Proyecto Fin de Carrera

Ingeniería Informática

Junio de 2011

AraWord:
Un procesador de textos para comunicación aumentativa y adaptativa

---

Joaquín Pérez Marco

Director: Dr. Joaquín Ezpeleta Mateo

Departamento de Informática e Ingeniería de Sistemas

Universidad Zaragoza

Agradecimientos

A mi director Joaquín Ezpeleta, por confiar en mí para este proyecto, implicándose en éste desde el primer momento.

A José Manuel Marcos y César Canalís, profesores del CPEE Alborada, y David Romero de ARASAAC, por su amabilidad y ayuda a lo largo del proyecto.

Y por supuesto, a mis compañeros de clase, amigos, familia, y en general a todas aquellas personas que me han apoyado durante estos meses de trabajo.

AraWord: Un procesador de textos para comunicación aumentativa y adaptativa

Resumen

Existen personas que por diversas lesiones padecen graves trastornos en el habla lo que les limita a la hora de comunicarse con el entorno que les rodea. Para facilitarles la expresión sin utilizar la palabra hablada surgen los llamados sistemas aumentativos y alternativos de comunicación que complementan o sustituyen el lenguaje oral. Muchos de ellos recurren al uso de pictogramas que representen, de forma clara y sencilla, los conceptos más empleados en la comunicación cotidiana.

Este proyecto fin de carrera tiene por objetivo el desarrollo de un procesador de textos, para el ámbito de la comunicación aumentativa y alternativa, que permita la escritura simultánea con texto y pictogramas. El procesador incluye algunas de las funcionalidades más habituales de este tipo de herramientas (creación, apertura y guardado de documentos, posibilidad de exportar a PDF y archivos de imagen; cortar, copiar y pegar textos, búsqueda de palabras en el documento, ayuda al manejo de la aplicación, etc), así como las específicas para el trabajo con el conjunto de pictogramas del Portal Aragonés de la Comunicación Aumentativa y Alternativa ARASAAC.

La interfaz del programa, así como la base de datos con la que trabaja, está traducida a ocho idiomas diferentes (castellano, inglés, francés, alemán, italiano, catalán, portugués, portugués brasileño). Es fácilmente ampliable según las necesidades que surjan en distintos centros educativos. Se distribuye como *software* libre bajo licencia GPL, y está programado en Java, lo que permite su ejecución en distintos sistemas operativos y plataformas (Windows, Linux, Mac...).

La aplicación ha sido probada por profesores del CPEE Alborada de Zaragoza, lo que permite garantizar que se ajusta a las necesidades de éstos desde el primer momento. Próximamente se empezará a utilizar en diversos colegios y asociaciones dedicadas a personas con necesidades especiales, y en la docencia de asignaturas universitarias relacionadas con la educación especial. Además, se ha empezado a utilizar en la radio pública aragonesa (Aragón Radio 2) para generar los titulares de las noticias adaptadas a estas personas con necesidades especiales.

Índice general

Capítulo 1

Introducción

En el primer capítulo de esta memoria se va a introducir el Proyecto Fin de Carrera (PFC), exponiendo en primer lugar la motivación para llevarlo a cabo, así como los objetivos que se buscaban con su realización. Se explicará además el trabajo previo en el que se basa y por último el contenido y alcance de este documento.

1.1. Motivación

A la hora de buscar un proyecto fin de carrera lo que pedía era poder aprovechar los conocimientos adquiridos en algo útil para otras personas. No quería que quedase exclusivamente en el ámbito universitario o investigador y por eso al ver la propuesta de este proyecto, un trabajo que tenía una repercusión social directa, lo solicité inmediatamente. Se trataba de desarrollar un nuevo *software* para su uso en un colegio público de educación especial, cuyos alumnos son jóvenes que presentan distintas discapacidades físicas o psíquicas, y este *software* les facilita herramientas para aumentar su capacidad de comunicación con su entorno.

Por otra parte no se trataba de ampliar un proyecto ya existente (como pudiera ser TICO[1]) sino de realizar un programa nuevo, lo cual puede ser complejo, pero que desde mi punto de vista considero mucho más interesante. Permite ser extremadamente creativo a la hora de desarrollar soluciones para los problemas que se van presentando, y no estás muy ligado a decisiones (de análisis, diseño, implementación u otras) tomadas por otras personas, con las que podrías no estar de acuerdo.

1.2. Trabajo previo relacionado

Desde hace varios años, miembros del Departamento de Informática e Ingeniería de Sistemas[2] (DIIS) vienen colaborando con el grupo de investigación Tecnodiscap[3]. En esta colaboración se desarrollan materiales *hardware* y *software* que sirvan de apoyo en las tareas

[1] `http://www.proyectotico.es`
[2] `http://diis.unizar.es`
[3] `http://tecnodiscap.unizar.es`

formativas que el Colegio Público de Educación Especial Alborada[4] (CPEE Alborada) lleva a cabo con personas que tienen diversas discapacidades físicas y/o psíquicas.

Este centro ya dispone de un programa similar al realizado en este proyecto, llamado Escribir con Símbolos[5]. Pese a ser muy completo, tiene una serie de deficiencias.

1. Es de pago (cuesta 80$ por licencia, para un único ordenador), lo cual imposibilita su uso en centros con escasos recursos económicos, o por particulares en similar situación.
2. Es un programa demasiado complejo, muy poco adaptado al uso que se le da en el colegio, y ello provoca que su curva de aprendizaje sea excesiva para la función que debe cumplir.
3. Es poco flexible a la hora de integrar los pictogramas de ARASAAC, que es el estándar *de facto* en Aragón y otras muchas Comunidades Autónomas e incluso países.
4. Los usuarios especializados de Aragón están acostumbrados a manejar los pictogramas junto con su categorización para búsquedas, inserción en documentos/proyectos TICO, etc. Integrar todo este conocimiento en Escribir Con Símbolos es una tarea muy costosa (se dispone de más de 6000 pictogramas y 40000 términos). Además, están en varios idiomas y poseen información adicional sobre el tipo de palabra que es (nombre, adjetivo, verbo, etc.) que no podríamos aprovechar.
5. Por último, Escribir Con Símbolos no permite el uso de formas verbales conjugadas (salvo pasado simple) lo cual es una característica muy demandada por los usuarios. AraWord sí permite el uso de formas verbales simples y compuestas en castellano, en todos los tiempos y modos.

