UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA

EN INFORMÁTICA

PROYECTO FIN DE CARRERA

eAula: desarrollando un prototipo desde una perspectiva de usabilidad infantil

Rosa María Carretero López

Septiembre, 2007

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Sistemas Informáticos

PROYECTO FIN DE CARRERA

eAula: desarrollando un prototipo desde una perspectiva de usabilidad infantil

Autor: Rosa María Carretero López

Directores: Elena María Navarro Martínez

Francisco Montero Simarro

Septiembre, 2007

A mis padres

y a Javi.

AGRADECIMIENTOS

En primer lugar quiero manifestar mi agradecimiento a mis tutores Elena Navarro Martínez y Francisco Montero Simarro cuya colaboración y sugerencias fueron fundamentales en el desarrollo de este proyecto.

Además quiero dar las gracias a mis amigos y a todas aquellas personas que en mayor o menor medida me han dado su apoyo.

También quiero expresar mi agradecimiento a todos los compañeros de clase que he tenido a lo largo de mis años de carrera por el clima agradable que creamos entre todos.

Así mismo, a los profesores que me han impartido clase, por transmitirme sus conocimientos de la manera que mejor han sabido.

Mis últimos agradecimientos, pero no por ello menos importantes, son para mi familia, en especial para mis padres porque siempre han estado ahí cuando los he necesitado, y para Javi por haberme ayudado en todo lo que ha podido y darme su apoyo siempre que lo he necesitado.

RESUMEN

Cada vez más la sociedad en la que vivimos se mueve mediante el uso de la Informática y hasta en el proceso más pequeño se usa un ordenador, por eso es bastante lógico que se intente introducir a los niños cada vez antes en este mundo. Una manera de que lo hagan sin apenas darse cuenta es empezando a "sentarlos delante de un ordenador" desde edades muy tempranas y ofreciéndoles juegos que les diviertan y además les ayuden a aprender, ya que de este modo se están consiguiendo dos objetivos muy importantes: El primero de ellos es que el niño se familiarice con la tecnología desde una edad muy temprana. El segundo es que mientras lo hace además está realizando actividades propias de su edad y está adquiriendo conocimientos, ambas cosas sin apenas darse cuenta de ello.

Además, se debe tener en cuenta que el juego está considerado como una muy buena herramienta de aprendizaje, sobre todo en edades tan tempranas como las que se tratan en este proyecto, dado que mientras que el niño juega adquiere los conocimientos con más facilidad y asimila las cosas con mucho menos esfuerzo.

Para el desarrollo del presente proyecto se han considerado las recomendaciones del "currículo infantil" del MEC, pretendiendo así aunar las actividades que éste propone y desarrollarlas desde un punto de vista ingenieril. Para este fin se ha desarrollado una aplicación para la educación infantil llamada eAula donde se han tenido en cuenta lo indicado en el currículo infantil a la hora de definir el funcionamiento de las actividades y las técnicas propuestas por la ingeniería en el proceso de desarrollo del software y la funcionalidad del mismo.

Por otro lado, especialmente cuando de desarrollo de software para niños se trata, se debe tener muy en cuenta la usabilidad, y además esto se debe hacer desde el

inicio del proceso de desarrollo, esto es especialmente relevante si lo que se va a desarrollar es una herramienta tan centrada en el usuario como es el caso que nos ocupa, ya que en eAula el usuario es el principal factor a tener en cuenta. Por ello la herramienta debe tener un alto grado de usabilidad, por lo que esta característica de calidad se ha considerado a lo largo de todo el desarrollo de este proyecto.

Además de la usabilidad, se han utilizado técnicas de Ingeniería del Software para la captura de los requisitos funcionales, específicamente casos de uso y diagramas de secuencia y clases. Así mismo, se han utilizado técnicas de Interacción Persona-Ordenador para poder describir aquellas necesidades desde el punto de vista del usuario, utilizando para ello diagramas de tareas, presentación y técnicas de prototipado.

Con este proyecto, se ha puesto en práctica un proceso de desarrollo basado en las disciplinas de Ingeniería del Software e Interacción Persona-Ordenador y se ha desarrollado una herramienta que consta de tres aplicaciones: Una para el profesor, otra para el alumno y una tercera (servidor) que es la encargada de la conexión entre las dos anteriores de un modo totalmente transparente para los usuarios, que pueden utilizar sus aplicaciones sin preocuparse de cómo y de cuándo se envían los datos entre ellas.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1 INTRODUCCIÓN

1.1 Consideraciones previas

De un tiempo a esta parte nos encontramos con que tanto la Informática como los ordenadores cada vez más forman parte de nuestro día a día y por tanto es importante en la sociedad en la que vivimos estar familiarizados con ellos para llevar a cabo una correcta realización de nuestro trabajo, prácticamente sea cual sea este, ya que cada vez están más informatizadas todas las tareas habituales.

Teniendo en cuenta esto, es lógico intentar que cada vez desde más pequeños se vayan adquiriendo unos conocimientos mínimos sobre este campo y se sepa manejar un ordenador, aunque sea de la forma más básica, ya que el contacto con la tecnología es cada vez más temprano.

Por esto, resulta muy interesante incluir en la educación de los niños desde sus inicios en la vida escolar (3-5 años), el uso de los ordenadores para que aprendan a manejarse con ellos sin que les suponga ningún esfuerzo. Además mediante el uso de los mismos nos encontramos ante la posibilidad de ofrecer un tratamiento más personalizado, ya que dentro de una misma clase puede haber varios alumnos realizando diferentes tareas y estar todos ellos supervisados por el profesor en tiempo real.

Para ello, aplicaciones como la que nos ocupa en este proyecto resultan interesantes ya que mientras el niño cree que esta jugando, esta aprendiendo dos cosas, por un lado los conceptos propios de su curso, mediante actividades propias para su edad, teniendo en cuenta, que ésta aplicación no ocuparía el 100% de su tiempo, sino

que se alternaría con las actividades clásicas y típicas para el rango de edad que nos ocupa; y por otro lado a manejarse con un ordenador o por lo menos a que éste le resulte familiar.

