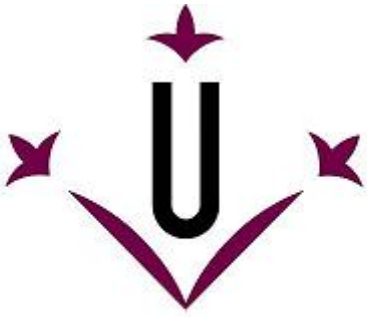
Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Sistemes

Treball de final de carrera



Adaptación de aplicación para la realización de Card Sorting para dotarla de elementos multimedia y capacidad de distribución online

Autor: Raquel Millanes Vilas
Director: Juan Miguel López Gil
Enero de 2009

Agradecimientos

Este trabajo surge a mediados de Abril de 2008. GRIHO, Grup de Recerca en Interacció Persona Ordinador quería realizar algunas modificaciones en una aplicación de Card Sorting ya existente añadiendo unas nuevas funcionalidades.

Por este motivo, Juan Miguel López Gil, profesor de la UdL, me ofreció la posibilidad de realizar estas modificaciones como proyecto fin de carrera.

Las modificaciones que este trabajo ofrece van desde hacer que la aplicación pueda ser distribuida tipo Web hasta incorporar la posibilidad de definir elementos multimedia en la representación de información. Como efecto colateral, ello redunda a beneficios en los que usuarios con discapacidades pueden acceder a distintas representaciones de dichos elementos multimedia.

Por otro lado, quiero dar las gracias a todas esas personas que han hecho posible este proyecto, ya sea de forma directa o indirecta:

A Héctor, a mi familia y a mis compañeros de piso por estar siempre ahí aguantando mis explicaciones y quejas. Pero especialmente por el apoyo moral y los ánimos que me habéis dado durante todo este tiempo. Al mismo tiempo, disculparme por todos esos momentos en los que no os he podido prestar toda la atención que merecíais.

Como no, a Juanmi, por ofrecerme la posibilidad de realizar este proyecto e intentar ayudarme en todos los problemas y dudas que han ido surgiendo a lo largo de este trabajo.
Y para terminar, a todas esas personas cuyo nombre no aparece aquí pero que de algún modo me han ayudado.

Raquel Millanes Vilas

Índice

Índice de figuras

Capitulo 1

Introducción

Actualmente, existe cierto interés en la forma en la que se puede actuar frente a las nuevas tecnologías, como pueden ser ordenadores, teléfonos móviles, etc. Podríamos pues hablar, de una nueva "era" dentro del campo de la tecnología, la cual se centraría en mejorar lo ya existente. Todo ello nos conduce a nuevas líneas de investigación que sugieren nuevos métodos en los sistemas de investigación de los campos informáticos. Éstos deberán centrarse en la figura del usuario y su correspondiente interacción ante el computador, es decir, que la misión que se le otorga a la nueva generación del campo de la informática no es otra que el facilitar al máximo la denominada Interacción Persona-Ordenador. Sin duda este es quizás, el mayor de los retos que actualmente deba tener cualquier investigador de la informática, puesto que el estudio del intercambio de información del usuario con su computador, debe ser siempre favorable para el mismo. Por tanto debemos actuar en la forma del objetivo que nos lleve a la evidencia y simplicidad en la que el usuario pueda sentirse satisfecho ante cualquier tipo de aplicación.

Para resolver estas dudas y conseguir una mejora surge CardSorting, que se centra en estudiar la forma en la que los usuarios organizan mentalmente la información de cualquier aplicación. Así, se pretende facilitar la interacción del usuario, organizando la información conforme a su modelo mental.
Este consiste en implementar una serie de tarjetas con la información que se desee estudiar y será el usuario quien las ordene según crea más oportuno.

1.1 Objetivos del trabajo

Este proyecto surge de la necesidad de mejorar un CardSorting ya existente de forma que, mediante un servidor, podamos usarlo de forma remota, y poder separar la aplicación en dos partes; ya que hasta el momento se encontraba en una sola. Con esto se consigue que la creación de las tarjetas pase desapercibida para el usuario y que a éste le resulte más simple el uso de esta aplicación. Con esto, se lograría que la aplicación pueda ser distribuida de forma Web.

También será modificado el tipo de tarjetas para poder introducir sonidos, videos e imágenes. Con lo cual conseguimos adaptar el proyecto a personas discapacitadas como pueden ser sordos e invidentes además de aportar un uso más amplio a cada una de las tarjetas

1.2 Estructura del documento

Este documento esta formado por ocho capítulos más un anexo. Todos estos capítulos tienen una pequeña descripción sobre su contenido al inicio y a su vez, estos están divididos en subapartados.

En el anexo podemos encontrar una pequeña explicación sobre cómo utilizar la aplicación.

Capitulo 2

Modelo de proceso centrado en usuario

El modelo de proceso centrado en el usuario es una guía a la hora de seguir el desarrollo de cualquier software. Tanto en aplicaciones completas o paginas webs como en aplicaciones basadas en Internet.

La técnica de Card Sorting se encuentra dentro del proceso centrado en el usuario, concretamente dentro de la arquitectura de la información. En este capítulo describiremos las principales fases de este modelo, así como la importancia de la arquitectura de la información y de la tarea como el Card Sorting para un modelo de proceso centrado en el usuario.

2.1 Conceptos previos

En primer lugar, haremos referencia sobre algunos conceptos que nos servirán para comprender mejor el modelo del proceso centrado en el usuario.
Principalmente nos centraremos en la usabilidad y la accesibilidad.

2.1.1 Usabilidad

Este concepto, popularizado por J.Nielsen, puede definirse de forma coloquial como la propiedad que tiene un determinado sistema para que sea “fácil de usar o utilizar y de aprender”, tratándose de una propiedad que no es mas aplicable a los sistemas software, sino que, como muestra D.Norman, es aplicable a los elementos de la vida cotidiana.
Esta definición, que en esencia es correcta, no deja de ser incompleta ya que el termino engloba muchas mas connotaciones.
A continuación, veremos dos definiciones puestas por el prestigioso organismo de estandarización ISO, Internacional Standarisation Organization, definidas en función de los términos que considera el momento de especificar o evaluar la usabilidad.

La primera de estas dos definiciones forma parte del estándar **ISO/IEC 9126**:

"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso"

Esta definición hace énfasis en los atributos internos y externos del producto, los cuales contribuyen a su funcionalidad y eficiencia. La usabilidad depende no sólo del producto sino también del usuario. Por ello, un producto no es en ningún caso intrínsecamente usable, sólo tendrá la capacidad de ser usado en un contexto particular y por usuarios particulares. La usabilidad no puede ser valorada estudiando un producto de manera aislada (Bevan, 1994).

La segunda definición pertenece al estándar **ISO/IEC 9241:**

"Usabilidad es la eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico"

Es una definición centrada en el concepto de calidad en el uso, es decir, se refiere a cómo el usuario realiza tareas específicas en escenarios específicos con efectividad.

2.1.2 Accesibilidad

La accesibilidad es el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas o físicas.

En nuestro caso, accesibilidad implica facilitar un sistema informático para que pueda ser usado, visitado o accedido en general para todo tipo de personas, especialmente por aquellas que poseen algún tipo de discapacidad.

Para promover la accesibilidad se usan ayudas, para evitar los obstáculos del entorno, consiguiendo de esta forma que las personas realicen la misma acción que podrían realizar otras sin ningún tipo de discapacidad. Estas facilidades reciben el nombre de ayudas técnicas. Entre estas se encuentra la lengua de signos, el alfabeto Braille, etc.

En el caso informático, la accesibilidad se proporciona mediante una combinación de hardware y software. El primero nos proporciona los mecanismos físicos que permiten salvar ciertas discapacidades y el segundo proporciona la manera eficaz de acceder a las funcionalidades e informaciones para este dispositivo y otros programas.

2.2 Esquema del Modelo de Proceso

Entre las características principales de usabilidad hay que destacar la claridad de la información y consistencia. Desde esta óptica y respetando lo anterior, se considera muy importante que un método disponga de un esquema, que en todo momento permita ver claramente al usuario en qué fase del desarrollo se encuentra y qué posibilidades tiene, a partir de la fase actual para continuar su desarrollo.

El Modelo de Proceso de la Ingeniería de la Usabilidad y Accesibilidad (MPIu+a) que propone Toni Granolles, profesor de la UdL, determina una serie de actividades bien organizadas que en grandes términos los podemos clasificar como:

- Actividades estructuradas de los análisis de requisitos de la usabilidad.
- Un conjunto de actividades explicito de objetivos de estabilidad.
- Actividades de soporte a una aproximación estructurada del diseño de la interface de usuario
- Actividades de evaluación de los objetivos de usabilidad mediante interacciones en el diseño.

El esquema de MPIu+a es el siguiente:

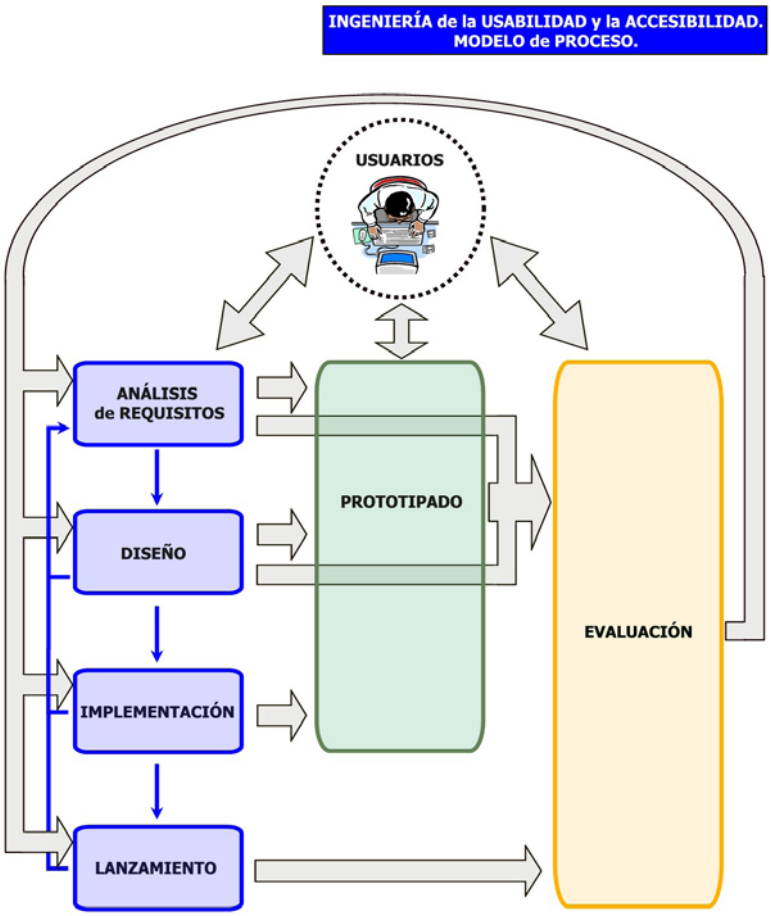


Figura 1: Ingeniería de la usabilidad y accesibilidad. Modelo de proceso.

En la Figura 1, observamos un esquema de un modelo circular en el que todas las etapas intervienen en la evaluación, ya sea inicialmente evaluando al usuario y su lugar de trabajo o posteriormente realizando algún tipo de prototipo en el que cuestiona la usabilidad.

En el diseño del esquema se ha intentado poner de manifiesto una serie de premisos básicos que se perciben con un solo golpe de ojo. Estas son:

1. **Organización Conceptual**

   El esquema se organiza sobre la base de una serie de módulos o nodos que determinan la fase de desarrollo en el que nos encontramos, además de situarnos en un paso concreto la actividad del conocimiento existente en IPO (Interacción Persona Ordenador).

2. **Tres pilares Básicos**

   El principal motivo del Modelo de Proceso es conseguir unir el modelo lineal de desarrollo de sistemas interactivos de Ingeniería del Software, con los principios básicos de Ingeniería de la usabilidad, incorporando una metodología que guíe el desarrollador del proceso de la construcción de un determinado sistema.
   Dentro de Ingeniería de la usabilidad hay dos conceptos muy importantes que han de realizarse: el prototipo y la evaluación
   Estos tres conceptos son los principales pilares básicos:

   -Ingeniería del Software clásica (columna azul a la izquierda del Análisis/Diseño/Implementación/Lanzamiento)
   -El Prototipo (columna verde del centro) como metodología que engloba técnicas que permitirán la posterior fase de evaluación.
   -Evaluación (columna roja de la derecha) que engloba y categoríza los métodos de evaluación existentes.