El presente PFC parte del concepto de dicho software, aunque adecuándolo a las necesidades reales del centro. Se ha desarrollado a tiempo parcial durante los meses de septiembre de 2010 a junio de 2011 en el Departamento de Informática e Ingeniería de Sistemas del Centro Politécnico Superior de la Universidad de Zaragoza.

1.3. Objetivos del proyecto

Este proyecto fin de carrera tiene por objetivo el desarrollo de un procesador de textos en el ámbito de la comunicación aumentativa y alternativa, que permita la escritura de texto y sus pictogramas asociados. Entre otras, las funcionalidades que debe tener son las siguientes:

1. Gratuidad absoluta tanto del software como de los pictogramas a utilizar.
2. Creación, apertura y guardado de documentos.
3. Posibilidad de exportar a PDF y archivos de imagen.
4. Cortar, copiar y pegar textos en el propio programa o desde/hacia aplicaciones externas (navegadores, suites ofimáticas, etc).

[4] `http://centros6.pntic.mec.es/cpee.alborada`
[5] `http://www.widgit.com/international/spanish/rebus-only.htm`

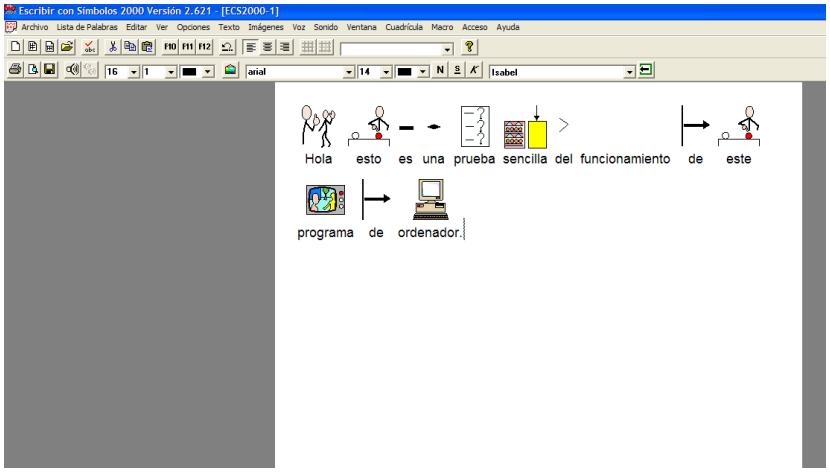

Figura 1.1: Ejemplo de documento de Escribir con Símbolos 2000

5. Aquellas específicas para el trabajo con el conjunto de pictogramas del Portal Aragonés de la Comunicación Aumentativa y Alternativa ARASAAC (marcos de palabras coloreados, uso de la BBDD de imágenes de dicho portal, etc.).

6. Flexibilidad para integrar dichos pictogramas, así como insertar nuevos según las necesidades del usuario. Uso de la información extendida de tipo de concepto: nombre, adjetivo, etc. así como manejo automático de formas verbales conjugadas, palabras compuestas...

1.4. Contenido y alcance del documento

Este documento se estructura en dos partes: una memoria que resume el trabajo realizado y algunos anexos con documentación técnica más detallada del mismo.

La memoria está dividida en una serie de capítulos siendo el primero de ellos la presente introducción. El capítulo 2 explica una serie de conceptos del ámbito de la educación especial necesarios para comprender el resto de la memoria. El capítulo 3 está dedicado a describir el proceso de desarrollo de los objetivos del proyecto. En el último capítulo se recogen las conclusiones del PFC desde el punto de vista del cumplimiento de los objetivos y los diversos problemas encontrados. También se detalla el posible trabajo que surge como continuación de este proyecto y se incluye una valoración personal del desarrollo de este PFC.

Acompañando a la memoria se adjuntan algunos anexos que explican con más detalle los resultados del proyecto y el camino seguido para obtenerlos:

1. Anexo **??**: Recoge el desarrollo detallado del *software*.

2. Anexo **??**: Recoge la distribución temporal del proyecto.

3. Anexo **??**: Recoge el manual de usuario de la aplicación.

4. Anexo **??**: Recoge la licencia bajo la que se distribuye el *software* realizado.

Capítulo 2

Conceptos previos

En este capítulo se explican brevemente algunos conceptos que se utilizan en el ámbito de la comunicación aumentativa y adaptativa, así como sus posibles usos. Estos conceptos son la base para comprender en su totalidad el entorno y objetivo de este proyecto.

2.1. Educación especial

Educación especial es aquella destinada a alumnos con necesidades educativas especiales debidas a superdotación intelectual o discapacidades psíquicas, físicas o sensoriales. La educación especial en sentido amplio comprende todas aquellas actuaciones encaminadas a compensar dichas necesidades, ya sea en centros ordinarios o específicos.

Aunque la atención educativa a deficientes sensoriales (generalmente auditivos y visuales) se viene prestando en España desde el siglo XVI, la adopción legal del término educación especial es reciente, viniendo a sustituir a otros aún vigentes en ciertos países de Hispanoamérica como defectología que tienen evidentes connotaciones negativas.

En los últimos años del siglo XX se ha propuesto en España y otros países la sustitución del término educación especial por el de necesidades educativas especiales siguiendo las recomendaciones del informe Warnock, publicado en 1978 y difundido a lo largo de la década siguiente. Esta nueva definición supone hacer énfasis en la concepción de la educación básica como un servicio que se presta a la ciudadanía para que alcance sus máximas potencialidades y por tanto en la obligación del sistema de proporcionar apoyos y medios técnicos y humanos para compensar los déficits del alumnado en el acceso a los aprendizajes básicos imprescindibles para afrontar la vida adulta.

2.2. Pictograma

Un pictograma es un signo que representa esquemáticamente un símbolo, objeto real o figura. En la actualidad es entendido como un signo claro y esquemático que sintetiza un mensaje sobrepasando la barrera del lenguaje; con el objetivo de informar y/o señalizar. Incorpora información semántica sobre el tipo de palabra, para saber si es un nombre común, un nombre propio, descriptivo, verbo, etc.

Figura 2.1: Ejemplo de pictograma con el concepto Abeja

En AraWord, para cada uno de estos pictogramas, guardamos una gran cantidad de información semántica al respecto. Por ejemplo, para el pictograma de arriba (palabra abeja) está traducida a ocho idiomas (abeja, bee, abeille, etc.) y sabemos que es un nombre común, lo cual es útil para categorizar por colores del borde del pictograma en la aplicación.