También se deberá tener en cuenta en el desarrollo de la aplicación que tal y como se indica en el currículo infantil del MEC (Ministerio de Educación y Ciencia), el cual se puede consultar de manera integra en la siguiente URL, http://www.mec.es/educa/jsp/plantilla.jsp?id=9&area=sistema-educativo., teniendo en cuenta la receptividad de los alumnos en una etapa tan temprana de su vida, se hace preciso dotar a este nivel educativo de cuantos elementos puedan contribuir a garantizar su calidad. Al estar referida a los niños de 3 a 5 años, la enseñanza ha de presentarse de forma globalizada y tener muy en cuenta las características individuales de los alumnos.

Son particularmente importantes en este nivel la evaluación y el seguimiento de los procesos de enseñanza y aprendizaje para conocer el grado de desarrollo que se va produciendo en cada alumno, las respuestas que provocan en cada uno los procesos que se plantean y el modo en que van desarrollándose sus capacidades.

Así mismo, hoy es especialmente necesario familiarizar a los niños con las tecnologías de la información y de la comunicación y fomentar su uso progresivo como un recurso más, ya que la mayor parte de estos niños las encuentran integradas en su entorno y deberán utilizarlas de forma ordinaria a lo largo de su vida. El aprendizaje y utilización en la escuela de las tecnologías de la información y de la comunicación deben ajustarse a su proceso madurativo.

Por este motivo se pretende llevar a cabo en este proyecto el desarrollo de una herramienta que cumpla estas características, es decir, que tenga en cuenta que va dirigida a niños muy pequeños, que permita que se puedan satisfacer las necesidades individuales de cada uno de ellos y que, tal y como se ha dicho antes, inicie a los alumnos en el uso de la nuevas tecnologías.

1.2 Motivación del proyecto

La idea de realizar este proyecto, surge después de haber visto varias herramientas que servían para presentar diversos conceptos mediante el juego; pero la mayoría de las veces estas herramientas estaban destinadas a unos usuarios de más edad que la que nos ocupa.

Lo que se pretende, es hacer una herramienta de este tipo, pero dedicada a un usuario con edades comprendidas entre los 3 y los 5 años. Así como poner en práctica conocimientos procedentes de diferentes disciplinas como son Ingeniería del Software ó IPO (Interacción persona ordenador), disciplinas que serán tratadas en el capítulo 2 de esta memoria.

Se intenta por tanto, llevar a cabo una propuesta metodológica que permita el desarrollo sistemático de aplicaciones software educativas pero no exentas de calidad y diseño estético/artístico destinadas a niños de edades entre los 3 y los 5 años donde se utilicen técnicas procedentes de distintas disciplinas como son la ingeniería del software y la interacción persona ordenador (IS/ IPO).

1.3 Objetivos para su realización

Lo que se pretende con este proyecto, es construir una aplicación que cumpla las características citadas anteriormente, es decir, llevar a cabo el desarrollo de una aplicación que sirva para que los niños aprendan mientras se divierten, del mismo modo se pretende que entren desde pequeños en el mundo de la Informática, que como se ha comentado antes es un sector que está cada vez más en auge y el que probablemente tendrán que utilizar en sus futuros trabajos e incluso en su vida cotidiana.

Además, diversos estudios que se han ido realizando a lo largo de estos años indican que el uso de los juegos es una muy buena manera de aprendizaje, ya que la persona que está interactuando con el juego no es consciente de que está aprendiendo mientras se entretiene jugando.

1.4 Organización de esta memoria

El presente proyecto se haya dividido en los siguientes capítulos:

En el **capítulo primero**, que es donde estamos ahora, se da una pequeña introducción sobre el proyecto donde se indica las consideraciones que se han tenido en cuenta para la realización del mismo, así como las razones que han llevado a realizarlo y los objetivos que se pretenden alcanzar con su realización.

En el **capítulo segundo**, se hará una introducción a las disciplinas utilizadas para el desarrollo del proyecto haciendo una comparativa entre ambas y se presentará la idea del juego como actividad educativa haciendo una reflexión sobre por qué debe ser

considerado de este modo; así mismo se hablará sobre algunas herramientas similares a la correspondiente a este proyecto y por último se ofrecerán unas conclusiones sobre el tema tratado.

En el **capítulo tercero**, se tendrá en cuenta el concepto de usabilidad así como los criterios ergonómicos, teniendo en cuenta su influencia en la aplicación que nos ocupa; analizando porque deben usarse y dónde se han tenido en cuenta durante el desarrollo de este proyecto.

En el **capítulo cuarto**, se presentará el proceso seguido para la confección de la herramienta eAula, para ello se seguirán técnicas provenientes de la Ingeniería del Software como son los casos de uso, diagramas de secuencia, diagramas de clases..., y también se seguirán técnicas relacionadas con la Interacción Persona-Ordenador como lo son el diagrama de tareas y el diagrama de presentación.

En el **capítulo quinto**, se hará una descripción de cómo se ha llevado a cabo la implementación de eAula, así como las partes de las que se compone y el funcionamiento de cada una de ellas. Se hará un análisis de la aplicación explicando que herramientas se han utilizado para realizar cada una de sus partes y cual es el funcionamiento final, esto último se llevará a cabo mediante un manual de usuario.

En el **capítulo sexto**, se hará una exposición de las conclusiones que se han obtenido de la realización de este proyecto y se indicarán posibles trabajos futuros para realizar a raíz del mismo.

CAPÍTULO 2 ESTADO DEL ARTE

Ingeniería del Software (IS) e Interacción Persona-Ordenador (IPO) son dos disciplinas que buscan aportar métodos y herramientas para potenciar la calidad que ofrecen los productos software. Los puntos de vista que se ofrecen en ellas son diferentes, aunque ambas disciplinas se solapan en parte de sus ámbitos de alcance.

La IS aporta procesos, métodos, notaciones y herramientas que afectan tanto al proceso como al producto, contempla factores internos y externos de un producto software. Mientras, la IPO ofrece otros métodos, en algunos casos complementarios, e intereses que se centran en el proceso, en el producto y en la identificación de dónde, cómo y por quién es utilizado el producto software. La IS está preocupada por ofrecer una visión arquitectónica del sistema, para ello ha propuesto, entre otras facilidades, un Proceso Unificado (UP) y un Lenguaje Unificado de Modelado ™ (UML[1]). IPO busca la complicidad del usuario y la determinación de su contexto y para ello propone un Diseño Centrado en el Usuario.