3. **El usuario**

   Un proceso de diseño centrado en el usuario, ha de dejar claro que es así, sólo con mirar el esquema la primera vez. Esto es lo que queda reflejado al disponer esta parte central sobre el resto de nodos. De esta forma el usuario esta en el centro del desarrollo y en las facetas que intervienen.

4. **Interactividad del Método**

   En todo proceso de desarrollo de Software existe una fase más o menos importante sobre la base de una serie de repeticiones, pasando de una aproximación de la solución ideal a la solución definitiva.

2.2.1 Análisis de requisitos

Un requisito se define como la descripción de una característica, capacidad o restricción del sistema.

De aquí deducimos, al analizar los requisitos de un sistema interactivo, que estos deben determinar, enumerar y clasificar todas sus características, capacidades y restricciones, que el sistema ha de cumplir o a las que se viera sometido.

2.2.1 Análisis de requisitos

Esta fase es muy importante a la vez que crítica y de ella depende la buena continuación de un proyecto. Pared Spool, argumenta que si realizamos un buen trabajo en esta fase, repercutirá directamente a disminuir un número de interacciones necesarias para conseguir el proceso global.

Su importancia es tal que algunos autores como Gerard Kotonya y Ian Sommerville han definido y desarrollado la conocida “Ingeniería de los Requisitos”, debido a la gran importancia de esta fase.

De este modo, definimos el “Proceso de la Ingeniería de los Requisitos” como un conjunto estructurado de actividades necesarias para desarrollar la actividad y análisis de los requisitos, las cuales conducen a la producción de un documento de requisitos.

2.2.2 Diseño

El principal aspecto que en los sistemas interactivos reside es la interacción con el usuario. La interface es la parte hardware y software del sistema informático que facilita al usuario el acceso a los recursos del computador. Thimbleby, sugiere que la interface determinara en gran medida la percepción e impresión que el usuario poseerá de la aplicación.

En esencia el diseño de sistemas interactivos es proporcionar el soporte necesario a las personas en su trabajo cotidiano. Esto nos muestra la importancia del proceso de desarrollo de cualquier aplicación. Este es el motivo por el que se debe tener en cuenta desde el principio.

No es del interés del usurario la estructura interna de la aplicación pero sí lo es la forma de usar la misma. No sería correcto realizar una implementación sin haber planteado previamente el diseño del mismo ya que en ese caso el diseño de interfaces sería muy independiente a los diseños de datos y funciones.

Lo correcto sería centrarnos en una idea clara basada en las interacciones y características del usuario para poder realizar las correspondientes especificaciones funcionales que nos servirán de guía en el diseño posterior.
Las actividades que se realizan durante la fase de diseño son:

- Análisis de Tareas
- Modelo conceptual
- Definir un estilo general
- Diseño detallado

También podríamos incluir en este listado la Arquitectura de la Información, AI (Capítulo 2.3). Éste hace referencia a la combinación de esquemas de organización.

Actualmente esta siendo uno de los pilares a la hora de diseñar un lugar Web basado en el modelo de proceso centrado en el usuario.

2.2.3 Prototipo

Referirnos a prototipos nos sugiere que estamos pensando en documentos, diseños o sistemas que simulan o tienen implementadas partes de un sistema final que habremos de implementar. Este término engloba todas las tareas que permiten realizar a los diseñadores de sistemas estas simulaciones.

Los prototipos responden cuestiones y dan soporte a los diseñadores a la hora de elegir entre distintas alternativas. Es mas, sirven para una gran variedad de propósitos como testear la fiabilidad técnica de una idea, clasificar requisitos que quedaron “imprecisos”, ver como responde con el resto de la aplicación, etc.

Algunas de las actividades que se realizan en el prototipo son:

- Maquetas
- Prototipos de papel
- Storyboards

2.2.4 Evaluación

El objetivo de los prototipos, no tiene sentido si no fuese por que estos fueron evaluados para poder comprobar anticipadamente el funcionamiento del sistema. El prototipo, es una tarea útil para hacer participar al usuario en el desarrollo y poder evaluar el producto ya en las primeras fases del diseño.

Existen tres tipos de actividades a realizar durante la evaluación:

- **Inspección:** Recorrido cognitivo, Heurística, etc.
- **Indagación:** Entrevistas, Cuestionarios, etc.
- **Test**: Thinking Aloud, Interacción, Constructiva, etc.

Inspección: Este término, aplicado a la usabilidad, engloba a un conjunto de métodos en los que hay unos contenidos expertos, como evaluadores, que explican el grado de estabilidad de un sistema basándose en la inspección o examen de la interface del mismo. Existen diversos métodos que se almacenan en la clasificación de evaluación para inspección.

Indagación: Este proceso trata de llegar al conocimiento de una conjetura o señal. En los métodos de evaluación realizados por indagación hay un gran trabajo centrado en dialogar con los usuarios y observarlos detenidamente usando el sistema de trabajo real

(no para un test de usabilidad) o obtener respuestas a preguntas verbalmente o por escrito.

Test: En los métodos de usabilidad para el test, usuarios representativos trabajan en tareas utilizando el sistema , o prototipo- y los evaluadores utilizan los resultados para ver como la interfase de un usuario soporta sus tareas.

2.2.5 Implementación

Este apartado o fase agrupa todo el trabajo de codificación de la aplicación que hasta este punto se han ido construyendo.

Llegados a este punto, es cuando hay que empezar a programar, esto implica haber escogido el lenguaje de programación que mejor se adapte a nuestro proyecto, las bases de datos correspondientes que se precisen, la tecnología que garantice el éxito, etc.
En definitiva, aquí es donde empiezan la mayoría de proyectos interactivos que hoy en día se desarrollan.

2.2.6 Lanzamiento

La fase de lanzamiento, en todo proyecto, sea interactivo o no, suele ser una de las mas criticadas en todo el proceso. Es el momento en que veremos concretadas en mayor o menor grado las expectativas puestas en el producto. Si el cliente se trata de una organización, dependerá de las personas, dentro de su estructura de jerarquía, examinaran los resultados. También indicar que la percepción del usuario final ante el producto tiene un peso especifico a la hora de indicar si el producto será aceptado o no.

En resumen, podemos indicar que el éxito del producto dependerá de dos factores muy importantes: por un lado que el usuario se sienta cómodo con el sistema y que los responsables obtengan los resultados esperados.

El ciclo de vida de la Ingeniería de la Usabilidad asegura que ambos aspectos se vean satisfechos ya que el diseño se ha hecho en base y para los usuarios, haciéndolos partícipes del mismo. De este modo hemos conseguido un efecto doble, por un lado, como se sienten responsables, en parte, de este diseño, no encontraran motivos para criticarlo y por otro, como todo ha estado evaluado por ellos no les supondrá una gran carga cognitiva ni de aprendizaje.

2.3 Arquitectura de la Información

La **Arquitectura de la Información (AI)** es la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de

información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos.

En relación con la World Wide Web el Information Architecture Institute, define la Arquitectura de la Información como:

1. El diseño estructural en entornos de información compartida.

2. El arte y la ciencia de organizar y rotular sitios web, intranets, comunidades en línea y software para promover la usabilidad y encontrabilidad.

3. Una comunidad emergente orientada a aplicar los principios del diseño y la arquitectura en el entorno digital.

La Arquitectura de la Información trata indistintamente del diseño de: sitios Web, interfaces de dispositivos móviles o gadgets (como los iPod), CDs interactivos, videoclips digitales, relojes, tableros de instrumentos de aviones de combate o civiles, interfaces de máquinas dispensadoras, interfaces de juegos electrónicos, etc.

Su principal objetivo es facilitar al máximo los procesos de comprensión y asimilación de la información, así como las tareas que ejecutan los usuarios en un espacio de información definido.

Capítulo 3

Card Sorting

CardSorting es una técnica para la clasificación de contenidos de forma que un usuario, pueda clasificar la información según su estructura mental, facilitándole la interacción con las nuevas tecnologías.
Esto nos permite crear, desde las primeras etapas de diseño, categorías organizadas y denominadas que se aproximan a la manera de pensar del usuario y a la vez su modelo mental.
Esta técnica se puede realizar mediante software pero también de forma manual mediante tarjetas de papel o cartón.

3.1 Tipos de Card Sorting

En la actualidad existen dos tipos de Card Sorting, abierto y cerrado:

- Card Sorting abierto es ese en el que el usuario puede llamar y agrupar las tarjetas libremente, creando el número de conjuntos que considere necesario. Este tipo de ordenación se usa para obtener una estructura de la información para el diseño de un nuevo sistema.

- Card Sorting cerrado tiene categorías predefinidas. El usuario simplemente debe colocar la categoría en el grupo que considere más oportuno. Este tipo se suele usar un nuevo contenido en un sistema existente.

Estos dos tipos de Card Sorting tienen objetivos distintos, Card Sorting abierto pretende descubrir el tipo de clasificación de las categorías sugeridas por los usuarios, mientras que el cerrado se suele utilizar para verificar si la clasificación de una información determinada es adecuada para el usuario.

3.2 Ventajas e inconvenientes

La utilización de Card Sorting nos proporciona distintos beneficios.
Para empezar, su facilidad de manejo es una de las principales ventajas de su uso y además este apenas tiene coste, solo el tiempo que cueste su preparación.

Ya que la sesión no dura mucho tiempo, permite que haya más personas interesadas en realizarla y por lo tanto mayor número de opiniones.
Pero sobre todo, ya que esto permite a los usuarios involucrarse directamente en el diseño de la aplicación ayuda a demostrar que el sistema esta creado a partir del modelo mental que estos tienen proporcionando una buena base, ya que se proporciona una estructura concreta para el sistema que estamos diseñando.

Como se ha podido ver el uso de Card Sorting nos proporciona distintas ventajas pero como no, también se puede encontrar en su uso algún inconveniente:

Debido a que el Card Sorting no tiene en cuenta las tareas que el usuario realizara, podríamos llegar a una estructura de la información que no fuese usable.
También cabe la posibilidad que exista una gran diferencia entre un resultado y otro convirtiéndose la sesión en incoherente.
El análisis puede resultar costoso en cuanto al tiempo. No precisamente por la agrupación de tarjetas sino respecto al análisis del mismo.

Este balance nos hace a ver que la eficiencia de Card Sorting es mucho mayor cuando se ha identificado meticulosamente el contenido. Además, ayuda a los diseñadores a resolver algunas preguntas durante la fase de diseño como podría ser la organización que el usuario considera más eficiente, cuáles son sus necesidades.

La ordenación de tarjetas es útil cuando queremos diseñar un nuevo sistema como podría ser un portal Web. También resulta de gran utilidad a la hora de diseñar o rediseñar una nueva área dentro de una Web.

Para mejorar su eficiencia, sería conveniente complementar la actividad del Card Sorting con otras actividades con análisis de la información, análisis de tareas y/o análisis de usabilidad para poder obtener una correcta arquitectura de la información del sistema de desarrollo.

3.3 Creación de tarjetas

La creación de tarjetas también pude realizarse de forma manual. Tanto creando las tarjetas manualmente como si lo hacemos mediante la aplicación de Card Sorting, es muy importante el contenido de estas. Los nombres o frases que introduciremos en las tarjetas deben ser cortas para que los participantes puedan leerlas rápidamente, identificarlas y entender su significado.

Figura 2: Preparación del Card Sorting

Es recomendable controlar el número de tarjetas ya que si este es muy elevado la prueba se hace más costosa para los participantes obteniendo peores resultados.

3.4 Selección de participantes

Los asistentes en una sesión de Card Sorting deberían ser usuarios finales reales o potenciales del sistema que estamos construyendo. En el caso de que el participante fuese propietario de la empresa o del sistema que estamos construyendo los resultados no serán validos ya que el modelo mental del propietario será distinto al del resto de usuarios.

Para que la sesión de Card Sorting sea favorable necesitaríamos entre 15 o 20 participantes. En el caso de tener menos participantes los resultados no serian suficiente fiables, por el contrario con un gran numero de participantes el rendimiento de la aplicación disminuye aumentando lentamente el grado de correlación de la sesión.

El número de participantes requeridos para Card Sorting puede resultar un poco elevado si lo comparamos con otros estudios en los que no se requieren más de 5 participantes. Esto es debido a que nosotros disponemos de un diseño y pretendemos cerciorarnos de si este es correcto o no, debido a que las personas no compartimos los modelos mentales ni el mismo vocabulario se necesita un numero de personas mas elevado para sacar conclusiones adecuadas para la mayoría.