2.3. Sistema de comunicación

La comunicación constituye una de las formas en que las personas interactúan entre sí y es la base del aprendizaje y la adquisición de conocimientos. Existen muchas formas de comunicación: gestual, a través de signos, mediante imágenes, verbal, escrita, etc. A veces, en determinadas personas, pueden aparecer problemas en la comunicación debido, entre otras causas, a trastornos auditivos, retrasos en su desarrollo o lesiones cerebrales, como puede ser la parálisis cerebral infantil (PCI) en el caso de los niños. Esto puede provocar graves dificultades en el habla, lo que les impide expresar sus necesidades, deseos o pensamientos de una forma adecuada. Como apoyo para facilitar la comunicación surgen los sistemas alternativos/aumentativos de comunicación que son formas de lenguaje diferentes del habla, que se utilizan cuando ésta se encuentra seriamente afectada.

La clasificación más general de los sistemas de comunicación distingue entre sistemas con apoyo externo y sistemas sin apoyo externo, dependiendo de la necesidad o no de elementos ajenos a la propia persona que trata de comunicarse. Entre los sistemas de comunicación sin apoyo externo se encuentra la dactilología (lengua de signos).

Por otra parte, los sistemas de comunicación con apoyo se orientan a mejorar la producción del habla, y recurren a la ayuda de sistemas ortográficos, pictográficos e informáticos, que suplan en todo o en parte las deficiencias expresivo-articulatorias de la persona que quiere comunicarse.

La principal aplicación de estos sistemas es conseguir el intercambio de información entre las personas que tengan alguna dificultad en el habla y las personas que estén a su cargo: padres, profesores, tutores o monitores. De esta manera pueden expresar sus necesidades básicas, deseos, sentimientos, etc.

Tomando la comunicación como base son también muy útiles en el ámbito de la enseñanza.

Pueden servir para favorecer el aprendizaje de la lengua escrita estableciendo la relación entre los sonidos del habla y las letras o para reconocer los ámbitos que les rodean: escolar, familiar y personal. También son muy útiles a la hora de manifestar sus conocimientos, planteando preguntas directas a las que deben responder mediante el uso de sí o no.

2.4. Sistemas alternativos y aumentativos de comunicación (SAACs)

Son instrumentos de intervención logopédica/educativa destinados a personas con alteraciones diversas de la comunicación y/o el lenguaje, y cuyo objetivo es la enseñanza mediante procedimientos específicos de instrucción, de un conjunto estructurado de códigos no vocales que permiten funciones de representación y sirven para llevar a cabo actos de comunicación (funcional, espontánea y generalizable) por sí solos o en conjunción con otros códigos, vocales o no vocales.

Capítulo 3

Proceso de desarrollo

Este capítulo está dedicado a explicar el trabajo previo al desarrollo de la aplicación y el proceso seguido durante su desarrollo. Se expone el desarrollo de cada una de las partes que han compuesto el proyecto por separado siguiendo el orden en que se han realizado, haciendo especial hincapié en las decisiones tomadas y explicando los motivos que llevaron a cada elección. En este capítulo no se detallan las pruebas realizadas al *software* desarrollado quedando incluidas en el apéndice **??**.

3.1. Trabajo previo

Antes de comenzar a estudiar los distintos objetivos de este PFC fue necesario dedicar un tiempo a familiarizarse con la aplicación anterior, Escribir Con Símbolos (el programa en el que se basa parcialmente AraWord) y TICO (software libre, Java, desarrollado también en el CPS bajo la dirección de la misma persona que AraWord, etc). En primer lugar se debía comprender la utilidad que prestaba en el ámbito de la educación especial. Para esto fue esencial la lectura sobre los sistemas aumentativos y alternativos de comunicación.

Tras ello, una visita al CPEE Alborada permitió conocer a los profesores encargados del proyecto TICO y este nuevo proyecto, así como comprobar de primera mano la problemática específica del centro, lo cual sirvió para crear un marco general de funcionalidad del software a desarrollar.

3.2. Definición de requisitos

La primera fase del análisis de la aplicación fue definir los requisitos que debía cumplir. Éstos se definieron en varias reuniones con los profesores del CPEE Alborada y están recogidos en el apéndice **??**.

3.3. Preparación del entorno de trabajo

Tras ello, hubo que preparar el entorno de desarrollo. Se acordó que fuera Eclipse[1], por homogeneidad con el resto de PFCs de la misma temática. No obstante, por mi parte prefería el editor gráfico de NetBeans, y finalmente se utilizó fundamentalmente Eclipse, empleando NetBeans[2] únicamente para el desarrollo de la interfaz de usuario (menús, ventanas...). Usando algún truco sencillo, es posible su coexistencia pacífica para el desarrollo de software y de este modo, la GUI[3] es más sencilla de realizar y sobre todo de modificar según surjan nuevas necesidades.

Adicionalmente, se configuró un sistema de control de versiones (Subversion) gratuito en SourceForge[4] (la mayor comunidad de desarrollo de software libre del mundo), accesible a través de Eclipse mediante un plugin (Subclipse). De este modo, era sencillo desarrollar pequeñas mejoras al programa, guardando los cambios en el servidor de Subversion cuando eran satisfactorios, de modo que sirviese a la vez como medio para compartir código (con mi director de proyecto y en su caso, con otras personas que pudieran estar interesadas en él), copia de seguridad, e histórico del código fuente. Por supuesto, en casos extremos sirve para volver rápidamente a una versión del día anterior (o de semanas atrás) estable y que sabemos que funciona correctamente. En general, es una herramienta muy útil y casi indispensable, sobre todo en proyectos más grandes o con más personas involucradas.

Para seguimiento del proyecto (tiempos, resultados conseguidos, etc) a mayor nivel, se optó por seguir una tecnología de desarrollo ágil (*agile software*), finalmente decantándonos por Scrum por su sencillez. En vez de utilizar una aplicación convencional para manejar Scrum, utilizamos una página web: PangoScrum[5] que nos permitía tener todos los datos accesibles desde cualquier lugar. Mediante esta herramienta, se puede conocer en todo momento el estado del proyecto, las tareas pendientes o con atraso sobre el plan previsto, etc. Además, se ajusta totalmente a los requisitos cambiantes por parte del CPEE Alborada a lo largo del proyecto, así como a los pequeños ciclos de análisis de problema, diseño de solución, implementación y pruebas, en vez de un proceso clásico de Ingeniería del *Software*, más apropiado para proyectos con objetivos estables de principio a fin. Se discute más detalladamente esta metodología y los buenos resultados obtenidos con ella en el Apéndice A.