La IS se centra fundamentalmente en controlar costes, y productividad relacionados con el desarrollo de software y de proporcionar procesos para asegurar de forma sistemática el logro de los mismos. La IPO se centra en la usabilidad y en la necesaria aceptación de un producto en función de las características que como vehículo ofrece y de cómo se las ofrece.

[1] Unified Modeling Language ™. http://www.uml.org

Por otro lado se tiene que el juego en la edad infantil es una actividad que se puede llevar a cabo de diferentes formas, una de ellas es utilizar el juego del niño para que éste ponga en marcha su imaginación, exprese su manera de ver el mundo y desarrolle su creatividad.

El juego, tiene por tanto, un papel muy importante en el desarrollo de la personalidad de cada niño. Tanto en casa, como en la escuela los niños ocupan gran parte de su tiempo con el juego bien sea simplemente para divertirse o para aprender mientras juegan.

En este capítulo se pretende explicar las dos disciplinas citadas anteriormente, para ello se verá una introducción de las mismas y también se hará una comparativa para ver las diferencias y semejanzas que se pueden encontrar entre ambas, de este modo se introducirá al lector en los procesos de ingeniería que se utilizarán en el desarrollo del proyecto. Por otro lado, se pretende defender la opción del juego como actividad educativa, dando razones por las cuales es bueno utilizar el juego como modo de enseñanza para los niños.

2.1 Introducción a la Ingeniería del Software (IS)

Por Ingeniería del Software se entiende el establecimiento y uso de principios robustos de la Ingeniería con el fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales. [Bauer, 1997]

Otra definición de IS que ofrece la IEEE establece que la misma es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software.

La IS es una tecnología multicapa [Pressman, 1992], como se puede observar en la figura 2.1:

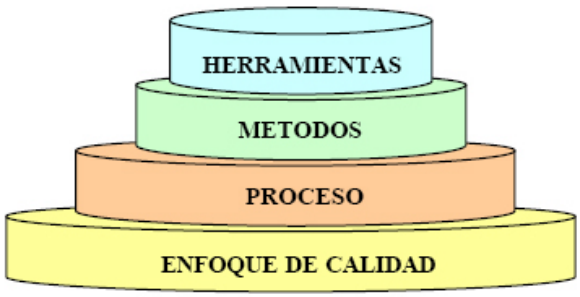

Figura 2.1 Tecnología multicapa de la IS.

Los cimientos de esta tecnología multicapa están orientados hacia la ***calidad***. La gestión de la calidad fomenta una cultura continua de mejora del proceso.

El fundamento de la IS es la capa de proceso. El ***proceso*** es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la IS. Las áreas clave del proceso forman la base del control de gestión de proyectos del software y establecen el contexto en el que se aplican los métodos técnicos, se producen resultados del trabajo, se establecen hitos, se asegura la calidad y se gestiona el cambio de manera adecuada.

Los ***métodos*** indican cómo construir de manera técnica el software, y abarcan una gama de tareas que incluyen análisis de requisitos, diseño, construcción, prueba y mantenimiento. Los métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

Por último, las ***herramientas*** proporcionan un soporte automático o semiautomático para el proceso y para los métodos. Todos los elementos que se han comentado son importantes e influyen en el desarrollo del software.

2.1.1 Evolución del desarrollo del software

Durante los ***primeros años*** de la era del ordenador, el software se contemplaba como un añadido. La programación era un "arte de andar por casa" para el que existían pocos métodos sistemáticos. El desarrollo del software se realizaba virtualmente sin ninguna planificación, hasta que los planes comenzaron a descalabrarse y los costes a correr, esta forma de desarrollar software se conoce con el nombre de *Código Espaghetti*. Los programadores trataban de hacer las cosas bien, y con un esfuerzo heroico, a menudo salían con éxito.

El software se diseñaba a medida para cada aplicación y tenía una distribución relativamente pequeña. En esta época se tenía que el software como producto, es decir, programas que se desarrollen para vender a uno o más clientes, estaba dando sus primeros pasos. La mayoría del software se desarrollaba y era utilizado por la misma persona u organización. La misma persona lo escribía, lo ejecutaba y, si fallaba, lo depuraba. Debido a este entorno personalizado del software, el diseño era un proceso implícito, realizado en la mente de alguien y, la documentación normalmente no existía.

La ***segunda generación*** en la evolución de los sistemas de ordenador se extiende desde la mitad de la década de los sesenta hasta finales de los setenta. La multiprogramación y los sistemas multiusuario introdujeron nuevos conceptos de interacción hombre - maquina. Las técnicas interactivas abrieron un nuevo mundo de aplicaciones y nuevos niveles de sofisticación del hardware y del software. Los sistemas de tiempo real podían recoger, analizar y transformar datos de múltiples fuentes, controlando así los procesos y produciendo salidas en milisegundos en lugar de minutos. Los avances en los dispositivos de almacenamiento en línea condujeron a la primera generación de sistemas de gestión de bases de datos.

La segunda generación se caracterizó también por el establecimiento del software como producto y la llegada de las "casas del software". En este punto el software ya se desarrollaba para tener una amplia distribución en un mercado multidisciplinar. Los patronos de la industria, del gobierno y de la universidad se aprestaban a "desarrollar el mejor paquete de software" y ganar así mucho dinero.

Conforme crecía el número de sistemas informáticos, comenzaron a extenderse las bibliotecas de software. Las casas desarrollaban proyectos en los que se producían programas de decenas de miles de sentencia fuente. Los productos del software comprados al exterior incorporaban cientos de miles de nuevas sentencias. Todos esos programas, todas esas sentencias fuente tenían que ser corregidos cuando se detectaban fallos, modificados cuando cambiaban los requisitos de los usuarios o adaptados a nuevos dispositivos hardware que se hubieran adquirido. Estas actividades se llamaron colectivamente *mantenimiento del software*. El esfuerzo empleado en el mantenimiento del software comenzó a absorber recursos en una medida alarmante.

La naturaleza personalizada de algunos programas los hacía virtualmente imposibles de mantener. En este punto, por tanto, ya había comenzado la *“crisis del software”*.

