritmos de Agrupamiento. Departamento de Computación Universidad de Oriente & Departamento de Lenguajes y Sistemas Informáticos Universidad Jaume I, 2007, nº Disponible en: http://marmota.dlsi.uji.es/WebBIB/papers/2007/1_Pascual-MIA-2007.pdf. ISSN 0302-9743.
15. RIVAS, T. y MARTÍNEZ ARIAS, R. Relación entre escalamiento multidimensional métrico y análisis de componentes principales. Psicothema, 1991, vol. 3, nº 2, p. 443-451. Disponible en: http://www.psicothema.com/psicothema.asp?id=2033. ISSN 0214-9915.
16. TOUS RAL, J. M. y FERRANDO PIERA, P. J. Aplicaciones de las EMD al análisis del diferencial semántico. *Psicothema*, 1991, vol. 3, nº 2, p. 453-466. Disponible en: http://www.psicothema.com/pdf/2034.pdf. ISSN 0214-9915.
17. VILLALOBOS, M. y TREJOS, J. Análisis de proximidades métrico usando búsqueda tabú. *Revista de Matemática: Teoría y Aplicaciones*, 2009, vol. 7, nº 1-2, p. 71-76. ISSN 1409-2433.

Anexos

Agrupamiento de tarjetas: Fase de preparación

Pantalla: Crear tarjeta

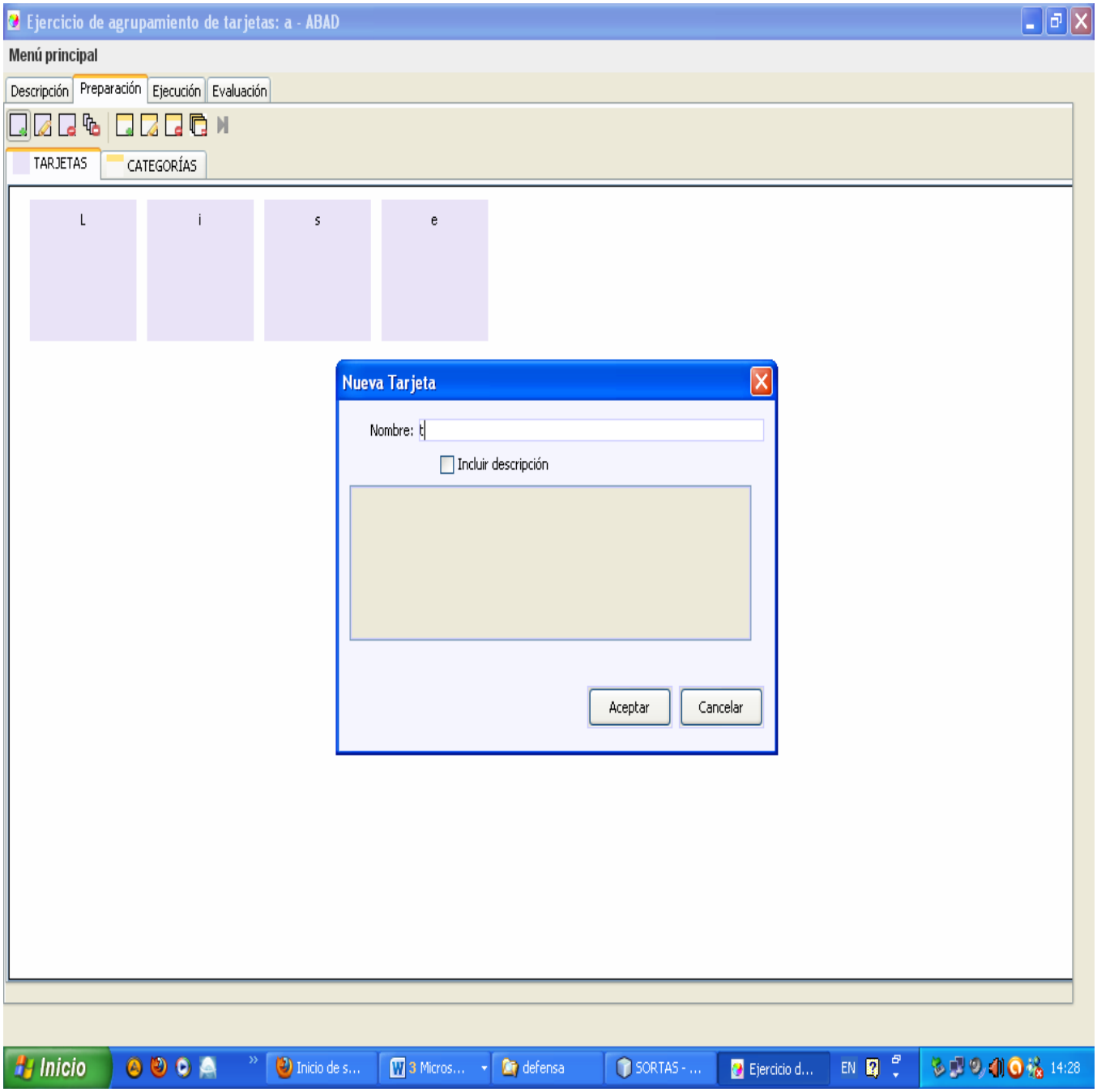


Figura 16. Crear tarjeta

Pantalla: Crear de categoría

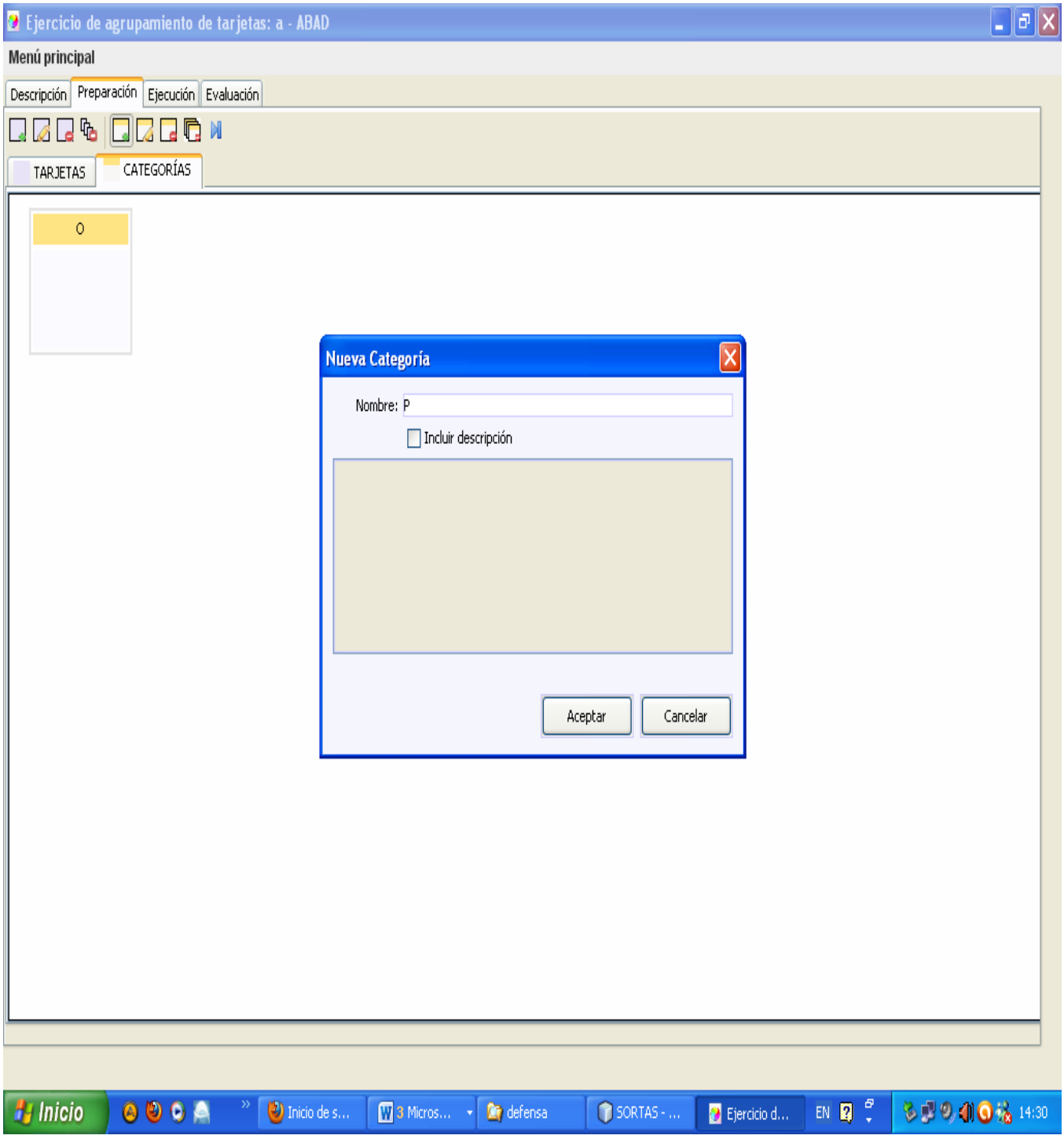