Como curiosidad, el nombre original del programa iba a ser PictoWord, pero vimos que ya existían programas llamados así (o con nombre muy parecido) y tras algún cruce de correos electrónicos, decidimos cambiar el nombre por AraWord (unión de ARASAAC y Microsoft Word, el procesador de textos conocido por todos).

3.4. Primeros bocetos

Tras un simplificado análisis del problema, se pasó a las primeras pruebas rápidas con código. Se intentó realizar un boceto extremadamente básico con cierta rapidez, para poder enseñárselo a los profesores encargados del colegio y que vieran que el proyecto avanzaba

[1] `http://www.eclipse.org`
[2] `http://netbeans.org`
[3] *Graphical User Interface*, interfaz gráfica de usuario
[4] `http://www.sourceforge.net`
[5] `http://www.pangoscrum.com`

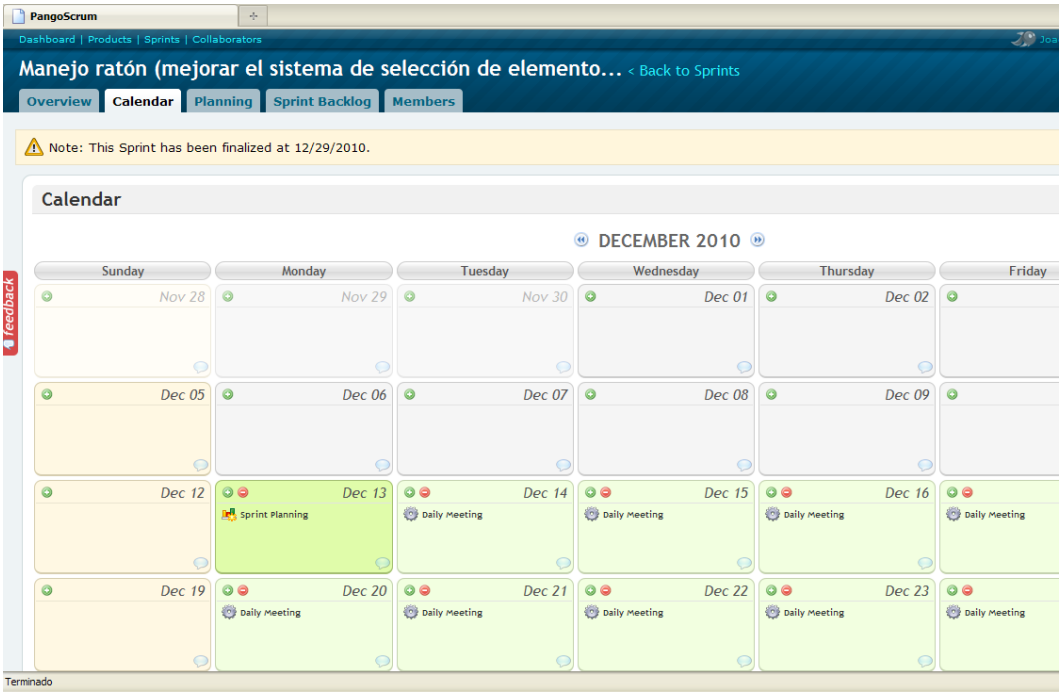

Figura 3.1: Web de PangoScrum

adecuadamente, para luego ir añadiéndole cada vez más funcionalidades y refinando el código (una aproximación al desarrollo de software bastante frecuente) hasta quedar satisfechos con el resultado.

La primera solución fue considerar una línea de texto entera en un campo de texto, pero el alineamiento de las imágenes con las palabras era muy complicado y se generaban otros problemas derivados, por lo que sustituimos esa idea con la de que un campo de texto es una palabra. Así, llegamos a la estructura fundamental que ha permanecido constante hasta el final del proyecto: los elementos (unidad mínima de tratamiento) del programa son las palabras.

Casi todo el espacio de la ventana principal del programa está ocupado por un JTextPane en el que insertamos componentes (paneles) separados por uno o más separadores (espacios en blanco, tabuladores, saltos de línea). Cada uno de estos componentes almacena una sola palabra, estando compuestos por un JPanel con dos elementos: una imagen en la parte superior (una JLabel con icono) y un campo de texto en la parte inferior (un JTextField). Tanto este JPanel como el JTextField tienen su funcionalidad por defecto muy ampliada (mecanismo de extensión de clases de Java), incluyendo una serie de atributos para cada uno de ellos, y manejadores de eventos interesantes (teclado, ratón, y foco).

Tras ello, había que pensar cómo resolver el problema de encontrar la imagen apropiada a una palabra concreta, y se decidió emplear una base de datos SQL sencilla. Con el driver JDBC y SQLite 3.x, fue bastante simple volcar los datos desde un archivo de texto generado por ARASAAC y tener un primer boceto sencillo que mostraba palabras y sus imágenes asociadas, cada una en un panel, todas ellas insertadas adecuadamente en la zona de escritura principal.

3.5. Introducción de funcionalidad básica

Después de obtener el visto bueno y algunas sugerencias de los profesores del colegio, empezó la fase de creación de funciones básicas. Primeramente, se creó una barra de menú sencilla, con muy pocas opciones (crear archivo nuevo, abrir y guardar ficheros en texto plano, importar la base de datos a partir del archivo de texto, mostrar ayuda del programa). Todas esas opciones a su nivel más básico. También se incluyó una función de exportar a PDF (con una librería de código abierto, gratuita, iTextPDF), de funcionalidad muy limitada, cuya mejora se dejó para más adelante.

Lo siguiente a realizar fue algo elemental: poder ir tecleando nuestro texto, creándose los elementos necesarios dinámicamente y cargando las imágenes adecuadas para cada palabra. Aquí incluimos un event listener de teclado. Fundamentalmente, filtramos caracteres no deseados y vamos escribiendo caracteres. Las teclas especiales, como el espacio, el tabulador o el salto de línea, marcan el final de una palabra, la creación de una nueva y demás ajustes. El backspace y la tecla Supr podrían provocar el borrado de un separador, juntando dos palabras en una, con sus acciones asociadas... Es posiblemente una de las partes más duras del proyecto, debido a la gran cantidad de casos particulares (primer elemento, último, partir a mitad de una palabra, etc.) y requirió una considerable cantidad de depuración y pruebas, sobre todo en esta fase (y detalles puntuales, a lo largo del resto de proyecto).