La ***tercera generación*** en la evolución de los sistemas de ordenador comenzó a mediados de los años setenta y continuo mas allá de una década. El sistema distribuido, múltiples ordenadores, cada uno ejecutando funciones concurrente y comunicándose con algún otro, incrementó notablemente la complejidad de los sistemas informáticos. Las redes de área local y de área global, las comunicaciones digitales de alto ancho de

banda y la creciente demanda de acceso "instantáneo" a los datos, supusieron una fuerte presión sobre los desarrolladores del software.

La conclusión de la tercera generación se caracterizó por la llegada y amplio uso de los microprocesadores. El microprocesador ha producido un extenso grupo de productos inteligentes, desde automóviles hasta hornos microondas, desde robots industriales a equipos de diagnósticos de suero sanguíneo, así como al ordenador personal. En menos de una década, los ordenadores personales llegarán a ser fácilmente accesibles al público, algo que hasta este momento había sido impensable.

La ***cuarta generación*** de la evolución de los sistemas informáticos se aleja de los ordenadores individuales y de los programas de ordenador, dirigiéndose al impacto colectivo de los ordenadores y del software. Potentes maquinas personales controladas por sistemas operativos sofisticados, en redes globales y locales, acompañadas por aplicaciones de software avanzadas se han convertido en la norma.

Las arquitecturas informáticas están cambiando de entornos centralizados de grandes ordenadores a entornos descentralizados cliente/servidor. Las redes de información en todo el mundo proporcionan una infraestructura que iguala a expertos y políticos en pensar sobre una *"superautopista de información"* y una *"conexión del ciberespacio"*.

La industria del software ya es la cuna de la economía del mundo. Las técnicas de la cuarta generación para el desarrollo del software están cambiando en la forma en que la comunidad del software construye programas informáticos. Las tecnologías orientadas a objetos están desplazando rápidamente los enfoques de desarrollo de software más convencionales en muchas áreas de aplicaciones.

A medida que la cuarta generación progresa, comenzaron a surgir nuevas tecnologías. Las tecnologías orientadas a objetos desplazaron rápidamente los enfoques de desarrollo de software más convencionales en muchas áreas de aplicaciones ya que aportan muchas ventajas como extensibilidad, reutilización, fiabilidad…además, la orientación a objetos ofrece dos técnicas que extienden la flexibilidad de los paquetes, la sobrecarga de rutinas y la genericidad.

La *sobrecarga de rutinas* (ligadura dinámica) es reutilizar el mismo nombre para más de una operación, en tiempo de ejecución se resuelve la llamada a la función y según el tipo de objeto que la llama se ejecuta un método u otro. Para funcionar

correctamente, requiere de los mecanismos de orientación a objetos y principalmente de la herencia.

La *genericidad* es la disponibilidad de módulos parametrizados por tipos. Según el tipo y número de parámetros con el que se llame a una función se ejecutará una u otra.

La programación orientada a objetos (POO) supuso un gran paso en la IS, ya que presentaba un modelo de objetos que parecía encajar de manera adecuada con los problemas reales. La cuestión era saber descomponer de la mejor manera el dominio del problema al que nos enfrentáramos, encapsulando cada concepto en lo que se dio en llamar objetos y haciéndoles interactuar entre ellos, habiéndoles dotado de una serie de propiedades. Surgieron así numerosas metodologías para ayudar en tal proceso de descomposición y aparecieron herramientas que incluso automatizaban parte del proceso.

El problema se encontraba en que la programación se realizaba para una plataforma específica y los problemas que se trataban eran específicos de cada plataforma, el siguiente paso en la programación es hacia una programación más abstracta, se trata de abstraer toda la información a modelos independientes de la plataforma y posteriormente una traducción a una plataforma específica.

El foco del desarrollo se traslada de escribir código para una plataforma específica a desarrollar una solución efectiva a nivel de negocio. Los modelos se utilizan para automatizar la generación del código de muchas aplicaciones, generando modelos específicos de una plataforma a partir del modelo independientes de la plataforma, como el propio código a partir de modelos específicos de una plataforma.

Es aquí donde empieza la ***quinta generación*** con los desarrollos basados en modelos. Los modelos se utilizan para dirigir el código, es decir, desarrollar código basado en ellos, y no viceversa. De esta forma, se mantiene la consistencia entre el modelo y lo que verdaderamente hace el sistema.

A continuación, en la figura 2.2, se puede ver un diagrama de la evolución del software desde la primera generación hasta la quinta.

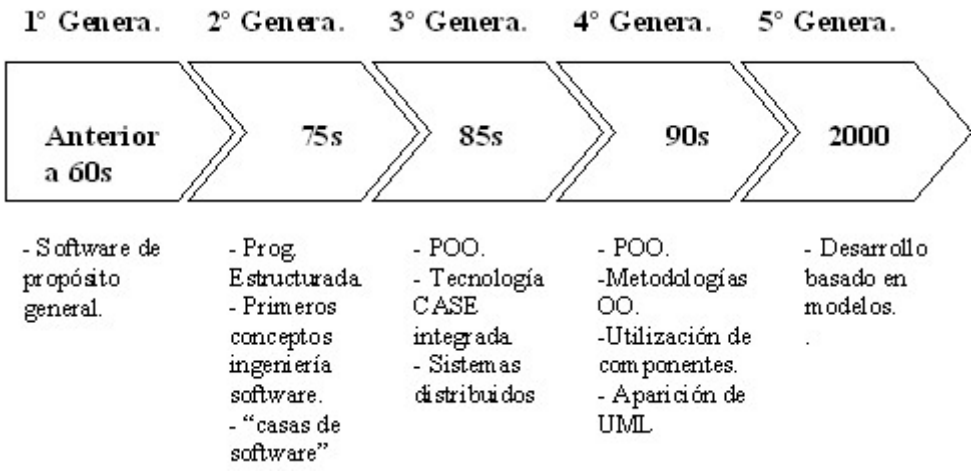

Figura 2.2 Evolución del software.

2.1.1.1 El Lenguaje Unificado de Modelos (UML)

Desde los inicios de la Informática se han estado utilizando distintas formas de representar los diseños de manera más bien personal o con algún modelo gráfico. La falta de estandarización en la manera de representar gráficamente un modelo impedía que los diseños gráficos realizados pudieran compartirse fácilmente entre distintos diseñadores lo que supone un verdadero obstáculo. Se necesitaba un lenguaje no sólo para comunicar ideas a otros desarrolladores sino también para servir de apoyo en los procesos de análisis de un problema. Con este objetivo nació el Lenguaje Unificado de Modelado (UML). A continuación, en la figura 2.3 se puede ver la evolución histórica que ha tenido este lenguaje.