3.5 Sesión de Card Sorting

Antes de realizar una sesión de Card Sorting sería conveniente realizar los siguientes pasos:

- Redactar un dossier para evaluar a los participantes. En este dossier debe aparecer una parte meramente explicativa sobre los objetivos del proyecto y la sesión de Card Sorting. También debe aparecer la frecuencia con la que aparecen las tramas de ordenación. Y para terminar, algunas preguntas realizadas a los participantes una vez finalizada la agrupación. En estas preguntas se le ofrece a los participantes la oportunidad de detallar algún detalle o incluso matizar algún aspecto.

- Se puede realizar una prueba piloto con dos o tres usuarios antes de realizar las pruebas reales. De esta forma, se comprueba que no haya ningún problema ni error en las tarjetas.

- Hay que disponer de tarjetas en blanco y lápiz para cada uno de los participantes. Así, estos podrán poner nombre a los grupos de tarjetas.

Una vez nos disponemos a iniciar la sesión, es necesario agrupar a los participantes y realizar una explicación sobre qué consiste y cuáles son los objetivos. Esto también se puede hacer de forma escrita y repartirlo entre los participantes. De este modo, nos aseguramos de que todos los participantes tendrán los mismos conocimientos a la hora de enfrentarse a la sesión.

Una vez iniciada la sesión, nuestra misión es escuchar y observar al participante. Es muy importante no influir en la opinión de los participantes. Del mismo modo, es muy importante que entre los participantes no haya intercambio de opiniones durante la sesión para evitar que un participante se vea influenciado por la opinión del otro.

Podemos coger notas en un cuaderno sobre las preguntas y sugerencias que vayan surgiendo a lo largo de la sesión. Puede ser muy interesante fijarse también en la expresión no verbal de los participantes.

Durante la sesión puede ser conveniente realizar preguntas de nuestro interés a los participantes. Por ejemplo, se les puede preguntar cuales de las tarjetas serían más utilizadas dentro de un portal Web.

Una vez terminada la sesión obtenemos un resultado similar a la Figura 3, tarjetas agrupadas en grupos aunque en algún caso podemos encontrar tarjetas desagrupadas.

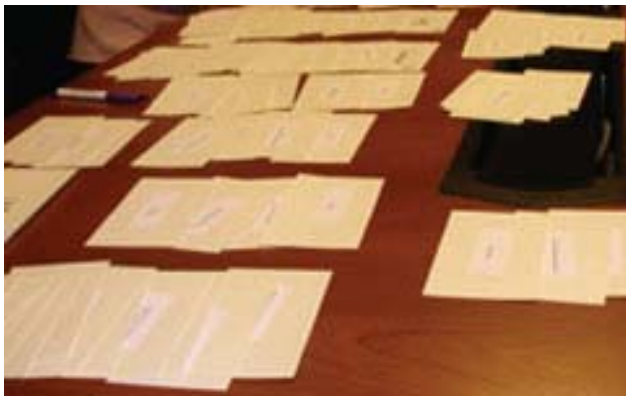


Figura 3: Resultado del Card Sorting

En el caso de que la sesión que estemos realizando sea de tipo abierto, una vez agrupadas las tarjetas, el participante ha de darle un nombre significativo a cada grupo. Este será el menú de nuestro sistema.

Una vez haya terminado el participante, podemos sugerirle que nos indique cual sería la navegación que él realizaría para llegar a una tarea en concreto. Esto nos ayudará a validar los resultados.

3.6 Análisis de los resultados

Una vez finalizada la sesión del Card Sorting se procede al análisis de los resultados obtenidos. Normalmente entre un 60% y un 80% de los participantes han realizado la misma agrupación.

La cantidad y cualidad de la información que se puede sacar de los resultados depende del tipo de análisis que vayamos a realizar. Este análisis se puede hacer de dos formas:

- Análisis cualitativo: se basa en observar los grupos creados por los participantes y buscar algún patrón dentro de estos grupos.

- Análisis cuantitativo: se basa en la aplicación de técnicas estadísticas sobre agrupaciones hechas por los participantes.

Capitulo 4

Análisis de requisitos

En Septiembre de 2005 un alumno de la Universitat de Lleida realizó como proyecto fin de carrera la implementación de un Card Sorting multiplataforma con el objetivo de crear un Card Sorting virtual y estudiar el modelo de proceso central de un usuario, centrándose concretamente en la Arquitectura de la Información del mismo. En 2006 otro alumno realizo una implementación para poder realizar analizar los resultados obtenidos.

4.1 Análisis de antecedentes

Como ya se ha dicho a lo largo del trabajo, el Card Sorting anterior, construido por Eduard Porta Miret en 2005, es una aplicación mediante la cual podemos construir y realizar una sesión de tarjetas. Ambas situaciones las realizaremos en la misma aplicación.

Las tarjetas constan de dos componentes: Título y Descripción. Una tarjeta siempre debe estar completa por lo menos por uno de los dos componentes. Estos dos componentes solo podrán ser tipo texto.

En la figura 4 podemos ver una representación de lo que nos ofrece el anterior Card Sorting. Las tarjetas tienen un tamaño específico para el contenido que pueden tener.

Las tarjetas pueden ser creadas, modificadas y eliminadas en todo momento excepto mientras se realizada la sesión.
En esta parte las tarjetas son agrupadas dando un nombre a cada uno de los grupos.

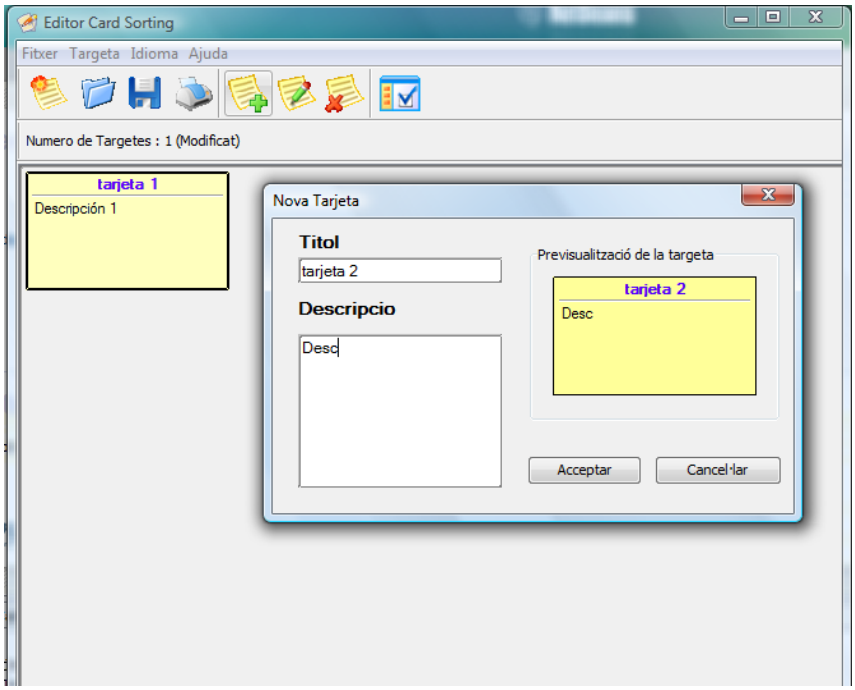


Figura 4: Card Sorting anterior

También existe otra aplicación, construida por Daniel Pardell Mateu como trabajo final de carrera. Esta se centra en analizar los resultados obtenidos tras una sesión de Card Sorting. Esta utiliza como fichero de entrada la salida de la sesión en la que se agrupan las tarjetas. Además, realiza un análisis sobre la agrupación de cada uno de los participantes.

4.2 Requisitos funcionales

A continuación realizaremos un estudio sobre cuales deben ser las modificaciones a tener en cuenta en este trabajo como resultado final.

Los requerimientos funcionales de un programa son aquellos en los que se establecen las necesidades del usuario.

Un usuario necesita que un programa mediante el cual pueda conocer la estructura mental de los usuarios que van a utilizar una determinada aplicación. En esta aplicación puede haber una gran diversidad de usuarios ya sea para crear las tarjetas como para agruparlas. Por este motivo la aplicación debe estar preparada para todo tipo de usuarios.

Las modificaciones que se van a tener en cuenta a lo largo de este trabajo van a estar pensadas principalmente para mejorar la interacción con los usuarios.

- La interfaz debe ser simple e intuitiva a lo largo de toda la aplicación. De esta forma los usuarios serán capaces de utilizar la aplicación sin necesidad de recibir ayuda.

- Actualizar las tecnologías utilizadas para la representación de la información contenida en las tarjetas. En este caso, utilizar esquemas XML en vez de DTD, como se realizaba en la anterior aplicación. De esta manera se prevé ganar en simplicidad y ahorrar espacio.

- La aplicación será dividida en dos partes de tal forma que, los usuarios que realicen la agrupación de tarjetas no vean como estas son construidas. De esta forma, el uso de la aplicación resultara más simple para estos. Con esta modificación también se logra que la aplicación pueda ser distribuida de forma Web. Con esto logramos que no sea necesario reunir a todos los usuarios para realizar la sesión, así hay menos problemática a la hora de reunir a los usuarios.

- Se añadirá en la parte de agrupación de tarjetas botones para elegir el idioma. Ya que hasta el momento, como era continuación de la creación de tarjetas, cogía por defecto el mismo idioma.

- Las tarjetas podrán mostrar, además de texto, iconos, videos y reproducir audio. Así se facilitará la comprensión de muchas de las tarjetas y a su vez podrán ser accesibles por personas con discapacidades como sordos o ciegos.

- Para elegir el tipo que queremos añadir introduciremos un combo box para facilitar la elección.

- Se adaptara el tamaño de las tarjetas para los nuevos componentes ya el tamaño de la anterior versión queda demasiado pequeño para estos.

- El fichero resultado de la sesión de Card Sorting debe ser compatible con la aplicación en la que se realiza el análisis de resultados.

- Monitorizar las sesiones de usuarios de tal manera que se registre información temporal sobre las acciones que los usuarios realizan a la hora de realizar la tarea de ordenación de tarjetas. De esta manera se pretende analizar las causas de posibles problemas a la hora de realizar dichas sesiones de ordenación de tarjetas.

Capitulo 5

Tecnología utilizada

En este capítulo se verán las características del lenguaje de programación sobre el que se ha implementado la aplicación, en este caso es sobre Java, así como los lenguajes utilizados para el almacenamiento y validación de la información XML, XSD y JDOM.
Para finalizar explicaremos el software que hemos utilizado para realizar el proyecto, se trata de un entorno integrado de desarrollo para Java, NetBeans.

5.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por James Gosling, de la empresa Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.
Los motivos que le llevaron a la creación de Java fueron:

- La necesidad de interfaces intuitivas para el usuario
- Fiabilidad de código y facilidad de desarrollo respecto a otros lenguajes de programación existentes en la época.
- El importante aumento de controladores electrónicos.

El primer proyecto de Java fracaso y fue a mediados de 1995, época de expansión de Internet, cuando se dieron cuenta de que este era ideal para la creación de software.

En 1996 se lanzo la primera versión de Java Development Kit (JDK 1.0) el software necesario para el desarrollo de programas en este lenguaje. A partir de aquí fueron saliendo sucesivas versiones del mismo y amentando considerablemente el número de usuarios.

5.2 NetBeans

Para realizar la implementación de esta aplicación hemos utilizado como plataforma NetBeans, que es un entorno de desarrollo (IDE). Hemos elegido NetBeans para este proyecto porque, entre las distintas opciones, esta es la que usa Sun.
NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

5.3 XML

XML son las siglas de eXtensible Markup Language (Lenguaje de marcado extensible). Son un conjunto de reglas para definir tags semánticas que componen un documento para poder identificar las diferentes partes de un documento. Es un lenguaje de meta-mercado que define una sintaxis para definir un dominio específico.

XML deriva del lenguaje de marcas SGML (estándar ISO) siendo a su vez un subconjunto de SGML que pretende que este pueda ser recibido y procesado en la web de la misma forma que HTML. A su vez, XML se diseño buscando la simplicidad de implementación con SGML i HTML.
Este metalenguaje nació en 1996 y fue desarrollado por un grupo de W3C, World Wide Web Consorcium. XML es un estándar de W3C.

5.4 XML Schema

Como indica su nombre es un lenguaje de esquema escrito en XML y utilizado para describir restricciones del contenido introducido en los documentos XML de una forma mas precisa. XML Schema (o XSD) surge como alternativa a las DTD, mas compleja con el objetivo de mejorar sus puntos débiles y definir estructuras en documentos XML La principal aportación de XML Schema es el gran numero de tipos de datos que incorpora. De esta forma, XSD aumenta las posibilidades y funcionalidades de aplicación de procesado de datos, incluyendo tipos de datos complejos.