Una vez lograda una versión primitiva de procesador de textos ya relativamente funcional, pasamos a un requisito pedido desde el colegio Alborada y que se intuía problemático: las palabras compuestas. En efecto, *cepillo de dientes* son tres palabras, pero corresponden a un único concepto y por tanto, a una única imagen. Tras analizar detenidamente el problema, la solución pasó por comprobar tras cada pulsación de tecla (editando el texto del documento) si sería posible formar una palabra compuesta con las n, n-1, n-2, ... 1 palabras anteriores y posteriores a la actualmente editada.

Esto requiere comprobar bastantes cosas. Que no estamos demasiado cerca del principio o final del texto, en cuyo caso hay algunas combinaciones que no son posibles, que hay una adecuada alternancia de elementos y separadores y que estos son del tipo correcto (por decisión de diseño, solamente combinaciones tipo ([elemento] [espacio en blanco])* [elementoActivo] ([espacio en blanco] [elemento])* son válidas), y que esa palabra compuesta existe en nuestra base de datos (en la mayor parte de ocasiones, no será así, por ejemplo *muy contento de* no corresponde a ninguna palabra compuesta razonable).

Veamos un ejemplo. Con n=3 (la llamada *ventana de comprobación de palabras compuestas*), si tenemos la siguiente frase: *Yo como carne y fruta*, borramos *carne* e insertamos en su lugar *patatas fritas* (palabra compuesta que tiene un pictograma asociado) el programa comprueba, por este orden: *como patatas fritas, patatas fritas y, fritas y fruta, patatas fritas.* Aquí deja de comprobar, pues encuentra un resultado satisfactorio. Como se puede apreciar, siempre se intenta encontrar la palabra compuesta más larga posible (hipotéticamente, podrían existir varias posibilidades de palabra compuesta a la vez, aunque de momento no se ha dado tal caso).

Como esta tarea es relativamente compleja y sobre todo, hay que ejecutarla muy a menudo, la longitud máxima de las palabras compuestas que comprobaremos tiene que ser baja (hasta 3 elementos, no da problemas de rendimiento en cualquier computador moderno). Si lo subimos, el programa funciona igual pero se notan más los saltos y pequeño retardo al redibujar. En

cualquier caso, se incluyó una función de búsqueda manual de palabras compuestas, para las escasas situaciones en las que una palabra compuesta tiene 4, 5 o más palabras en su interior, y que se puedan juntar de igual modo.

Tras ello, hubo que rehacer el método de partir palabras con espacio en blanco, Supr, backspace y demás, llegando a soluciones de compromiso en ciertos casos por rendimiento. En general, se ha intentado agilizar el caso más frecuente, dejando siempre la posibilidad manual de buscar palabras compuestas en caso de que el programa no las convierta automáticamente bien (pasa en casos muy concretos). Además, en este momento también se llevó a cabo una fuerte reestructuración, limpieza y depurado del código del programa, dividiendo en distintos módulos y ficheros lo que antes estaba en uno o dos solamente.

Ahora llega el turno de una función muy básica también en un procesador de textos: cortar, copiar y pegar. Para ello hay que incluir un event listener de ratón. Se realiza una primera versión sencilla de selección de elementos, con variables booleanas de estado de la selección y una zona para elementos *en portapapeles*, por hacer un símil con Word. Pero su depurado es difícil, y no contempla todos los casos posibles, con lo que se decide pasar el control a una máquina de estados adecuada, representada a continuación:

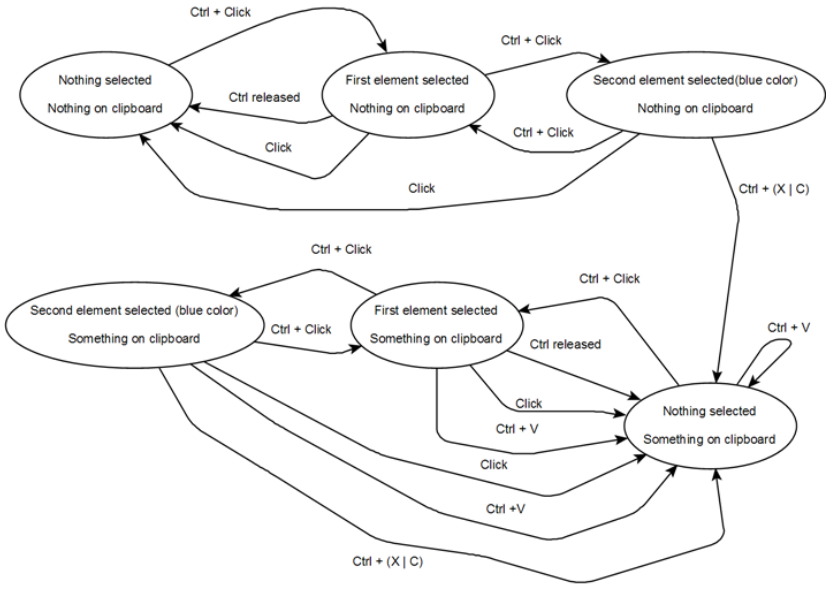

Figura 3.2: Máquina de estados para Cortar, Copiar y Pegar (versión 1)

Como el sistema Ctrl+clic no termina de convencer, se intenta realizar el de *arrastrado de componentes*, con lo que hay que añadir otro *event listener* para la acción de arrastre del ratón. Es evidente que la máquina de estados se puede simplificar bastante, y se deja así:

Se realizan algunas mejoras menores (eliminar la selección texto interna de los JTextField que impide al sistema de seleccionar elementos funcionar correctamente al hacer drag sobre un JTextField; posibilidad de suprimir directamente los elementos seleccionados con la tecla Supr; imposibilidad de cortar TODOS los elementos del texto (perderíamos un lugar donde pegarlos: pequeña limitación del programa); inclusión de un nuevo menú Edición con Cortar, Copiar, Pegar y otros comandos explicados a continuación).