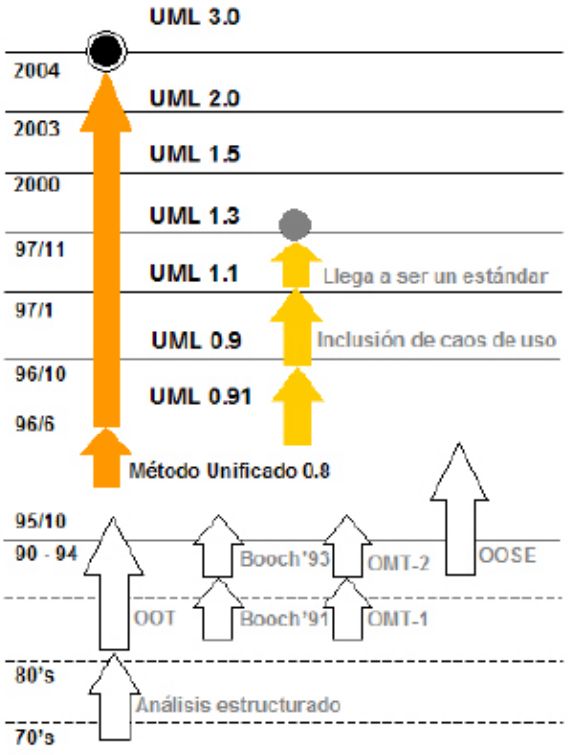

Figura 2.3 Evolución histórica de UML.

UML fue adoptado en noviembre de 1997 por OMG (*Object Management Group*) como una de sus especificaciones y desde entonces se ha convertido en un estándar de hecho para visualizar, especificar y documentar los modelos que se crean durante la aplicación de un proceso software.

Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. Los diagramas más interesantes son los de casos de uso, clases y secuencia. El resto de diagramas muestran distintos aspectos del sistema a modelar. Para modelar el comportamiento dinámico del sistema están los diagramas de interacción, colaboración, estados y actividades. Los diagramas de componentes y despliegue están enfocados a la implementación del sistema.

Con la utilización de UML podemos comunicar y compartir el conocimiento de una arquitectura donde se entrelazan los procesos de negocio de la organización y las aplicaciones informáticas, mediante la combinación simultánea y sinérgica de las siguientes tareas:

- Definir conceptos a través de sus atributos distintivos y trazar sus límites.
- Formalizar unas reglas de relación y actuación.
- Hacer visible la granularidad y las relaciones entre elementos.
- Mantener la trazabilidad entre la concepción de un problema y su posterior solución y viceversa.
- Tomar decisiones y evaluar su impacto dentro de una arquitectura definida.
- Organizar la experiencia de los usuarios en patrones de conocimiento.
- Comprobar la coherencia, completitud y usabilidad de los entregables.
- Alinear la solución tecnológica con la cadena de valor de los actores.
- Usar un vocabulario controlado dentro de un espacio de colaboración.
- Realizar un plan de producción en base a una factoría de componente.

2.1.2 El Proceso Unificado de Desarrollo (RUP)

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML propusieron su propia metodología de desarrollo, denominada *Proceso Unificado de Desarrollo*. En la figura 2.4 se tiene un gráfico de RUP.

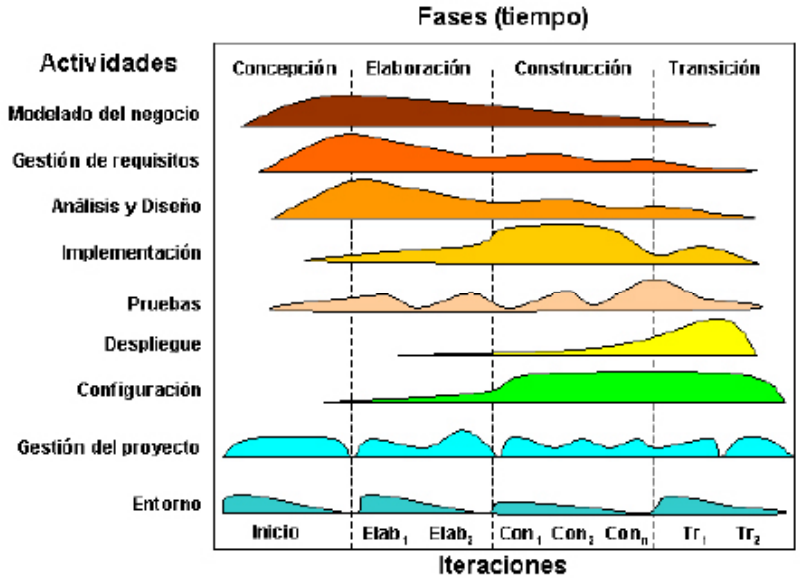

Figura 2.4 RUP: Proceso Unificado de Desarrollo Software.

El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas. El Proceso Unificado utiliza UML

para expresar gráficamente todos los esquemas de un sistema software. Pero, realmente, los aspectos que definen un Proceso Unificado son tres:

- **Dirigido por casos de uso:** Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes.
- C**entrado en la arquitectura:** En la arquitectura relacionada con la construcción de edificios, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, conducciones eléctricas, fontanería, etc. Cada uno de estos aspectos está representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema.
- **Iterativo e incremental:** Todo sistema informático complejo supone un gran esfuerzo que puede durar desde varios meses hasta años. Por lo tanto, lo más práctico es dividir un proyecto en varias fases. En los proyectos en los que se realizan varios tipos de trabajo (flujos), cada recorrido por las distintas fases se denomina iteración. A su vez, cada iteración parte de la anterior incrementando o revisando funcionalidad implementada (proceso).

En el capítulo siguiente de esta memoria se realizarán los diagramas correspondientes al proceso de desarrollo de la aplicación que nos ocupa.