Fue desarrollado por World Wide Web Consortium (W3C) en octubre de 2001. Esta edición no fue la que ahora utilizamos ya que en octubre de 2004 se reviso la segunda edición de XSD y es la que actualmente utilizamos.

5.5 DOM

DOM son las siglas de Document Object Model. DOM es esencialmente un modelo computacional a través del cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML. Su objetivo es ofrecer un modelo orientado a objetos para el tratamiento y manipulación en tiempo real, o de forma dinámica, a la vez que de manera estática de páginas de Internet. En este sentido, DOM es un API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como Javascript. El responsable del DOM es el consorcio W3C, *World Wide Web Consortium*.

5.6 JDOM

JDOM es una biblioteca de código fuente para leer, crear y manipular datos XML optimizados para Java. Surge para controlas APIs como DOM y SAX, las cuales fueron ideadas sin pensar en ningún lenguaje concreto.

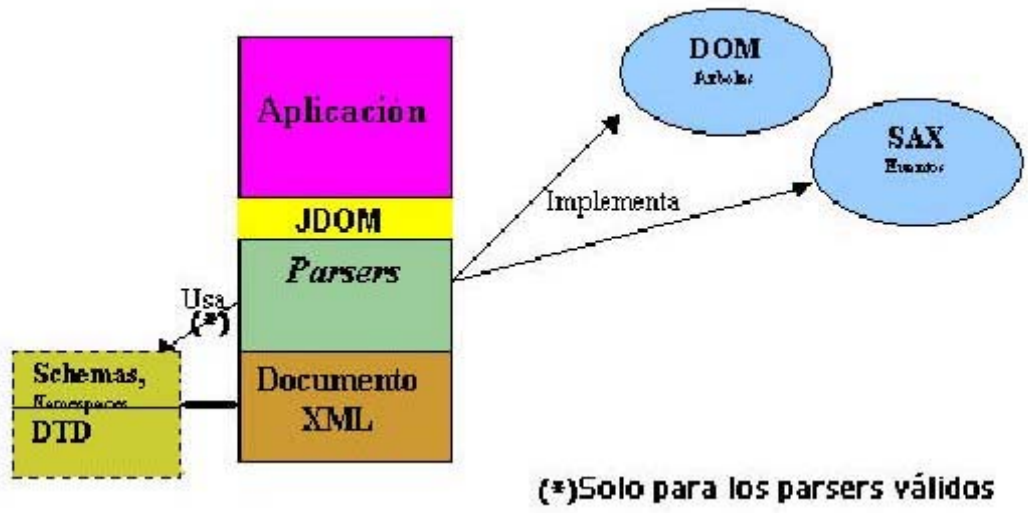


Figura 5: Estructura de JDOM

JDOM usa una parser para su trabajo. Este solo nos aporta una capa de abstracción en el tratado de documentos XML

El API de JDOM consta de 5 paquetes.

5.7 JDIC

JDIC (JDesktop Integration Component), es uno de los proyectos desarrollados dentro de la red de desarrolladores de Sun Microsystems. Está enmarcado dentro de las iniciativas de Sun para la integración completa del escritorio en aplicaciones Java. En este sentido, complementa tanto el aspecto visual nativo de Windows XP y GTK (herramienta abierta y multiplataforma para crear interfaces gráficas de usuario) que ya incluía Java 2 Standard Edition (J2SE), como las mejoras en integración de escritorios del nuevo J2SE 5.0.
JDIC ayuda a los desarrolladores a crear aplicaciones capaces de ejecutarse en múltiples sistemas operativos, además de ofrecer a los usuarios una experiencia consistente con sus entornos de escritorio.

5.8 JFC

JFC es la abreviatura de Java Foundation Classes, que comprende un grupo de características para ayudar a construir interfaces gráficos de usuario (GUIs).

Los componentes Swing
Incluye todo desde botones hasta splitpanes o tablas.

Soporte de Aspecto y Comportamiento Conectable
Le ofrece a cualquier componente Swing una amplia selección de aspectos y comportamientos. Por ejemplo, el mismo programa puede usar el Aspecto y Comportamiento Java o el Aspecto y Comportamiento Windows. Esperamos mucho más de los paquetes de Aspecto y Comportamiento -- incluyendo algo que use sonido en lugar de un 'look' visual.

API de Accesibilidad
Permite tecnologías asistidas como lectores de pantalla y display Braille para obtener información desde el interface de usuario.

Java 2D API (sólo JDK 1.2)
Permite a los desarrolladores incorporar fácilmente graficos 2D de alta calidad, texto, e imágenes en aplicaciones y applets Java.

Soporte de Drag and Drop (sólo JDK 1.2)
Proporciona la habilidad de arrastrar y soltar entre aplicaciones Java y aplicaciones nativas.

Las tres primeras características del JFC fueron implementadas sin ningún código nativo, tratando sólo con el API definido en el JDK 1.1. Como resultado, se convirtieron en una extensión del JDK 1.1. Esta versión fue liberada como JFC 1.1, que algunas veces es llamada 'Versión Swing'. El API del JFC 1.1 es conocido como el API Swing.

Capitulo 6

Implementación

En este capítulo serán expuestas las causas, resultados y justificaciones de este trabajo. Inicialmente explicaremos de forma simplificada la estructura y organización de este trabajo centrándonos mas tarde en aquellas clases que requieren una mayor atención y explicación. En el primer apartado se verá el contenido de cada uno de los paquetes que forma la aplicación y en el siguiente que hace cada clase y las modificaciones realizadas en cada una de ellas. El resto de apartados estudian en profundidad las modificaciones de este trabajo.

6.1 Estructura de la aplicación

En todo momento, se ha intentado mantener la estructura existente del Card Sorting. Esta aplicación ha sido dividida en dos partes. En la primera aplicación, "Diseño de Sesión de Card Sorting", se encuentra tanto la creación de tarjetas como la agrupación de las mismas. Y en la segunda aplicación, "Ejecución de Sesión de Card Sorting", solo podremos realizar la agrupación de las tarjetas.

En los siguientes apartados encontramos la distribución de paquetes en cada una de las aplicaciones.

6.1.2 Diseño de Sesión de Card Sorting

- **Editor:** Es el paquete en el que se encuentran contenidos todos los demás. En este podemos encontrar todas las clases que dan lugar a la interficie grafica, las ayudas del programa, la creación, modificación y eliminación de las tarjetas también se encuentra en este apartado.

- **Editor.EstructuresDades:** Aquí encontramos las clases en las cuales almacenaremos la información de todo aquello que se vaya creando. Por ejemplo, los datos con los que crearemos una tarjeta serán almacenados en Carta.java.

- **Editor.idiomes:** Esta carpeta da lugar a que nuestra aplicación pueda modificar el idioma en función de las necesidades del usuario. Encontramos tres archivos correspondientes a los tres idiomas que tiene la aplicación.

- **Editor.images:** En este paquete encontramos todas las imágenes que la aplicación utiliza para sus botones.

- **Editor.imprimir:** Como hay varias clases implicadas en la impresión de tarjetas, debido a que resulta una tarea costosa, las hemos agrupado todas en un mismo paquete.

- **Editor.ordenacio:** Corresponde a la representación la interface grafica de la parte de agrupar las tarjetas.

- **Editor.video:** Encontramos las plantillas para generar la pagina Web que será vista como video.

- **Editor.xml:** En este apartado se tratan los xml. Es decir, se crean y se leen, en función de la necesidad, los archivos xml.

- **Editor.xsdxml:** La validación de los xml, es decir, los xsd, se encuentran en este paquete.

6.1.2 Ejecución de Sesión de Card Sorting

- **Editor:** Es el paquete en el que se encuentran contenidos todos los demás. En este podemos encontrar todas las ayudas del programa y la creación de las tarjetas también se encuentra en este apartado.

- **Editor.EstructuresDades:** Aquí encontramos las clases en las cuales almacenaremos la información de todo aquello que se vaya creando. Por ejemplo, los datos con los que crearemos una tarjeta serán almacenados en Carta.java.

- **Editor.idiomes:** Esta carpeta da lugar a que nuestra aplicación pueda modificar el idioma en función de las necesidades del usuario. Encontramos tres archivos correspondientes a los tres idiomas que tiene la aplicación.

- **Editor.images:** En este paquete encontramos todas las imágenes que la aplicación utiliza para sus botones.

- **Editor.imprimir:** Como hay varias clases implicadas en la impresión de tarjetas, debido a que resulta una tarea costosa, las hemos agrupado todas en un mismo paquete.

- **Editor.ordenacio:** Corresponde a la representación la interficie grafica de la aplicación.

- **Editor.video:** Encontramos las plantillas para generar la pagina Web que será vista como video.

- **Editor.xml:** En este apartado se tratan los xml. Es decir, se crean y se leen, en función de la necesidad, los archivos xml.

- **Editor.xsdxml:** La validación de los xml, es decir, los xsd, se encuentran en este paquete.

6.2 Diagrama de clases

Una vez vista la distribución de paquetes que forman la aplicación, se explicará cual es la relación entre las clases. Dado que tanto la parte 1 como la parte 2 tienen las mismas clases, salvo algunas pequeñas modificaciones, centraremos este apartado en la parte 1, ya que esta contiene ambas partes.

En primer lugar, explicaremos cuales son las funciones heredadas para que el diagrama se entienda mejor.
La clase JFrame, representa un objeto ventana, es una extensión de la clase Frame del paquete AWT. En esta clase se base de toda la aplicación de ventanas.

La clase JDialog, igual que JFrame es un contenedor de alto nivel. Representa una ventana de dialogo dentro de una aplicación. Esta clase puede depender de otra como por ejemplo, JFrame.

Para finalizar, la clase JPanel, que a diferencia de las anteriores, es un contenedor de bajo nivel. Normalmente se suele introducir dentro de una ventana, como puede ser JFrame. Dentro de un JPanel podemos introducir, botones, texto, imágenes…

Llegados a este punto veremos el diagrama de la aplicación:

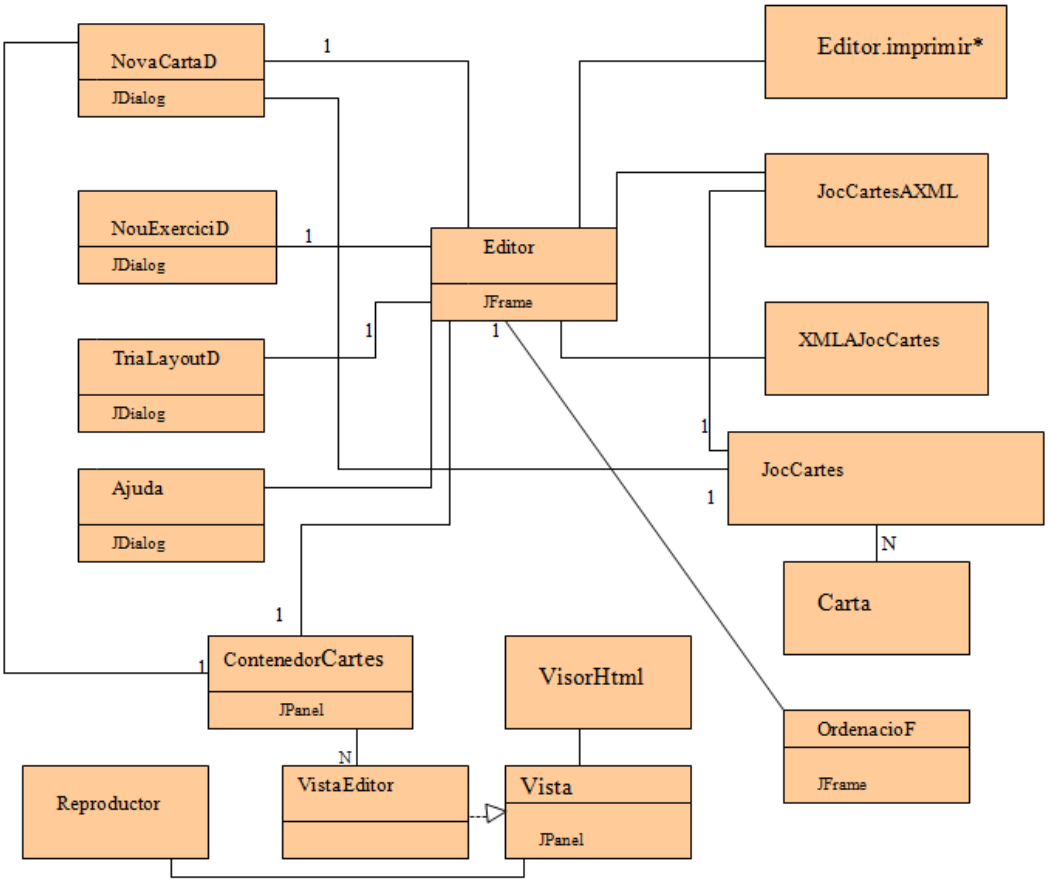


Figura 6: Diagrama de clases

Al ver el diagrama, observamos que la clase principal que representa el JFrame es Editor.