Figura 17. Crear de categoría

Agrupamiento de tarjetas: Fase ejecución

Pantalla: Agrupar las tarjetas en las categorías

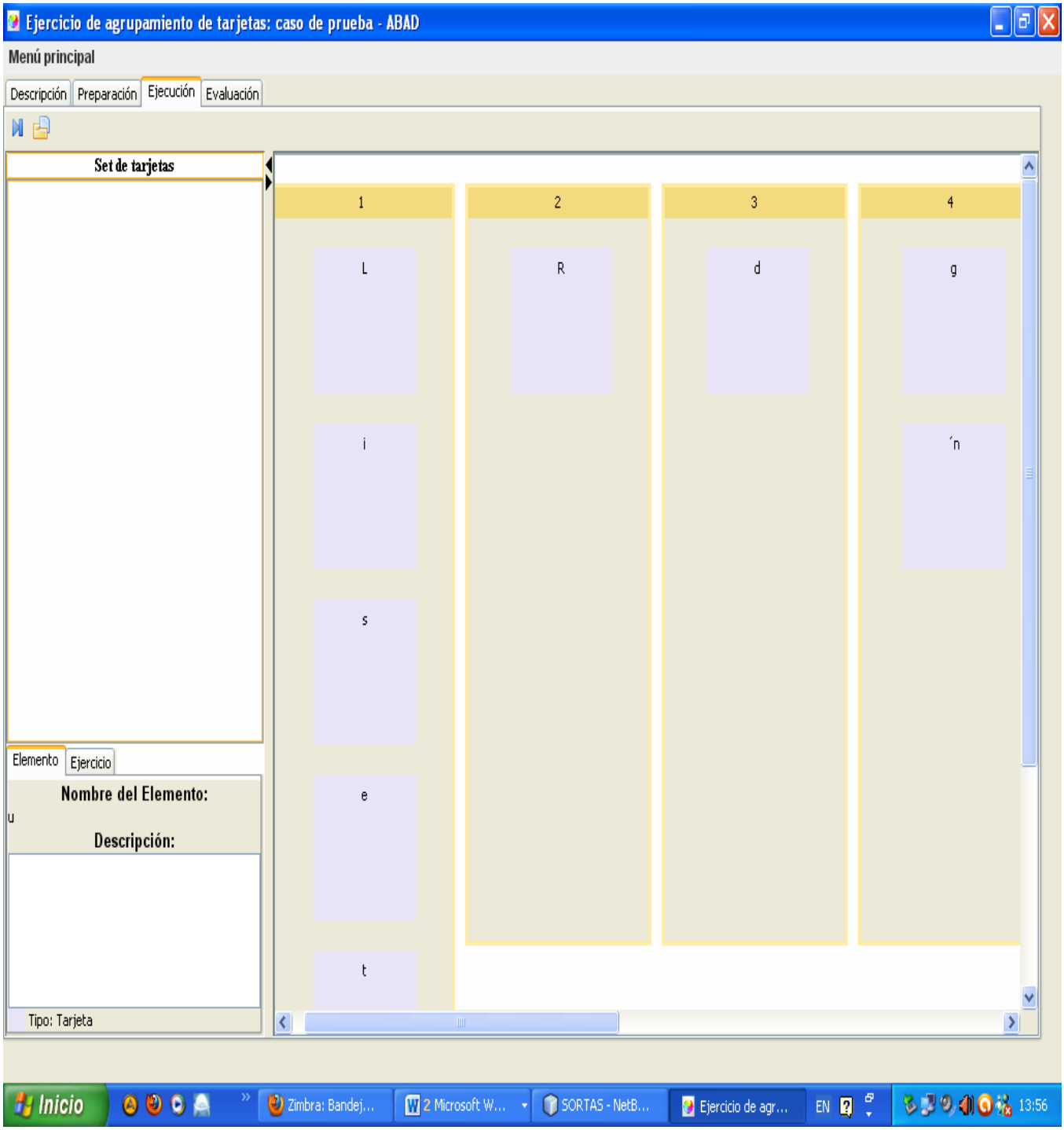


Figura 18. Agrupar las tarjetas en las categorías

Agrupamiento de tarjetas: Fase evaluación

Pantalla: Seleccionar algoritmos de evaluación