2.2 Introducción a la Interacción Persona-Ordenador (IPO)

La disciplina IPO se ocupa del estudio y elaboración de productos software que faciliten la realización de tareas a sus usuarios atendiendo a diversos criterios, como pueden ser las facilidades de uso, el tiempo de ejecución, evitar errores y, en consecuencia, aumentar su satisfacción. En (Perlman, 1996) se define IPO como aquella disciplina relacionada con el diseño, evaluación e implementación de sistemas de computación interactivos para uso humano y el estudio de los principales fenómenos asociados con ello.

La IPO se ocupa del análisis y diseño de interfaces entre el hombre y la máquina, estas interfaces se conocen como interfaces de usuario.

[Booth, 1989] define interacción como el intercambio de símbolos entre dos o más partes, asignando los participantes en el proceso comunicativo los significados a esos símbolos. La interacción entre una persona y un ordenador puede ser analizada en función de su estilo, de su estructura, de su contenido y de su comportamiento. El estilo se refiere a la forma en que el usuario introduce y recibe la información. La estructura tiene que ver con la forma de organizar las componentes. El contenido trata el significado semántico y pragmático que se produce durante el diálogo. Por último, el comportamiento define las acciones que serán realizadas al interactuar con los objetos de la interfaz.

La investigación en IPO está orientada al desarrollo de nuevos dispositivos e identificación de nuevos estilos de interacción donde se incorporan las capacidades del lenguaje entre personas. Para ello, se basa en el establecimiento de similitudes entre esos dos tipos de diálogo, es decir, ambas partes (emisor y receptor) necesitan compartir unos conceptos y se encuentran en un contexto. El contexto incluye características fundamentales de las personas y de los sistemas. Esas características son tanto las limitaciones físicas (p. e., la capacidad de memoria, los límites de transferencia de información, la habilidad lógica de la máquina) como las conceptuales (p.ej. el conocimiento de las tareas y de los modelos mentales de los objetos y de los procesos) [Marchionini et al., 2003].

2.2.1 Evolución de la Interacción Persona-Ordenador

El interés por el estudio de la IPO está inevitablemente ligado al de la Informática, ya que siempre ha existido la necesidad de estudiar la necesaria comunicación que debe establecerse entre los ordenadores y las personas que los usan. Además, ese mismo estudio es paralelo al de la ergonomía, que estudia el manejo de máquinas por parte de personas y cómo éste se produce. A pesar de que podemos situar el inicio de un interés por aspectos relacionados con la IPO en los años 50, hasta la década de los 70 no se aprecia un volumen importante de publicaciones científicas en el ese campo. De la misma forma, los avances de sus aplicaciones no resultan notables sino a partir de los años 80, momento en que comienzan a extenderse el uso de los PCs.

En esa década de los 80 se produjo también una explosión de publicaciones

periódicas, asociaciones y encuentros especializados para el tratamiento de consideraciones relacionadas con la IPO, todo ello se vio favorecido por los grandes avances logrados en las tecnologías informáticas. Estudios más detallados de la evolución de la IPO pueden encontrarse en [Carroll, 2001] y [Karat et al., 2003].

2.2.2 Diseño Centrado en el Usuario (DCU)

Si el proceso que determinaba la realización de propuestas metodológicas en IS, con el paso del tiempo, ha llegado a ser un Proceso Unificado, en IPO lo que marca la realización de propuestas similares es la puesta en práctica de metodologías y técnicas de Diseño Centrado en el Usuario (DCU).

Por Diseño Centrado en el Usuario (DCU) se entiende aquel proceso por el que al usuario y a sus datos se les establece como criterio con el que evaluar los diseños, y como la fuente de ideas de diseño. Los principios para el desarrollo de dicho proceso pasan por:

- **conocer al usuario y sus tareas**, según lo cual los diseñadores deben saber quiénes son los usuarios, estudiando sus capacidades cognitivas, comportamientos, y actitudes, y por otro lado la naturaleza de las tareas que llevan a cabo.
- **realización de medidas empíricas**. En cuanto sea posible, y cuanto antes en el proceso de desarrollo, a los usuarios se les deben proporcionar prototipos y simulaciones con los que llevar a cabo sus tareas, y con ellos deben observarse, grabarse y analizarse las prestaciones y las reacciones que estos usuarios experimentan.
- **aplicar un diseño iterativo**. Fruto de observar problemas con la puesta en práctica de evaluaciones reiteradas, se debe proceder, a través del diseño, a eliminar aquellos problemas surjan en una versión.

2.3 IS Vs IPO

La construcción de software según los métodos y prácticas de la IS no trata de forma adecuada la usabilidad a lo largo del proceso de desarrollo. Así, una organización

que quiera construir software usable y que tenga un proceso definido, tiene que realizar un esfuerzo de integración de las prácticas IPO en su proceso. Debido a que dichas prácticas no están encajadas con las prácticas IS, tal integración no resulta trivial y acaba por suponer un esfuerzo considerable. Se necesitan estrategias que proporcionen a los ingenieros software descripciones de las técnicas y actividades IPO que pueden formar parte de su proceso de desarrollo. Las principales dificultades para afrontar dicho problema son las siguientes:

- Los procesos de desarrollo del campo de la IPO no están descritos en términos del proceso de desarrollo de la IS.
- Existen importantes diferencias de terminología entre la IPO y la IS.
- Es necesario hacer comprender a los desarrolladores por qué es necesario tratar la usabilidad desde los momentos tempranos del desarrollo, como defienden los expertos IPO.

En la figura 2.5 mostrada a continuación se puede ver de manera gráfica la diferencia entre ambas disciplinas.

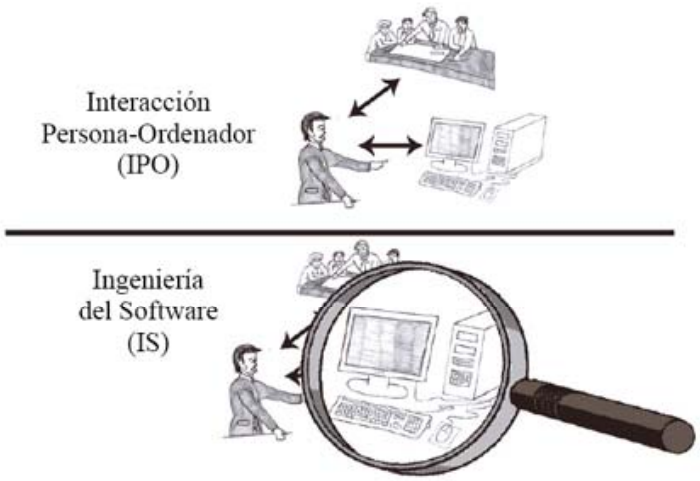

Figura 2.5 Comparativa entre la IPO y la IS.