En la parte izquierda de la figura 5, podemos ver que las clases pertenecen a Editor. Estas representan la interface grafica de la aplicación. Podemos ver que algunas de estas clases son JDialog y otras JPanel.

Las JDialog realizan distintas funciones dentro del editor:

- **NouExerciciD:** muestra una pequeña ventana donde podremos elegir el tipo de juego de cartas que deseamos. Por ejemplo si solo queremos titulo y descripción o titulo y adjunto, etc. En esta clase se han modificado los tipos ya que hasta el momento solo se contemplaban las distintas combinaciones entre Título y Descripción.

- **NovaCartaD:** (ver apartado 5.3) Esta clase muestra una pequeña ventana donde podemos rellenar los datos para crear o modificar una nueva tarjeta. Aquí hemos realizado modificaciones para poder añadir en cada tarjeta el tipo de tarjeta y su adjunto correspondiente.

- **TriaLayoutD**: muestra una pequeña ventana donde podemos elegir si queremos que al realizar la agrupación de tarjetas estas, inicialmente, se encuentren aparrilladas, agrupadas o de forma aleatoria.

- **Ajuda:** completa las posibles ayudas. Esta clase no ha sido tratada en esta versión.

Las JPanel realizan las siguientes:

- **ContenedorCartes**: forma el panel en el que se encuentran contenidas las tarjetas. Se mantiene la misma clase que en el Card Sorting anterior.

- **Vista:** (apartado 5.6) Esta clase es la representación gráfica de cada una de las tarjetas. Se han creado dos clases adicionales "Reproductor.java" como "VistaHtml.java" son subclases de Vista creadas para mejorar la comprensión de la aplicación.

    - **Reproductor:** Realiza las operaciones correspondientes al audio. Por ejemplo, play.

    - **VisorHtml:** Genera la página Web en la que será reproducida el video.

Vista hace de clase base a VistaEditor y a VistaOrdenació tal como podemos observar en la figura siguiente:

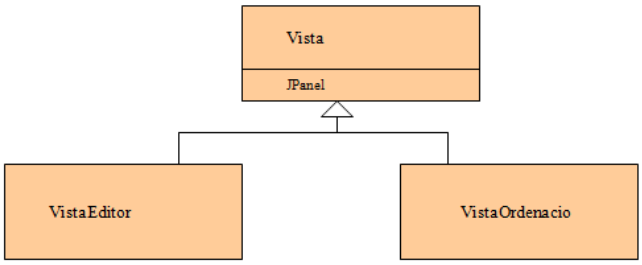


Figura 7: Diagrama clase Vista

- **VistaEditor:** Esta clase, igual que en la versión anterior de Card Sorting, se encarga de colorear las tarjetas cuando el usuario las selecciona.

- **VistaOrdenacio:** Las funcionalidades de esta clase solo las encontramos en la agrupación de tarjetas (Diseño y ejecución de la sesión de Card Sorting). Es la clase encargada de mover las tarjetas con el ratón y crear un grupo en el momento en el que estas se solapan. En esta clase se han realizado modificaciones para poder agrupar tarjetas de tipo video. (ver apartado 6.6.3)

- **EstructuraDades.Carta :** Esta clase almacena toda la información que tiene una carta: Título, Descripción, Tipo y Ruta. De esta forma siempre que necesitemos la información podemos acceder a ella.

- **EstructuraDades.JocCartes:** Representa un juego de cartas. Para ello, usa la clase Vector de java como contenedor de objetos de tipo Carta y dos enteros. Uno de los enteros nos indica el tipo de juego y el otro el número de cartas que componen el juego. Con esta clase nos es posible añadir, borrar y modificar una Carta. En esta versión no ha sido modificada esta clase.

- **EstructuraDades.ExerciciCS:** Esta clase, igual que en el Card Sorting anterior, guarda la información de una sesión de agrupación de tarjetas.

- **Ordenacio.OrdenacioF :** Esta clase se considera la clase principal de la agrupación de tarjetas. Es una ventana derivada de JFrame, en la que se realiza la ordenación. Esta clase es la que solicita y almacena el nombre del participante, controla los botones de esta aplicación. En la aplicación, Ejecución de la sesión de Card Sorting esta clase ha sido modificada para poder añadir las funcionalidades necesarias para poder realizar un cambio de idioma.
  Esta clase contiene un JLayedPane, que permite superponer componentes para poder realizar la creación de grupos.

- **Ordenacio.PanelOrdenacio:** Es la clase derivada del JLayedPane. Esta clase es la encargada de gestionar la estructura de ExerciciCS.

- **Ordenacio.EtiquetaGrup:** Esta clase deriva de un JPanel y en este trabajo no ha sido modificada. Permite mover un grupo de tarjetas dentro del PanelOrdenacio. Una vez llegamos al segundo paso, esta es la clase que nos permite asignar un nombre a cada grupo.

- **Xml.ExerciciCSAXML:** (ver apartado 5.3) Esta clase genera el xml tras la agrupación de tarjetas. Ha sido necesario añadir nueva información como el tipo de tarjeta o el nombre del archivo contenido en esta.

- **Xml.JocCartesAXML:** (ver apartado 5.3) Genera el xml tras la creación de tarjetas. Ha sido necesario añadir nueva información como el tipo de tarjeta o el nombre del archivo contenido en esta.

- **Xml. XMLAJocCartes:** (ver apartado 5.3) Esta clase lee el xml anterior y vuelve a generar las tarjetas con la información almacenada en el xml. Se han contemplado los nuevos valores incluidos en el xml.

6.3 Gestión de datos

En distintos momentos la aplicación debe almacenar los datos o resultados para más tarde utilizarlos. Este puede ser el caso de:

- Abrir un juego de tarjetas
- Guardar juego de tarjetas
- Guardar los resultados obtenidos de un juego de tarjetas

En la primera versión de Card Sorting esto se hacía mediante DTD.
Cuya función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD.
Actualmente, ha surgido XSD como alternativa a DTD. XSD es considerada una mejora de este último y por este motivo se ha considerado necesario realizar esta modificación en la aplicación.

6.3.1 Guardar juego de tarjetas

En primer lugar creamos un XSD, que nos servirá para validar mas tarde los valores del XML. Para ello es necesario tener muy presentes los atributos que forman el XML que será validado contra este.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="JocDe">
         <xs:complexType>
            <xs:sequence>
          <xs:element name="JocDeTargetes" type="xs:string"/>
          <xs:element name="Targeta">
             <xs:complexType>
                <xs:sequence>
                  <xs:element name="Titol" type="xs:string"/>
                  <xs:element name="Descripcio" type="xs:string"/>
                  <xs:element name="Tipo" type="xs:string"/>
                  <xs:element name="Ruta" type="xs:string"/>
                </xs:sequence>
             </xs:complexType>
          </xs:element>
          </xs:sequence>
          </xs:complexType>
        </xs:element>

</xs:schema>
```

A continuación, se genera desde **JocCartesAXML.java**, situado dentro de la carpeta xml, el archivo XML con los datos de nuestro juego de tarjetas.
Y mediante la función 'SaxBuilder' le indicamos que realice la validación con xsd.

```
….