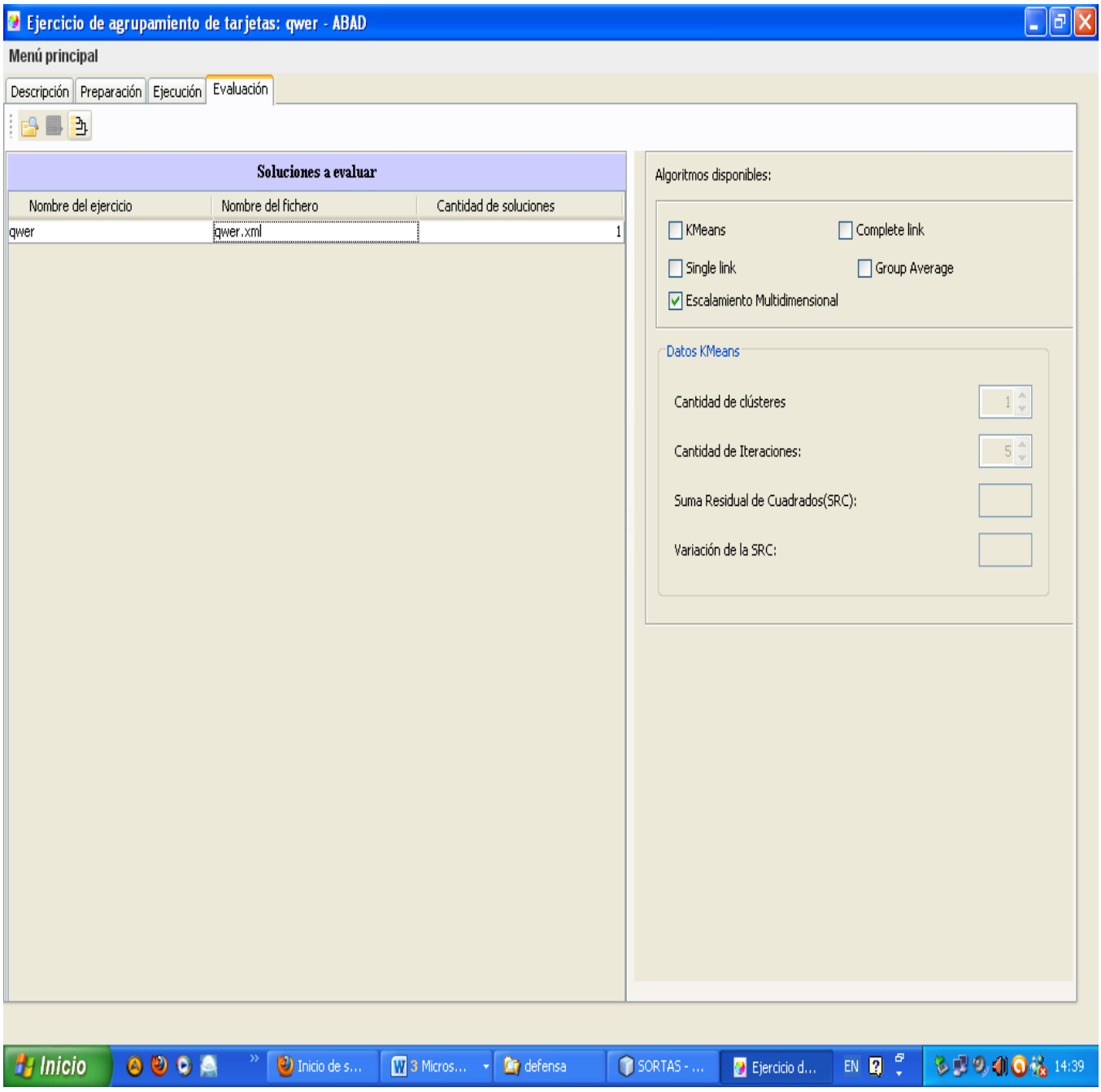


Figura 19. Seleccionar algoritmos de evaluación

Pantalla: Mostrar resultados

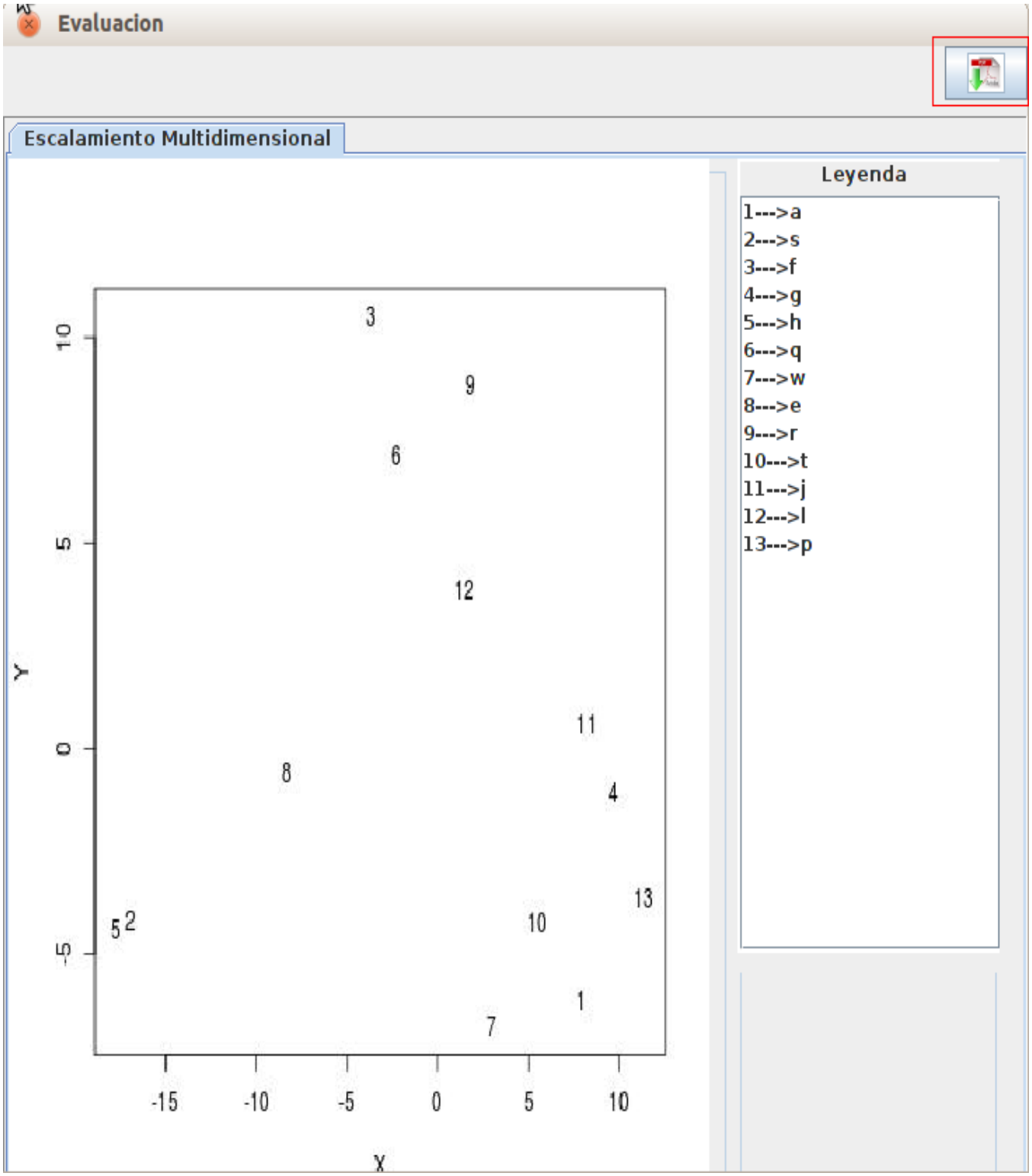


Figura 20. Mostrar resultados

Pantalla: Informe PDF

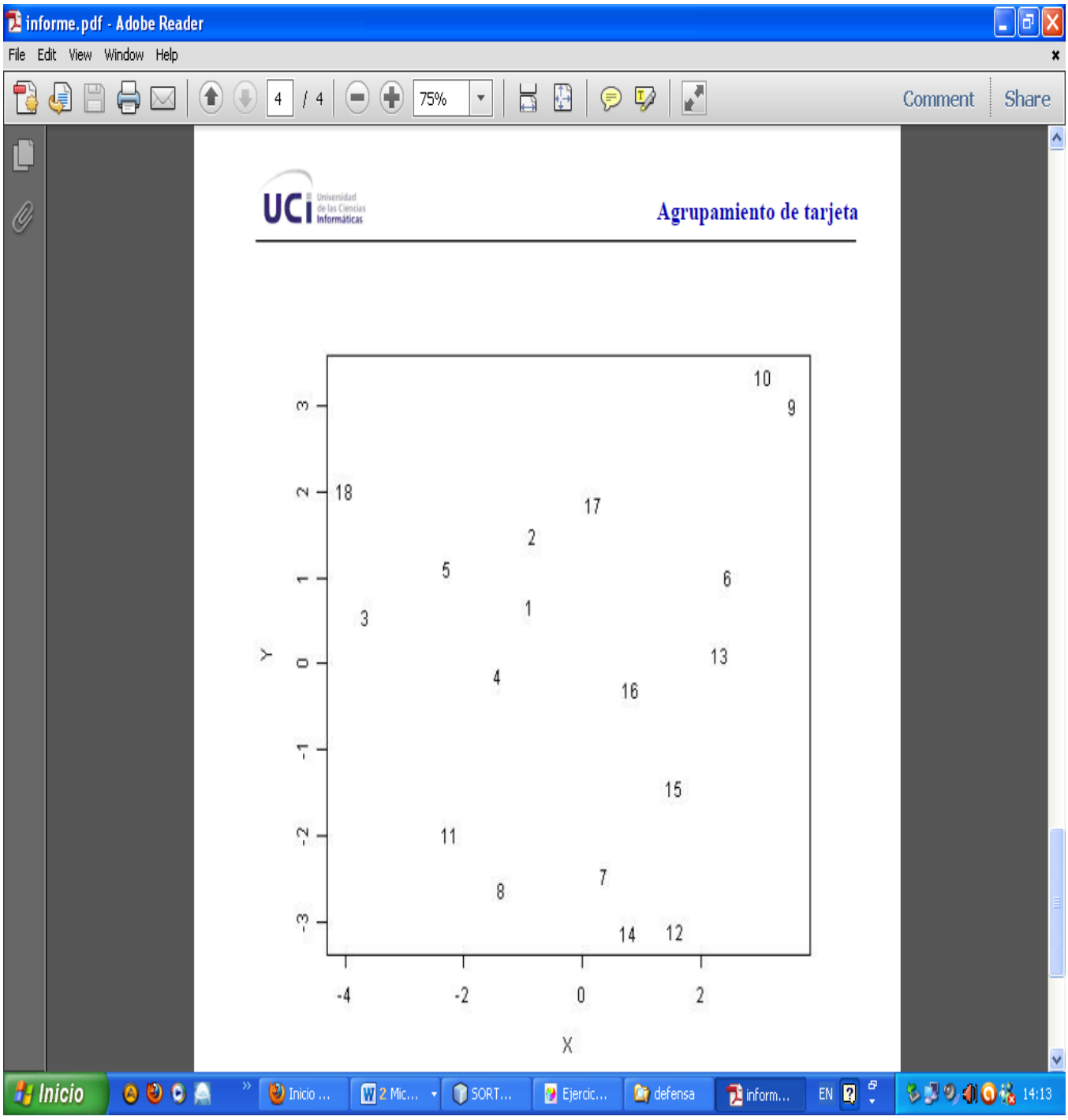


Figura 21. Informe PDF