Los procesos de desarrollo del campo de la IPO no tienen el nivel de detalle habitual en los procesos IS. Los procesos IPO no son procesos completos que se ocupen de todos los aspectos del desarrollo. El proceso de usabilidad por tanto debe integrarse en el proceso de desarrollo general.

La visión de los procesos separados provoca el siguiente conflicto de intereses: La definición de qué software hay que construir(frente a cómo construirlo), lo que constituye el área de los requisitos, es un área que ambas disciplinas consideran como su ámbito propio; lo más lógico en este caso sería por tanto la integración de ambas perspectivas.

La IPO trabaja con una perspectiva centrada en el usuario, esta preocupación por el usuario, el empeño en que su punto de vista sea considerado en todas las actividades del desarrollo, es preciso inculcarlo en ciertas actividades del proceso IS, pues éstas se verían enriquecidas cumpliendo más satisfactoriamente su objetivo.

La idea de la usabilidad como un atributo de calidad del cual se ocupa únicamente la actividad de diseño de la Interfaz de Usuario (IU) contribuye a una confusión terminológica. Gran parte de los procesos IPO describen como su objetivo el diseño de la IU; del mismo modo, un ingeniero software interpreta a menudo que las técnicas que se incluyen en dicho proceso únicamente se aplican al diseño de la parte visual de la IU y su comportamiento asociado. Por el contrario, en dichos procesos IPO de “diseño” se incluye una parte muy importante de actividades de conceptualización, de diseño de invención.

2.4 El juego como elemento de ayuda al aprendizaje en etapas tempranas de desarrollo del niño/a

El juego se suele asociar con la diversión y el descanso, pero su importancia es mucho mayor. [Rodríguez et al., 2003] Desde que aparecen las primera teorías psicológicas sobre el juego, en la segunda mitad del siglo XIX, han sido muchas y muy diversas las explicaciones que se han dado sobre el papel que ha desempeñado y sigue desempeñando en la vida humana. Para unos es una forma de quemar las energías sobrantes del individuo; para otros un modo de relajación. Ha habido autores que lo han considerado como una especie de preparación para la vida adulta, mientras que para otros es una práctica que permite a los individuos recapitular la historia de la humanidad y conectar con ella. No han faltado quienes han relacionado el juego con la necesidad de satisfacer los impulsos instintivos, o como una forma de resolver los problemas.

La psicología actual ha destacado la importancia que tiene el juego en la mayoría de los procesos de desarrollo cognitivo, social, emocional o afectivo, de personalidad y moral. También ha incidido en su naturaleza educadora y motivadora; en su capacidad para favorecer el desarrollo de la creatividad, la expresión o la motricidad; en su papel mediador entre el individuo y la sociedad en la que éste vive: mediante la actividad lúdica las culturas transmiten valores, normas de conducta y educan a sus miembros, de forma que los juegos se convierten en un reflejo de una determinada situación social y cultural.

Siempre ha existido un amplio consenso, aunque desde distintas perspectivas, para considerar el juego como un factor importante del desarrollo tanto físico como psíquico del ser humano, especialmente en la etapa de infantil, donde aparece como una actividad natural y espontánea, a la que el niño le dedica todo el tiempo posible. Cualquier capacidad del niño se desarrolla más eficazmente en el juego que fuera de él.

De ahí la importancia que tiene el juego como factor potenciador del desarrollo y también del aprendizaje. De ahí lo necesario que es ofrecer al niño contextos educativos familiares y escolares con muchas oportunidades lúdicas. Especialmente en la edad infantil, donde las actividades de juego y aprendizaje resultan ser y se convierten en la actividad predominante. El juego es un instrumento didáctico básico para la integración del niño a su entorno escolar y social.

Los juegos de ejercicio, al principio relacionados con la repetición de conductas motoras, la adquisición de resultados inmediatos y el placer de hacer cosas, más que con la contemplación de la obra concluida ayudan al niño al desarrollo de sus habilidades motrices, al conocimiento del propio cuerpo y a familiarizarse y comprender los objetos y sus características. La familiaridad con los objetos, coordinada con un mayor desarrollo de las capacidades cognitivas, introduce al niño en nuevos campos de juego cada vez más complejos y relacionados con la comprensión de la conservación, la clasificación la seriación, la numeración y otras actividades.

Los juegos simbólicos, por otra parte, implican el desarrollo del lenguaje y de la capacidad de representación mental infantiles. La imaginación, la representación, la simulación y la simbolización son cruciales en este tipo de juegos, permiten a niño dar todo tipo de significados a objetos, acontecimientos, escenas y personajes; representar

personajes más o menos estereotipados; le ayuda a liberarse del presente y a acceder al pasado y al futuro, a la creatividad, al mundo real y al mundo de la ilusión.

Los distintos tipos de juego en función del número de participantes, de sus contenidos o materiales y de las distintas etapas evolutivas facilitan y aceleran los procesos de aprendizaje. Su función educativa se observa en su capacidad de estimular no sólo el desarrollo de las distintas capacidades del niño, sino también su interés por descubrir y conocer el mundo que le rodea, e incluso conocerse a sí mismo. De hecho, numerosos investigadores de la educación han llegado a la conclusión de que el aprendizaje más valioso a ciertas edades es el que se produce a través del juego, de todo tipo de juegos, aún los que no están directamente pensados como juegos educativos.

Y esto es así porque durante el juego el niño siempre está por encima de su nivel efectivo de desarrollo, superando en eficacia la realización de cualquier otra tarea cotidiana que no esté contextualizada como actividad lúdica. Su capacidad de atención, interés, concentración, comprensión y memorización se duplican. Su motivación es mayor y su eficacia mejora. De forma que los efectos positivos del aprendizaje mediante el juego no sólo son inmediatos, en los aprendizajes del momento, sino que son observables, según diversas investigaciones, en los aprendizajes futuros y en los desarrollos psicomotor, cognitivo y socio emocional posteriores.