SAXBuilder builder = new SAXBuilder(true);
 builder.setProperty(
                "http://apache.org/xml/properties/schema"
               + "/external-noNamespaceSchemaLocation", "JocDeTargetes.xsd" );
…
```

Llegados a este punto nuestra información queda correctamente guardada en un fichero del tipo ".jdt". Para cada juego de tarjetas habrá dos copias. Uno de estos archivos se guardará donde nosotros le indiquemos y la otra en la carpeta en la que la aplicación ha ido guardando todos los adjuntos de la aplicación.

6.3.2 Abrir juego de tarjetas

Esta situación podemos encontrarla en dos casos. Cuando queremos abrir un juego de tarjetas ya sea para añadir más tarjetas, para modificarlas o cuando hay que ejecutar la segunda parte de la aplicación, la realización del juego de tarjetas.

Esta parte se realiza desde **XMLAJocCartes.java.** Esta clase consiste en leer el XML creado previamente para poder cargar el juego de tarjetas con los valores que se ven reflejados en el archivo.

6.3.3 Guardar resultados obtenidos en un juego de tarjetas

Una vez finalizada la agrupación de tarjetas es necesario almacenar los resultados para más tarde poder sacar conclusiones o estadísticas de los mismos. Para ello realizaremos los mismos pasos que en el apartado 5.1.1 pero en este caso los campos a almacenar serán un tanto distintos. En este caso la información la coge de EstructuraDades.ExerciciCS. El resultado que obtenemos es un fichero del tipo ".ecs".
Se ha considerado oportuno añadir nuevos campos para conocer el orden y el momento en el que el usuario trata cada una de las tarjetas. (ver apartado 6.8)

6.4 Creación de una tarjeta

En este apartado nos centraremos especialmente en las modificaciones realizadas al pretender crear una nueva tarjeta, **NovaCartaD.java.** Siendo esta la pantalla de dialogo, JDialog, que se muestra cuando pretendemos añadir una nueva tarjeta en nuestro juego de cartas.

6.4.1 Apariencia

Ya que el objetivo fundamental de este trabajo es poder incorporar un componente en la tarjeta (video, audio o imagen) hemos considerado oportuno realizar una ampliación de las tarjetas.
Previamente estas simplemente constaban de titulo y descripción por lo que no era necesario tener una tarjeta muy grande. Ahora, al añadir la incorporación de dichos componentes las tarjetas han sido ampliadas para conseguir así una mejor percepción de las mismas.

Debido a que una tarjeta puede ser de tipo imagen, audio, video o texto se ha considerado crear un combo box, mediante la función JComboBox(), para que el usuario pueda seleccionar el tipo de tarjeta que desee.

En el caso de que nuestra selección sea un tipo “texto” la aplicación no sufre ninguna modificación respecto a la primera aplicación de Card Sorting.

Justo debajo del combo box se encuentra el campo ruta cuya función será mostrar, para luego guardar, el nombre del fichero que muestre esta tarjeta.

El campo ruta se encuentra siempre deshabilitado de tal forma que un usuario debe seleccionar siempre el botón que se encuentra junto al mismo para realizar la búsqueda del adjunto.
Al seleccionar dicho botón se nos carga un JFileChooser (ver figura 7) para que podamos realizar la búsqueda del archivo que deseamos incluir en nuestra tarjeta.

Dependiendo del tipo de tarjeta que hayamos seleccionado tendremos una restricción u otra de los archivos que podemos incluir. Es decir, si el tipo es video solo podremos cargar archivos flash, “.swf”.

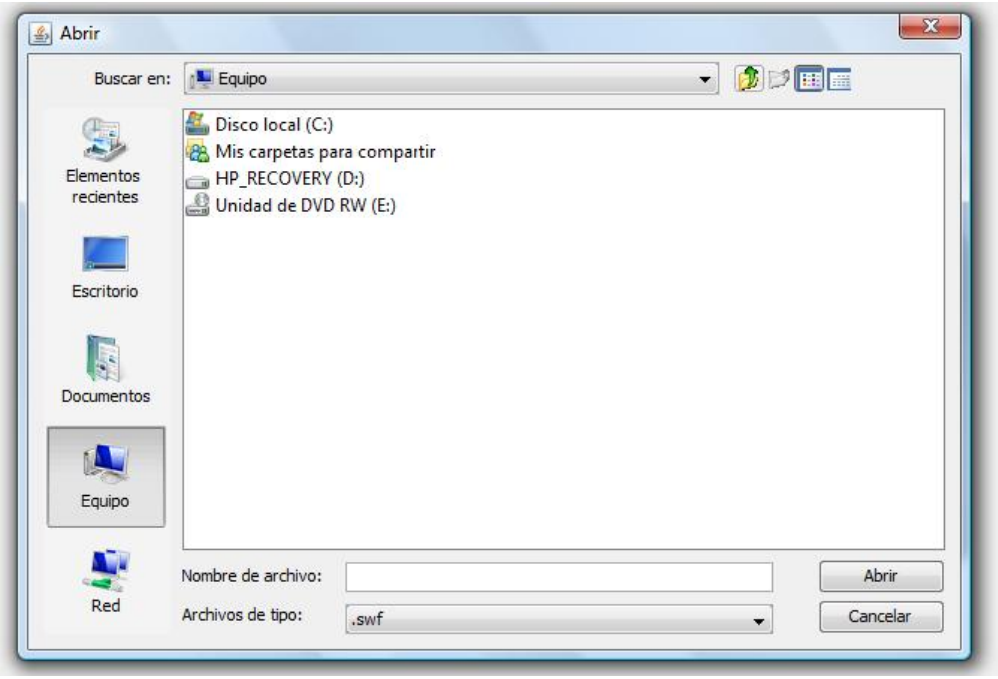


Figura 8: JFileChooser

Una vez seleccionada la ruta en la que se encuentra el video, imagen o audio que deseemos añadir a nuestra tarjeta, este se copia en una carpeta, que ha sido creada al ejecutar el programa, para así poder transferir más tarde todos los archivos utilizados para crear las tarjetas a la segunda parte de la aplicación. Es decir, que teniendo todos los archivos en una misma tarjeta podemos enviarlos junto al conjunto de tarjetas de una forma más simple.

6.5 Modificación de una tarjeta

Una vez creada una o varias tarjetas cabe la posibilidad de realizar una modificación. Para hacerlo solo hay que dar doble click sobre la tarjeta a modificar o bien, seleccionar la tarjeta y apretar el botón modificar (ver Figura 8). En ambos casos se carga una pequeña ventana idéntico al de crear una nueva tarjeta pero con los campos completados con la información correspondiente a la tarjeta a modificar.

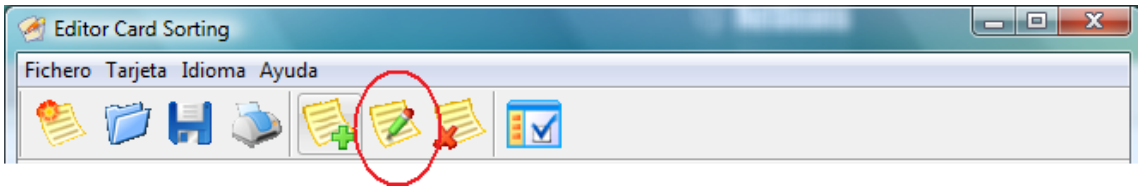


Figura 9: Botón para modificar tarjetas

6.5.1 Modificación

En la anterior versión de Card Sorting, ya que esta solo constaba de titulo y descripción, ambos de tipo texto, al realizar una modificación de la tarjeta solamente era necesario realizar una modificación de estos.
En nuestro caso, ya que el usuario puede modificar el tipo de tarjeta, no es suficiente con sobre escribir las tarjetas. Cada tipo de tarjeta tiene una distribución totalmente distinta respecto al resto.

En primer lugar, en NovaCartaD.java, se completan los campos de la ventana con la información correspondiente a la tarjeta seleccionada para su modificación.

Realizamos las modificaciones oportunas, incluso si es necesario modificar el tipo de tarjeta. Al seleccionar el botón de aceptar, detectamos que se trata de una modificación y en lugar de sobre escribir como se hacía antes, ahora es conveniente eliminar la tarjeta anterior y añadir una con las modificaciones realizadas. De este modo, la visión es la correspondiente a nuestra modificación y en ningún caso nos encontraremos dos componentes en la misma tarjeta.

6.6 Componentes de una tarjeta

Como ya se ha explicado previamente, el objetivo fundamental de este trabajo es incorporar en las tarjetas de Card Sorting componentes como pueden ser Imágenes, Videos y Audios.
Sin olvidar que estas también pueden ser simplemente de tipo texto, constando simplemente de un título y una descripción. Todas las tarjetas, sean del tipo que sean, constan de título y descripción. El tamaño de la descripción varía en función del tipo de componente que sea añadido.

La clase que crea las vistas y pre visualizaciones de una tarjeta es **Vista.java.** Ya que en función del tipo de tarjeta la distribución de esta será de una forma u otra. Esta clase realiza una comprobación para saber de qué tipo se trata. Una vez conocido el tipo se asigna distribución correspondiente al tipo de tarjeta.

6.6.1 Imágenes

Para poder ver una tarjeta con un componente tipo imagen fue necesario recurrir a la clase ImageIcon(). Esta nos permite ver una imagen de un determinado tamaño como componente de nuestra tarjeta.
Debido al tamaño reducido de las tarjetas las imágenes deben tener un tamaño máximo de 190x50.

En este caso la descripción se verá en la parte inferior de la tarjeta.

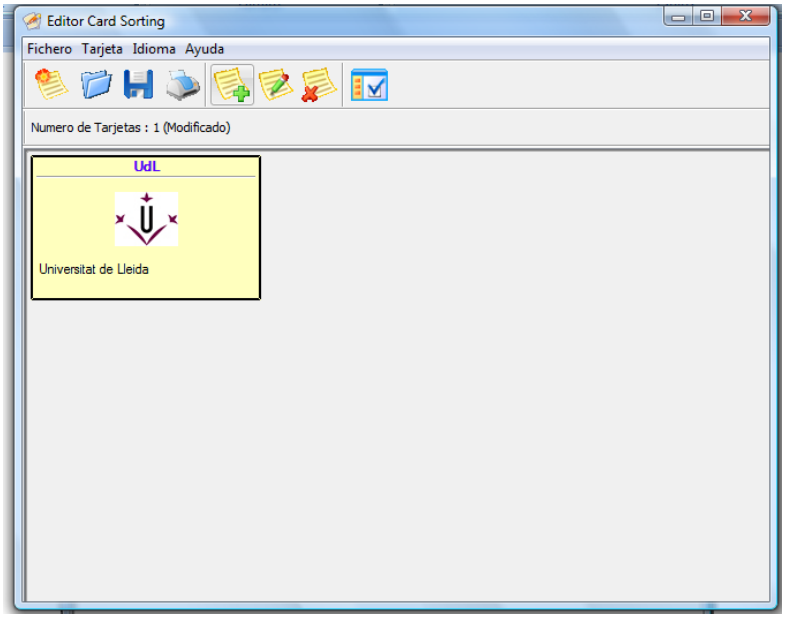


Figura 10: Tarjeta de tipo imagen

6.6.2 Audio

Nuestras tarjetas son capaces de reproducir un archivo audio. Para conseguirlo se ha recurrido a la clase *BasicPlayer* perteneciente a la librería jlGui. Esta clase nos permite reproducir cualquier archivo audio ya sean de formato mp3 o wma.

Las tarjetas de tipo audio están compuestas, además del título y la descripción, de dos botones para poder controlar las funcionalidades típicas de un reproductor. Estos dos botones, uno de play y el otro de stop, que nos permiten una mayor facilidad de manejo.
Las acciones de este tipo de tarjeta están implementadas en la clase **Reproductor.java**.

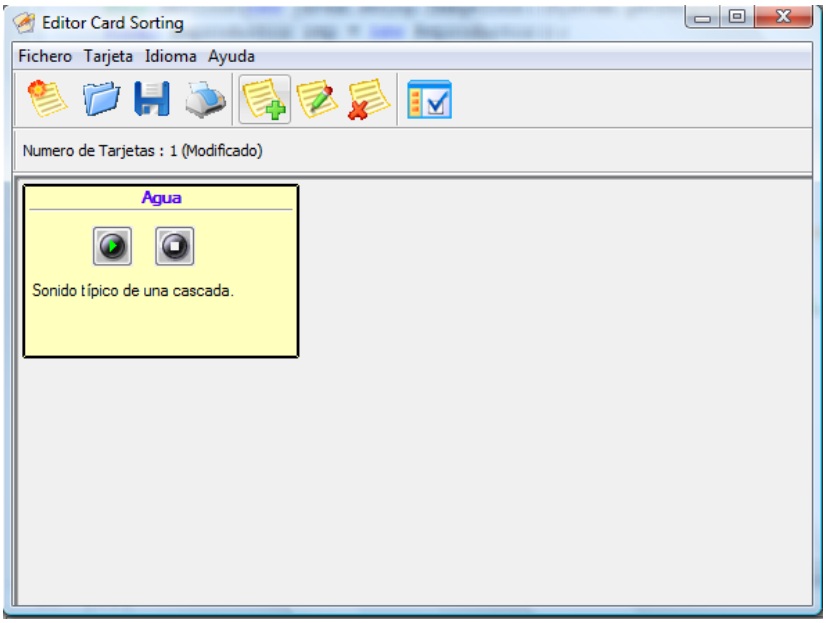


Figura 11: Tarjeta de tipo audio

6.6.3 Video

El último de los componentes que puede tener una tarjeta es un video.
Inicialmente se tanteo entre distintas posibilidades para plasmarlo en una tarjeta.
Entre las distintas opciones estaban JMF, JFlashPlayer o JDIC.
La primera opción JMF, nos la ofrece Sun por defecto en Java. El problema era que los videos debían estar creados con un codec específico lo cual resultaría mas tarde un problema.
La segunda opción JFlashPlayer, es un conjunto de API Java que permite a los desarrolladores de jugar e interactuar con Adobe Flash Player películas dentro de las aplicaciones Java. El problema está en que sus licencias son de pago.
Finalmente nos decantamos por JDIC. Web Browser, componente de JDIC, es una aplicación software que le permite a un usuario visualizar e interactuar con imágenes, videos, música y otro tipo de información situada en una página Web. Es decir, que Web Browser nos permite incorporar en nuestra aplicación un componente Web cuyo tipo puede ser cualquiera de los citados.