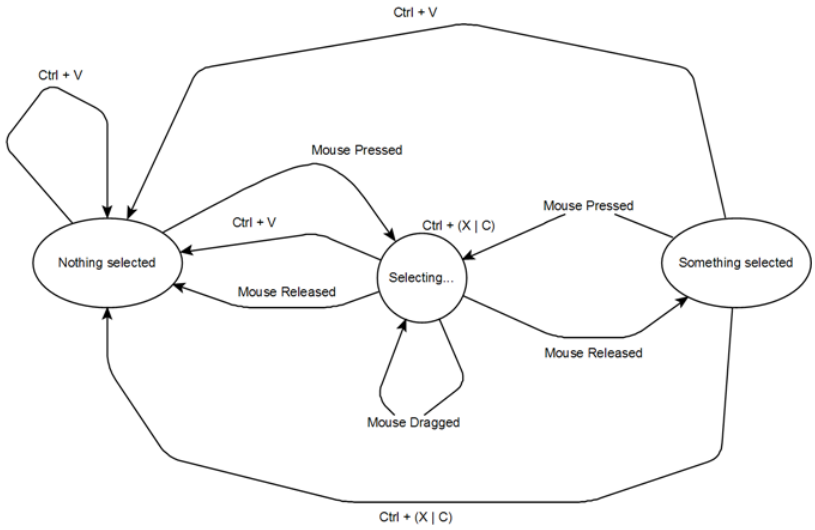

Figura 3.3: Máquina de estados para Cortar, Copiar y Pegar (versión 2)

Siguiendo con funciones del menú de Edición, le toca el turno a la búsqueda de texto, la cual se implementó sin mucha dificultad (salvo el tema de centrar el foco en el elemento apropiado). Memoriza la última posición en la que has buscado, para poder implementar la función *buscar siguiente* de cualquier programa de tratamiento de textos (si no fuera así, sólo podrías buscar la primera ocurrencia de una subcadena en el texto completo). No localiza varias ocurrencias de una subcadena en la misma palabra (mejora innecesaria dado el contexto de la aplicación). Por la misma razón, no se implementó la función Reemplazar, dado que se consideró innecesario en el entorno de uso del programa. No obstante, es muy sencilla de realizar en caso de verla útil.

Deshacer la última acción realizada requirió un poco más de tiempo. Tras varios intentos y pruebas, se llegó a un diseño adecuado, en el cual tenemos varias listas de elementos: la principal, que por comodidad se mantuvo con el mismo nombre, y un *conjunto deshacer* de listas de elementos, con fotografías del estado (y elementos) del programa en la modificación realizada 1, 2, ..., N pasos antes. Por defecto N (ventana de deshacer) es 5, parámetro ajustable por el usuario (como siempre, incrementarlo aumenta tamaño de la ventana de deshacer, memoria necesaria y tiempo de CPU invertido en manejo de la funcionalidad, y viceversa). Al realizar alguna modificación de interés en los datos del documento, se copia antes un estado de los datos (copia de la lista de elementos a la primera posición de las listas viejas) y se realiza la modificación, borrando la última de la lista en caso de que superemos el tamaño de la ventana de deshacer (caso habitual). Al dar la orden de deshacer, simplemente pasamos a la lista anterior a la actual (1, 2, 3...) guardando la modificación realizada en el conjunto de listas rehacer.

Se llevó a cabo una nueva reestructuración del código fuente en módulos manejables y con funciones relacionadas entre sí, así como una incorporación al menú Edición de la función buscar y deshacer, y algunas mejoras menores del programa en general. La función Rehacer se añadió sin ninguna dificultad, tan sólo teniendo en cuenta que cada cierto tiempo el programa se encargue de efectuar limpieza del conjunto de listas rehacer, ya que durante el funcionamiento normal de la aplicación no se van limpiando automáticamente.

Se realizó un cuadro de preferencias generales del programa, así como la funcionalidad de ocultar o mostrar la imagen asociada a un determinado pictograma. Posteriormente, se incorporaron las características multiidioma de la interfaz (español e inglés en un principio) mediante el uso de ficheros de lenguaje (situados en /lang) y algunos trozos de código de TICO (para manejo de ResourceBundle). Las preferencias del programa se guardan en un fichero XML situado en /conf, y se cargan y guardan mediante algunas funciones recicladas también de TICO.

3.6. Mejora del rendimiento general, y extensión de funcionalidades

Llegados a este punto, el programa empieza a ser perfectamente utilizable en un entorno real. No obstante, al ejecutarlo en un PC menos potente que aquel en el que se desarrolló hasta ahora (un Quad Core a 2.4 GHz con 4GB de RAM) la velocidad de respuesta era excesivamente lenta, hasta el punto de ser molesto para el usuario. Debido a que esto es totalmente inaceptable, hubo que revisar qué podía provocar tal falta de rendimiento.

Fue fácil encontrar una de las causas del problema: el sistema de Deshacer y Rehacer. Copiar todos los elementos del texto tras cada modificación es extremadamente lento, ya que los objetos deben serializarse, incluso dejando aparte consideraciones de uso excesivo de memoria del sistema. Para solucionarlo se decidió rediseñar completamente el código de dichas funciones, guardando el texto del documento como String, clase elemental con un rendimiento muy superior. Ello incrementó el rendimiento notablemente, con una ventaja añadida no prevista, y es que el código es mucho más sencillo, legible, mantenible y extensible, amén de menos proclive a errores.

Este cambio funcionó tan bien que se decidió extender a las funciones de Cortar, Copiar y Pegar, con excelentes resultados (aunque menos notables debido a la menor frecuencia de la operación). Además, permitió implementar otro requisito de la aplicación, y es poder copiar y pegar texto directamente desde o hacia otras aplicaciones (navegadores web, otros procesadores de textos, etcétera).

Faltaba incluir una característica muy demandada por el CPEE Alborada, y es la utilización de información extendida de las palabras de la base de datos acerca de su tipo sintáctico (nombre común, nombre propio, verbo, adjetivo...) con distintos códigos de color (estandarizados por el Sistema Pictográfico de Comunicación, SPC) colocados a modo de marco sobre cada palabra del texto. Debido al uso de un JPanel para almacenar cada palabra y el hecho de que tengan un marco (border) asociado totalmente personalizable, fue muy sencillo de incorporar, requiriendo tan sólo un sencillo cambio en la estructura de la BD interna.

Se adecentó un poco la interfaz de usuario, agrupando funciones similares en submenús y eliminando algunas características obsoletas. También se adoptó el formato XML para los documentos AraWord, de forma que se pueden guardar características específicas del documento (idioma, tipo y tamaño de la fuente, tamaño imágenes, etc.) Esto es necesario porque un mismo educador puede tener a su cargo a alumnos con necesidades visuales muy diferentes, e incluso en diferentes idiomas, y facilita enormemente su tarea.

Por último, en este momento se inició la elaboración formal de la memoria (para organizar

adecuadamente lo que hasta ahora eran conjuntos de anotaciones sueltas), tarea que continuó en paralelo con el desarrollo del código hasta el final del proyecto.

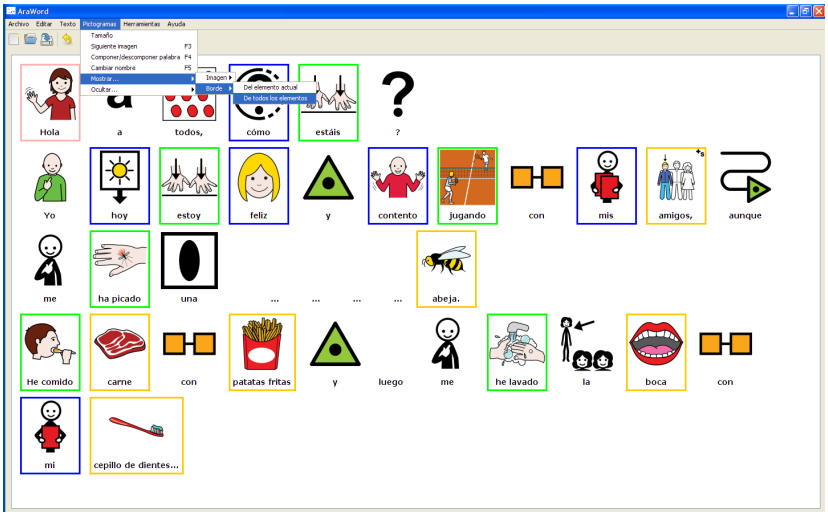

Figura 3.4: Una muestra de Araword ejecutándose

3.7. Integración con el resto de aplicaciones de ARASAAC: Tico y AraPicto

Para poder utilizar una base de datos y una carpeta de pictogramas común para las tres aplicaciones, se tuvo que rediseñar la parte de base de datos de la aplicación, para incluir las funciones desarrolladas por Adrián Gómez (del PFC AraPicto), al que desde aquí quiero agradecer su ayuda y paciencia conmigo. El problema es que es una base de datos relativamente compleja, y ralentizaba la búsqueda de las imágenes (que en AraWord es un asunto crítico para un manejo de usuario confortable, y en AraPicto no es tan relevante). Tras probar varias soluciones, hallamos la adecuada: una vista específica para AraWord de las tablas reales de la base de datos. De este modo, manteniendo las tablas que se usan en el resto de aplicaciones, podemos mantener unos tiempos de acceso razonables a imágenes de la base de datos.

3.8. Reconocimiento de formas verbales conjugadas en castellano

Hasta ahora, sólo era posible reconocer los verbos en infinitivo, con lo que para escribir por ejemplo *yo como patatas* y mantener los iconos, hay que escribir *yo comer patatas*. Este es un problema clásico en este tipo de software, y se pudo resolver gracias a la obtención de un listado de verbos en castellano con todas sus formas verbales asociadas, tanto simples como compuestas. Una vez logrado eso, crear una base de datos que contuviese listas de formas verbales con su verbo en infinitivo asociado e incorporar dicha información a la aplicación fue trivial.

3.9. Primera versión estable. Reunión y posteriores modificaciones

Tras una reunión con los responsables de ARASAAC y el CPEE Alborada, se elaboró una lista de detalles a mejorar o personalizar, entre ellos, detección automática del último pictograma usado para una palabra dada, posibilidad de asignar texto libre a un pictograma (por ejemplo, sustituir la palabra *niño* por *Antonio* sin perder la imagen asociada), mejorar la calidad de las imágenes exportadas, posibilidad de elegir entre diferentes listas de palabras (conjunto de pictogramas de ARASAAC, o de lengua de signos española, etc), cambio del color del texto, así como poder ponerlo en la parte superior o inferior del pictograma. Se corrigieron también problemas con símbolos de puntuación. También se reorganizaron las funciones de los menús para que fuesen más intuitivas para usuarios acostumbrados a manejar otros procesadores de texto convencional como Microsoft Word.

3.10. Cambio en la base de datos. Gestor de recursos común

La base de datos común que se venía utilizando hasta ahora, aunque funcional, presentaba un tiempo de importación de imágenes bastante alto debido a su complejidad interna. Tras una reunión decidimos que buena parte de su complejidad era innecesaria y se pasó a una versión simplificada, con dos tablas auxiliares de idioma y tipo de palabra, y una tabla principal con palabra, código de lenguaje, código de tipo, nombre del fichero y un campo más con el nombre del fichero no normalizado, que se mantuvo por retrocompatibilidad con TICO, aunque podría desaparecer en un futuro próximo. De este modo, el tiempo de importación de imágenes se redujo muy apreciablemente (de cuatro minutos a menos de uno, para idénticos parámetros como tamaño de base de datos y velocidad de la máquina) y el tiempo de respuesta al usuario durante el manejo normal de la aplicación, también en menor medida.

Para gestionar las operaciones de importación, exportación, agregación y eliminación de imágenes, se creó una pequeña aplicación aparte (llamada ResourceManager) con las opciones mínimas y necesarias. Es bastante funcional, aunque queda pendiente para futuros PFCs ampliarla, dotarle de una interfaz mejor, mejorar la eficiencia de la operación de exportar base de datos, etc.

3.11. Últimos retoques

Se corrigieron diversos errores y pequeños fallos detectados durante pruebas, ninguno realmente grave, pero sí en mayor o menor medida molestos. También se realizaron las traducciones de la interfaz al resto de idiomas (hasta ahora, sólo estaba en español e inglés) gracias a la red de colaboradores de ARASAAC.

Capítulo 4

Conclusiones y trabajo futuro

En este último capítulo de la memoria se hace un balance de los resultados obtenidos y se recoge una reflexión sobre el cumplimiento de los objetivos inicialmente planteados para este PFC. Se detallan los principales problemas encontrados durante su desarrollo y se describen las líneas de posible trabajo futuro sobre AraWord. Por último se hace una valoración personal del trabajo realizado.