Se intentó incorporar en una página Web un reproductor para poder pasarle cualquier tipo de formato video: .avi, .mov, etc. Esto dio resultado pero al no poder modificar el tamaño del reproductor lo suficiente, la tarjeta en la que aparecería un video sería mucho más grande que el resto. Para evitar esto se debería haber aumentado considerablemente el tamaño de todas las tarjetas, desvirtuando en gran medida el proceso de Card Sorting al suponer necesariamente superponer elementos por falta de espacio en la pantalla para poder representarlos de manera separada.

Estos son los motivos por los que se considero como mejor opción, incorporar en la página Web simplemente el video, sin reproductor. Pero esto nos limita a poder reproducir simplemente videos flash, es decir, formato swf.

Web Browser no queda integrado en un JPanel, lo cual significa que al intentar agrupar tarjetas sobre una de tipo video este siempre permanecía visible. Para evitarlo, los videos no se muestran directamente sobre el panel que en esencia es la tarjeta. Al crear la página que contiene el video, este es plasmado sobre un JPanel. Este será el componente de la tarjeta, que a su vez es un JPanel.
Se ha creado un botón de stop cuya función es ocultar el JPanel que contiene el video. De este modo el video dejara de estar visible cuando el usuario desee y sobretodo, cuando nos sea conveniente; por ejemplo, en el momento de agrupar las tarjetas.

Por lo tanto, al detectar que la tarjeta es agrupada llamamos a la función “StopVideo” y este deja de ser visible. Si mas tarde el usuario desea desagrupar la tarjeta y reproducir el video, esto es posible ya que la tarjeta tiene un botón de play para poder reprocir el video.

También había problemas al intentar mover una tarjeta tipo video con este en reproducción. Web Browser no refresca la reproducción quedándose el video colgado y al intentar volver a reproducirlo este no respondía. Por ello, al mover una tarjeta de tipo video, si este está en reproducción, este se oculta y posteriormente éste puede volver a ser reproducido.

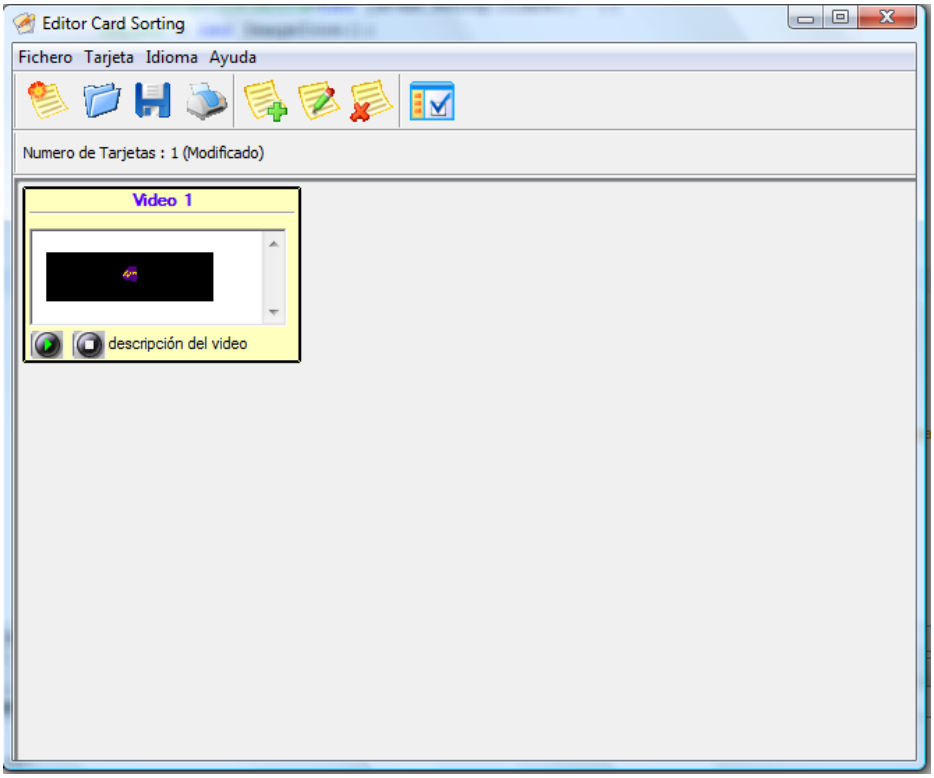


Figura 12: Tarjeta de tipo video

En la clase **VisorHtml.java** donde se genera la pagina Web correspondiente al video seleccionado. Se crea una página Web con el mismo nombre que el video y se copia el código de una plantilla creada previamente. De esta forma y pasándole el video que

hemos seleccionado se genera la pagina a reproducir. Es en esta clase también, donde se reproduce esta.

En este tipo de tarjetas y como bien muestra la imagen anterior la descripción puede ser de un tamaño mucho menor respecto al resto de tipos.

6.7 Separación de la aplicación

Como ya se ha mencionado a lo largo del trabajo, se ha considerado necesario realizar una partición de la aplicación inicial. Así, se consigue que el usuario que realiza la agrupación de las tarjetas no vea la construcción de las mismas, consiguiendo que para este la aplicación sea más fácil de utilizar.

6.7.1 Diseño de Sesión de Card Sorting

Esta parte consiste en la creación de las tarjetas. El usuario que desee realizar el juego de tarjetas deberá recurrir a esta parte. Esta, consta de la aplicación que genera las tarjetas, completadas por el mismo usuario y también se ha considerado oportuno que este pueda ver y agrupar la segunda parte, la agrupación de tarjetas.
De esta forma, el usuario que use esta aplicación tendrá una visión completa de lo que es en sí el Card Sorting.

6.7.2 Ejecución de Sesión de Card Sorting

Esta es la parte en la que un usuario realizara la agrupación que considere oportuna de las tarjetas creadas en el Diseño de Sesión de Card Sorting.
Esta aplicación, a diferencia del Card Sorting inicial, es independiente a la creación de tarjetas y por este motivo se han tenido que tratar algunas modificaciones respecto a lo que puede ser la unión de tarjetas que agrupa la primera parte.

En primer lugar nos centramos en la parte del idioma ya que hasta el momento solo era posible realizar modificaciones de idioma en la creación de tarjetas cogiendo toda la aplicación ese mismo idioma.

Al realizar la separación, se ha considerado que el usuario que crea el juego de cartas no debe elegir el idioma de la segunda parte. Este no sabe qué idioma hablará cada uno de los participantes. Por esto, se han creado unos botones en esta aplicación para que cada participante pueda seleccionar el idioma que en el que se sienta más cómodo.

Figura 13: Botones de idioma en Ejecución de Sesión Card Sorting

También es necesario que esta página reciba la carpeta en la que se han ido almacenando, durante la creación de las tarjetas, los archivos video, audio e icono.

Estos archivos, como ya se ha comentado anteriormente, están guardados en una carpeta creada por defecto al construir las tarjetas. Su ruta es "C:\CS".
Una vez guardamos un juego de cartas se ejecutamos el comando:

➔ C:\"Program Files"\Java\jdk1.6.0_10\bin\jar uf parte2.jar CS

Este comando copia la carpeta CS, en la cual están los archivos utilizados, dentro del ejecutable correspondiente a la sesión de agrupación de tarjetas. De esta forma las tarjetas, en esta parte, podrán mostrar los distintos archivos.

De esta forma el ejecutable se puede expandir tipo Web.

6.8 Monitorización de las sesiones de usuarios

Se ha considerado oportuno, para conocer mejor a los usuarios participantes en la sesión de Card Sorting, monitorizar las acciones de los mismos.
Estas modificaciones se han realizado en ExerciciCSAXML.java situado dentro de la carpeta xml.
Para ello guardamos en cada momento lo que este está haciendo, es decir, quedan reflejados los movimientos de tarjetas aunque estos no sean los definitivos. Con esto podemos ver las posibles dudas o modificaciones de cada usuario. Además, cada una de estas acciones viene acompañada por el momento en que se ha realizado.

Con esto, conocemos el tiempo que ha tardado cada usuario en agrupar una tarjetas las modificaciones que ha realizado y podemos realizar un análisis mucho mas detallado sobre los resultados de la sesión.

Capitulo 7

Pruebas realizadas

Para comprobar que la aplicación funciona correctamente realizaremos una serie de pruebas y verificaremos su correcto funcionamiento.
Esta serie de pruebas están centradas especialmente en el funcionamiento con respecto a los tipos de tarjetas. Los distintos tipos de tarjetas deben funcionar correctamente.
Para afirmarlo realizaremos una prueba para cada tipo de tarjetas en el que únicamente aparezcan tarjetas de dicho tipo. Finalmente, realizaremos una prueba en la que haya una mezcla de los distintos tipos.

Ha sido comprobado, mediante las siguientes pruebas el correcto funcionamiento de la aplicación. Cada una de estas ha sido ejecutada en la aplicación de Diseño de Sesión de Card Sorting y en Ejecución de Sesión de Card Sorting para así tener un resultado global de toda la aplicación.

7.1 Prueba 1

En esta primera prueba nos centraremos en las tarjetas de tipo texto. Creamos 10 tarjetas con las distintas posibilidades: solo título, solo descripción y ambas.

Guardamos este juego de tarjetas y lo volvemos a abrir para comprobar que funciona correctamente. En el caso de que el usuario haya seleccionado como prototipo para la tarjeta cualquier opción en la que no aparezca el título o la descripción y desee abrir el juego que acabamos de guardar, estas se cargan tal como las habíamos guardado.

Modificamos tarjetas de la siguiente forma:

- Modificar a tipo audio.
- Modificar a tipo video.
- Modificar a tipo imagen.
- Modificar titulo.
- Modificar descripción
- Añadir titulo
- Añadir descripción

Hay que tener en cuenta que nunca podremos tener una tarjeta sin titulo ni descripción. Siempre debe aparecer uno de los dos componentes.

Eliminamos dos tarjetas para verificar el funcionamiento.

Para terminar debemos comprobar que la Sesión de agrupación también funciona correctamente.
Llegados a este punto podemos afirmar que esta prueba ha tenido resultados satisfactorios.

7.2 Prueba 2

Este juego de pruebas será muy similar al anterior. En este caso crearemos 10 tarjetas de tipo imagen. Es conveniente que las imágenes no se repitan.
Debemos crear tarjetas distintas, es decir, deben aparecer tarjetas que únicamente tengan título o descripción y también tarjetas en la que aparezcan los dos campos completos.

Realizaremos las siguientes modificaciones en las tarjetas:

- Modificar a tipo texto
- Modificar a tipo video
- Modificar a tipo audio
- Modificar titulo
- Modificar descripción
- Añadir descripción
- Añadir titulo

A continuación, eliminamos 3 tarjetas. Preferentemente, las tarjetas deben ser distintas, es decir, es conveniente que no eliminemos dos tarjetas que tengan, por ejemplo, título y descripción.

Finalmente, realizamos la agrupación de tarjetas y verificamos que el archivo de salida es correcto. Para comprobarlo, utilizamos la aplicación de análisis de resultados.

7.3 Prueba 3

Para esta prueba, nos centramos principalmente en las tarjetas de tipo video. Para ello, creamos 10 tarjetas de tipo video. Es conveniente que todos los videos sean distintos.
Igual que en los anteriores casos, crearemos tarjetas en las que únicamente aparezca título o descripción y tarjetas con los dos campos completados.

Activamos los videos y comprobamos que el funcionamiento de sus botones es el correcto.

A continuación realizamos las siguientes modificaciones:

- Modificar tarjeta a tipo texto
- Modificar tarjeta a tipo audio
- Modificar tarjeta a tipo imagen
- Modificar titulo
- Modificar descripción
- Añadir titulo
- Añadir descripción

Una vez realizadas las modificaciones procedemos a eliminar 2 tarjetas.

Comprobamos que en la sesión de agrupación de tarjetas al desplazar una tarjeta con el video en funcionamiento este deja de ser visible. También comprobamos que al dejar la tarjeta en un determinado lugar, podemos volver a activar el video.
Para finalizar, verificamos que al realizar un grupo en el que uno o varios videos estén activos estos también se ocultan.

7.4 Prueba 4

Para esta prueba creamos 10 tarjetas de tipo audio. Es importante saber, que para crear una tarjeta de tipo audio no puede haber ninguna tarjeta audio en funcionamiento. También es importante que los archivos audio no se repitan en ninguna tarjeta.
Comprobamos que los botones funcionan correctamente.

Realizamos las siguientes modificaciones:

- Modificar a tipo texto
- Modificar a tipo imagen
- Modificar a tipo video
- Modificar titulo
- Modificar descripción
- Añadir titulo
- Añadir descripción

Comprobamos que el botón de eliminar tarjetas realiza su función correspondiente.
Para terminar, comprobamos que la agrupación de tarjetas funciona correctamente.

7.4 Prueba 5

En esta ultima prueba, crearemos tarjetas de distintos tipos. Crearemos 10 tarjetas de los distintos tipos:

- 3 tarjetas tipo texto
- 3 tarjetas tipo imagen
- 4 tarjetas tipo video

- 3 tarjetas tipo audio

En cada uno de los tipos crearemos una tarjeta con solo título, una con solo descripción y una o dos con titulo y descripción.

Realizamos las siguientes modificaciones:

- Modificamos de tipo texto a tipo imagen
- Modificamos de tipo texto a tipo video
- Modificamos de tipo texto a tipo audio
- Modificamos de tipo imagen a tipo texto
- Modificamos de tipo imagen a tipo video
- Modificamos de tipo imagen a tipo audio
- Modificamos de tipo video a tipo texto
- Modificamos de tipo video a tipo imagen
- Modificamos de tipo video a tipo audio
- Modificamos de tipo audio a tipo texto
- Modificamos de tipo audio a tipo imagen
- Modificamos de tipo audio a tipo video
- Modificamos un titulo
- Modificamos una descripción

Eliminamos alguna tarjeta. Es importante no eliminar 3 tarjetas del mismo tipo ya que necesitamos tener todos los posibles tipos en la agrupación de tarjetas para comprobar su funcionamiento.

Capítulo 8

Conclusiones y trabajos futuros

La elaboración de este proyecto me ha ayudado a adquirir y comprender mejor las técnicas de programación java, ya que mi nivel de programación no era muy elevado. También me ha gustado el haber podido aplicar las técnicas y clases aprendí a lo largo de mis estudios, como por ejemplo xml.

Pero este proyecto, no solo me ha servido en temas de programación, también he podido comprender mejor las técnicas de usabilidad y accesibilidad centradas en la informática. Hasta el momento desconocía la existencia de estos métodos como puede ser la del Card Sorting, una técnica que se viene realizando desde hace mucho tiempo y cuya utilidad puede ser muy productiva.

Como trabajo futuro estaría bien, incluir un soporte de la API de accesibilidad de Java en esta aplicación. En este caso esta tarea sería costosa debido a componentes externos como WebBrowser.

También, como trabajo futuro estaría bien pasar la aplicación a Web. De esta forma, se conseguiría que un mayor número de personas utilizasen Card Sorting para sus proyectos.

Bibliografía

- Porta Miret, Eduard (2005). Treball Final de carrera. Implementació d'un Editor de Card Sorting multiplataforma. Universitat de Lleida.

- Pardell Mateu, Daniel (2006).Treball Final de carrera. Eina multiplataforma per analitzar els resultats en exercicis de Card Sorting. Universitat de Lleida.

- Granollers Saltiveri, Toni. Modelo de Proceso de la Ingeniería de la usabilidad y de la accesibilidad. http://griho.udl.es/mpiua/modelo.htm

- Granollers Saltiveri, Toni (2004). Tesis Doctoral : *"MPIu+a. Una metodología que integra la ingeniería del software, la integración persona-ordenador y la accessibilidad en el contexto de equipos de desarrollo multidisciplinarios"*

- Página principal de Java. http://java.sun.com/

- Hassan Montero, Yussef; Martín Fernández, Francisco J. (2004). Card Sorting: Técnica de categorización de contenidos. http://www.nosolousabilidad.com/hassan/cardsorting.pdf

- Kotonya, Gerald; Sommerville, Ian. *Requirements Engineering: Processes and Techniques*

- Maurer, Donna; Warfel, Todd Card Sorting: a definitive guide. http://www.boxesandarrows.com/view/card_sorting_a_definitive_guide

- Brett McLaughlin *Java & XML, 2nd Edition (2001)* O'Reilly

- Nielsen, Jacob, 2004. *Card Sorting: How Many users to Test*. Disponible a: http://www.useit.com/alertbox/20040719.html

- Nielsen, J. (1993). *Usability Engineering.* Academic Press Professional, Boston, MA.

- Norman, D. (1990). *The design of everyday things.* Doubleday, Nova York.

- Robertson, James.(2002) *Information Design Using Card Sorting*, http://intranetjournal.com/articles/200202/km_02_05_02a.html
- Rosenfeld, L.; Morville, P. (2002). *Information Architecture for the*

World Wide Web. O'Reilly (2nd ed.).

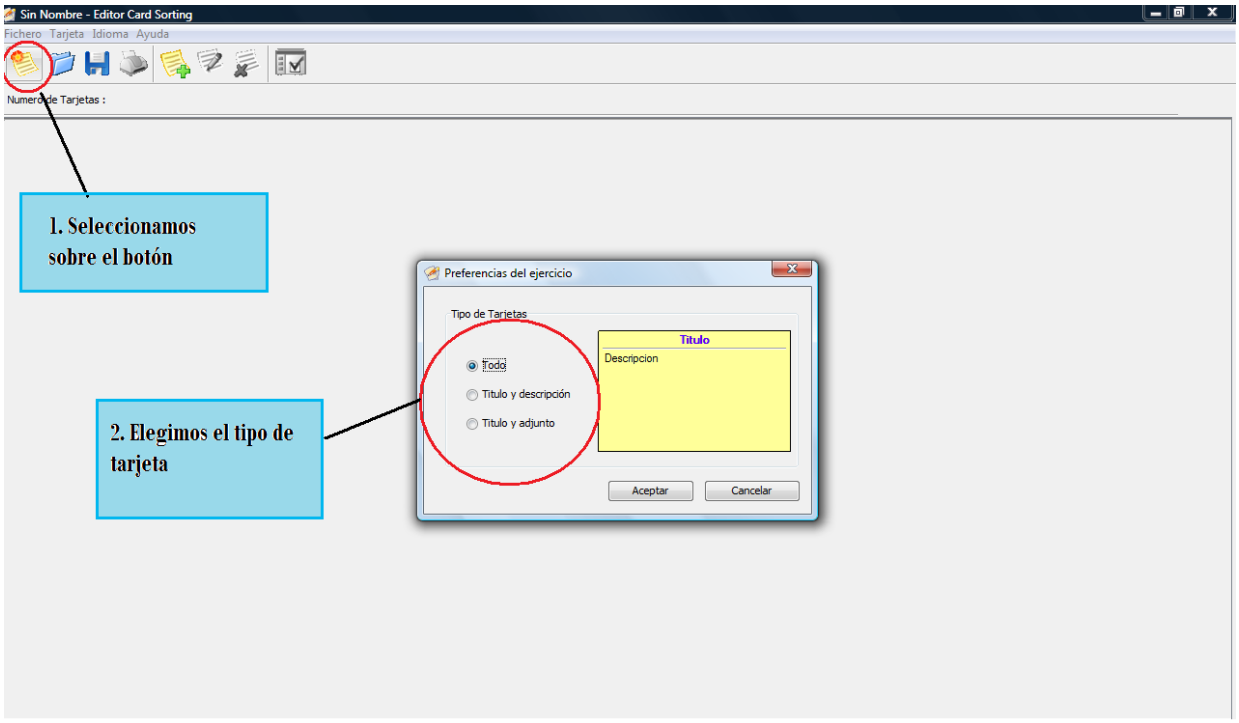
- Saul Wurman, Richard, (1997). *Information Architects*. Watson-Guptill.
- Spool J.M ,Bailey, R.; Molich, R.; Dumas, J. (2002). *Usability in Practice: Formative Usability Evaluations*. CHI2002 Proceedings. ACM.
- Thimbleby, H. (1990). *User Interface Design.* Addison-Wesley, Reading, MA.
- Tullis, T., and Wood, L. (2004), "*How Many Users Are Enough for a Card-Sorting Study?*" Proceedings UPA'2004
- Página Principal del proyecto JDOM: http://www.jdom.org/
- DOM según W3C: http://www.w3.org/DOM/
- Especificación de Modelos de Objetos del Documento Lavel 1. http://html.conclase.net/w3c/dom1-es/cover.html
- Conjunto de información XML (Segunda edición) http://www.spanish-translator-services.com/espanol/t/infoset.htm
- XML Shema Part 0 Primer http://www.w3.org/TR/xmlschema-0/
- Programa de verificación del Schema http://java.sun.com/webservices/

- Anexo -

Funcionamiento de las aplicaciones de Diseño y Ejecución de Sesión de Card Sorting

En este explicaremos los pasos y acciones que podemos realizar en el Card Sorting.

Crear juego de tarjetas



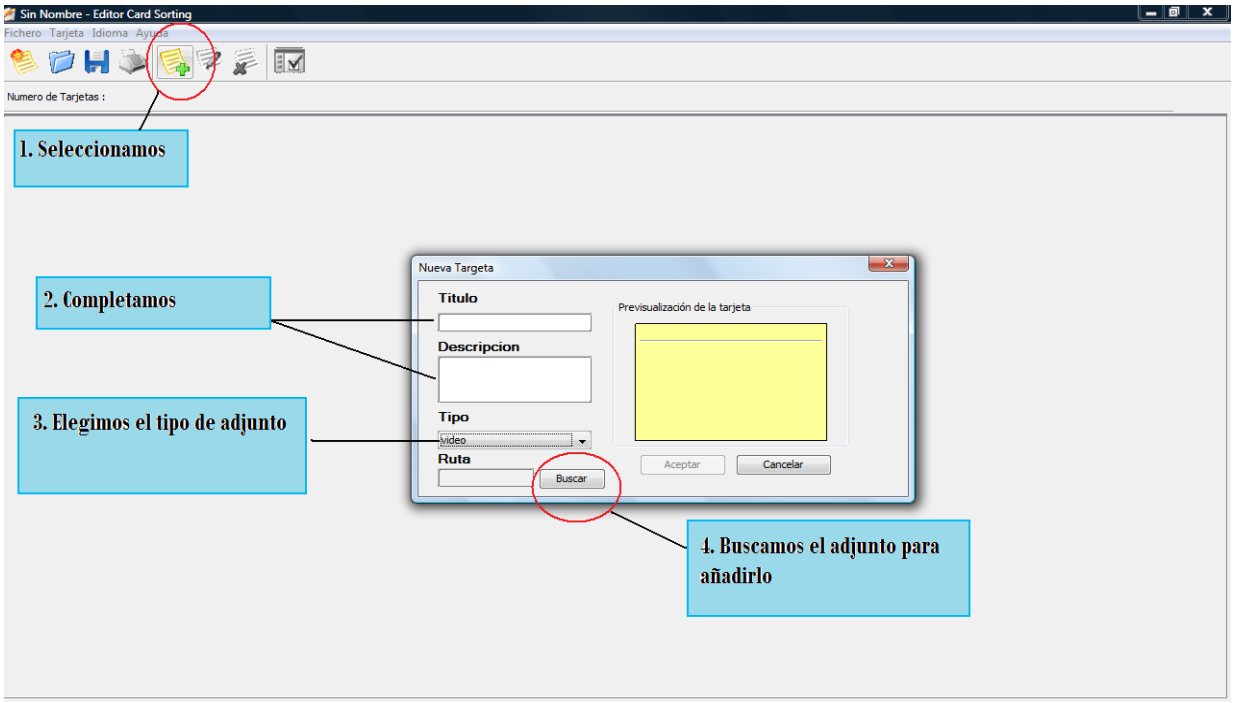
Como tipo de tarjeta podemos elegir:

- **Todo:** Las tarjetas estarán formadas por Título, descripción y el tipo de adjunto que elijamos (audio, video o imagen)
- **Titulo y descripción:** Las tarjetas estarán formadas por título y descripción.
- **Titulo y adjunto:** Las tarjetas no podrán tener una descripción.

El tipo de tarjetas que seleccionemos será para todas las tarjetas del juego. Es decir, durante la creación de tarjetas no podrá ser modificado.

Añadir tarjetas

Para generar cada una de las tarjetas seguiremos los pasos deleitados a continuación.

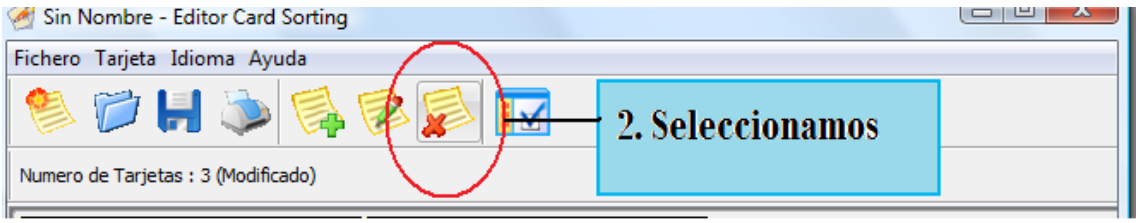


Modificar tarjetas

1. En primer lugar, seleccionamos la tarjeta que va a ser modificada.
2. Hacemos doble clic sobre la misma tarjeta.
3. Realizamos las modificaciones oportunas. Estas también pueden ser el tipo y el adjunto.

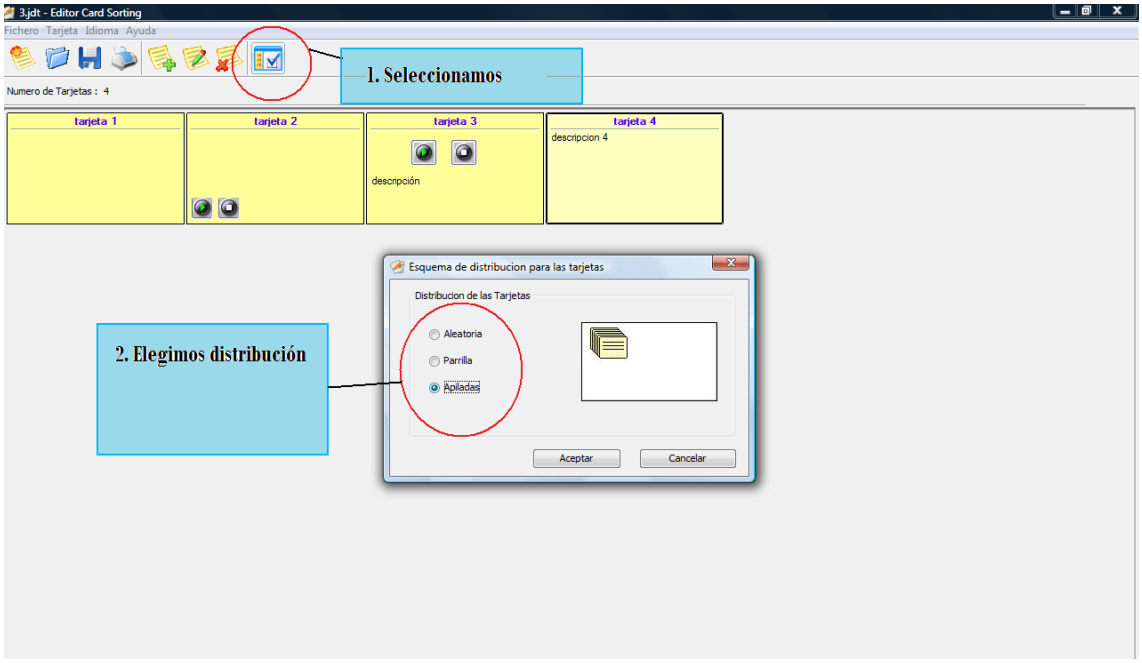
Eliminar tarjetas

1. Seleccionamos la tarjeta que deseamos eliminar.



Sesión de agrupación de tarjetas

Esta parte sirve de explicación tanto de la parte de Diseño de Sesión de Card Sorting, como de la Ejecución de la Sesión de Card Sorting. En el primer caso deberíamos realizar el primer paso y en el segundo no sería necesario.



1. Indicamos nuestro nombre cuando sea solicitado.
2. Agrupamos las tarjetas según consideremos.
3. Seleccionamos el botón Siguiente