4.1. Resultados obtenidos

Con la finalización de este PFC se ha obtenido una primera versión de AraWord lo suficientemente estable para ser distribuida con garantías de tener un buen funcionamiento. Los objetivos que se plantearon inicialmente se han cubierto para la mayoría de usuarios de la aplicación y han sido ampliados y refinados con el fin de dotarla de las funcionalidades necesarias que fueron surgiendo a lo largo del proceso de desarrollo, con algunos detalles por depurar que sirven como punto de partida para futuras ampliaciones del *s*oftware.

La aplicación, que ha estado en permanente fase de pruebas, actualmente se está utilizando con los alumnos y monitores del CPEE Alborada. Asimismo los profesionales del colegio la han puesto en conocimiento de profesores y educadores de otros centros educativos españoles y europeos que es posible que comiencen próximamente a utilizarla. Además, se ha empezado a utilizar en la radio pública aragonesa (Aragón Radio 2[1]) para generar los titulares de las noticias adaptadas a estas personas con necesidades especiales.

4.2. Problemas encontrados

En general, a lo largo del desarrollo de este PFC no ha habido demasiados problemas graves que hayan supuesto un riesgo real a la hora de cumplir plazos de proyecto, y los pocos que ha podido haber son de índole humano. No obstante, algunos aspectos técnicos sí que tuvieron una dificultad extra no prevista y merecen ser señalados aquí.

Uno de esas tareas que resultó ser más costosa de lo que parecía inicialmente fue probar la

[1] `http://www.aragonradio2.com/catedu/arasaac`

aplicación en distintos sistemas operativos para garantizar que funcionaba correctamente y de la misma forma en todos ellos. La mayoría de los problemas se encontraron en el sistema operativo Linux. La ejecución de AraWord en este sistema provocaba una serie de fallos (mayoritariamente gráficos) que finalmente fueron solventados.

Otra dificultad a señalar durante el desarrollo de este proyecto ha sido la atención permanente de las distintas sugerencias realizadas por parte de los profesores del CPEE Alborada. Esto ha favorecido enormemente la corrección de errores y al enriquecimiento de la aplicación con las nuevas ideas que se plantean con el uso del *software*. Sin embargo, en muchas ocasiones, la atención ágil y el estudio de estas ideas ha requerido un esfuerzo adicional para hacerlo compatible con el trabajo ya planificado. Como dato orientativo, en la fase final de desarrollo del software se intercambiaron más de 140 correos electrónicos con sugerencias, mejoras e indicaciones.

4.3. Líneas futuras

Puede decirse que la versión actual de AraWord cubre casi todas las funcionalidades básicas necesarias para la elaboración de textos con pictogramas asociados, asegurando un correcto funcionamiento. Sin embargo, como en la mayoría de las aplicaciones *software*, se presentan una buena cantidad de posibilidades de mejora. Algunas de ellas no se han podido realizar por falta de tiempo, la mayoría por exceder con creces el objetivo del proyecto siendo como es una versión inicial, y todas van encaminadas a dotar al programa de ciertas características adicionales. Este posible trabajo futuro que surge como continuación de este PFC queda resumido en los siguientes puntos:

- **Lectura del texto con voz sintetizada:** Una funcionalidad muy interesante que no fue implementada por falta de tiempo, muy sencilla de realizar (dado que existe una librería de lectura de textos con voz sintetizada, con licencia de *software* libre, desarrollada por el grupo de trabajo de Eduardo Lleida, profesor del DIEC en el CPS, y que ya ha sido utilizada con éxito en otros proyectos).
- **Portabilidad a dispositivos móviles:** Debido al auge de dispositivos móviles cada vez más potentes y asequibles (y de código abierto, compatibles con Java, como Android) podría estudiarse la posibilidad de crear una versión ligera de este procesador de textos para dispositivos móviles.
- **Detalles pendientes de la aplicación:** Falta mejorar el gestor de recursos, particularmente el tiempo de la operación de exportar la base de datos, así como pulir del todo la opción Exportar a PDF, ya que la librería que se encarga de ello es bastante compleja y difícil de usar y por ello, dicha funcionalidad no está todo lo desarrollada que debería.

4.4. Valoración personal

Respecto al trabajo realizado, me siento muy satisfecho con los resultados conseguidos. Los objetivos planteados inicialmente se han cubierto y actualmente la aplicación se está usando, que es la mejor garantía de haber hecho un trabajo útil. Esto no habría sido posible sin el

interés e implicación del director del proyecto para que éste saliera adelante y el constante apoyo de los profesores del CPEE Alborada, excelentes profesionales y mejores personas.

La posibilidad de realizar un proyecto para un cliente real, así como las reuniones mantenidas en el colegio a lo largo del mismo, han sido una experiencia muy enriquecedora. El hecho de trabajar con profesionales de otro ámbito me ha servido mucho ya que casi siempre aportan un punto de vista diferente o nuevas ideas sobre el mismo proyecto. Con las visitas al colegio he tenido la oportunidad de conocer el entorno real donde se utilizaría la aplicación y las distintas actividades que tenían en mente para realizar con AraWord. Todo esto ha supuesto una motivación para seguir trabajando en este proyecto. Además, el acercamiento a personas que tienen distintas discapacidades, me ha ayudado a entender las dificultades que pueden llegarse a encontrar en lo que para nosotros es una actividad cotidiana que realizamos sin esfuerzo.

En cuanto al desarrollo del proyecto he podido comprobar la importancia que tiene gestionar bien el tiempo y fijar unas metas realistas, sobre todo cuando los objetivos se van ampliando conforme se cubren etapas. He podido aplicar y reforzar los conocimientos adquiridos en la carrera y he tenido la oportunidad de profundizar en algunos de ellos (particularmente, programación en Java). Asimismo he aprendido el uso de nuevas herramientas que no conocía, como el manejo de LaTeXpara elaborar la documentación del proyecto.

También he aprendido mucho de la experiencia de mi director que me ha ido enseñando la manera de enfrentarme a los distintos problemas que iban surgiendo para buscar una solución. A él debo agradecer el tiempo invertido en explicaciones e intercambios de puntos de vista y en muchos momentos la confianza depositada en mí, que me ha llevado a realizar distintas ideas que han ido surgiendo a lo largo del proyecto y que lo han enriquecido.