Las condiciones que hacen del juego una óptima actividad de aprendizaje y de desarrollo se resumen en que el niño se encuentra más libre y con menos responsabilidad; lo aprendido le resulta más familiar y significativo; se desenvuelve en el mundo de la imaginación y de la creatividad; y generalmente obliga a la interacción social con otros niños y adultos. En consecuencia, el colegio debe procurar restaurar el valor pedagógico del juego y tener previsto un tiempo y un espacio de juego.

La propia L.O.E. señala, cuando se refiere a la etapa de Educación Infantil, que es imprescindible destacar la importancia del juego como la actividad propia de esta etapa. En el juego se aúnan, por una parte, un fuerte carácter motivador y, por otra, importantes posibilidades para que el niño y la niña establezcan relaciones significativas y el profesorado organice contenidos diversos, siempre con carácter global, referidos sobre todo a los procedimientos y a las experiencias, enviando la falsa dicotomía entre juego y trabajo escolar.

En las Orientaciones Didácticas Generales de esta etapa educativa se considera que el juego es un instrumento privilegiado, para el desarrollo de las capacidades que se pretende que alcance el niño, por el grado de actividad que comporta, por su carácter motivador, por las situaciones en que se desarrolla y que permiten al niño globalizar, y por las posibilidades de participación e interacción que propicia, entre otros aspectos. El juego es un recurso que permite al niño hacer por sí solo aprendizajes significativos y que le ayuda a proponer y alcanzar metas concretas de forma relajada y con una actitud equilibrada, tranquila y de disfrute. Por ello, el educador, al planificar, debe partir del hecho de que el juego es una tarea en la que el niño y la niña hacen continuamente ensayos de nuevas adquisiciones, enfrentándose a ellas de manera voluntaria, espontánea y placentera.

En las Orientaciones Didácticas Específicas de cada una de las tres áreas de Educación Infantil se hace también mención al juego. Por ejemplo en el área de Identidad y Autonomía personal se habla de la planificación de espacios que inviten a los niños y niñas a realizar variadas actividades, que contribuyan al descubrimiento de su propio cuerpo y del de los demás, de sus posibilidades y limitaciones. En el área del Medio Físico y Social se dice que el educador ha de ofrecer al niño, principalmente en los primeros tramos de la etapa, actividades que posibiliten el juego, la manipulación, la interacción y la exploración directa del mundo que le rodea: actividades que deben adquirir mayor complejidad a medida que va creciendo. En el área de Comunicación y Representación, se señala que el juego es un elemento educativo de primer orden, que ofrece al niño la posibilidad de explorar las distintas formas de expresión e interactuar con los iguales y adultos.

Cuando se refiere a la Educación Primaria, como establecía la L.O.G.S.E. y recoge la L.O.E. se señala también que la actividad lúdica es un recurso especialmente adecuado en esta etapa, principalmente en algunas áreas. El juego está presente en los principios metodológicos de la etapa. Se dice que es necesario romper la aparente oposición entre juego y trabajo, que considera éste último asociado al esfuerzo para aprender, y el juego como diversión ociosa. En muchas ocasiones las actividades de enseñanza y aprendizaje tendrán un carácter lúdico y en otras exigirán de los alumnos y alumnas un mayor grado de esfuerzo, pero, en ambos casos, deberán ser motivadoras y

gratificantes, lo que es una condición indispensable para que el alumno construya sus aprendizajes significativos.

Incluso en el Proyecto Curricular, las situaciones de juego se incluyen entre los instrumentos de evaluación más útiles del proceso de aprendizaje de los alumnos. El proyecto curricular (currículo infantil) se puede leer de manera integra en la URL MEC: http://www.mec.es/educa/jsp/plantilla.jsp?id=9&area=sistema-educativo .

Por último, cabe señalar que la importancia de la función educativa del juego tanto en el medio familiar como escolar, no excluye otro tipo de actividades y aportaciones didácticas. La idoneidad del juego no debe suplantar otras formas de la enseñanza, donde el niño aprenda, por ejemplo, a desarrollar la concentración, las actitudes de trabajo y la aceptación de metas.

En los últimos años estamos asistiendo a un gran desarrollo de la Informática y de los juegos electrónicos. Mediante este tipo de juegos los niños, desde edades muy tempranas, pueden crecer con una tecnología a la que podrán sacarle provecho a lo largo de su vida, además de favorecer, dado su carácter motivador y estimulante para el niño actual, el desarrollo de su atención selectiva, percepción, memoria, coordinación, estrategias de solución de problemas, fantasía.....

Conseguir darle a un juego una naturaleza virtual puede abrir un abanico de posibilidades y oportunidades a niños que de otra forma estarían privados de la posibilidad de optimizar su desarrollo mediante materiales como éste, que en principio requiere de la manipulación. Mediante la Informática, los niños con minusvalías serias pueden conseguir una cierta independencia en movilidad, comunicación y control de su entorno, además de aumentar sus habilidades para la vida diaria, su interacción con otros y su capacidad de aprendizaje.

2.5 Herramientas similares a eAula

Teniendo en cuenta lo que se ha planteado en el punto anterior, a continuación se van a presentar varias aplicaciones del mismo tipo que eAula, es decir aplicaciones que usan la informática como modo de juego y a su vez el juego como modo de aprendizaje, en los siguientes apartados de este punto se irán mostrando distintas herramientas que están actualmente disponibles o que lo estarán en un breve período de tiempo.

2.5.1 JClic, ¿Qué es y como funciona?

JClic [http://clic.xtec.es/es/jclic/] es una herramienta con la que se pueden realizar tanto actividades de asociación como puzzles, sopas de letras, crucigramas o actividades de texto. Está compuesto por un entorno para la creación de actividades, el JClic author que es una herramienta que permite crear y modificar proyectos JClic y un entorno para la realización de dichas actividades, el JClic propiamente dicho.

Para utilizar esta herramienta lo primero que se debe hacer es instalarla y una vez que está instalada se debe cargar un determinado paquete de actividades (previamente construido mediante el JClic author) y a partir de aquí podemos empezar a realizar las actividades.

El menú principal que nos encontramos al iniciar el JClic, se muestra a continuación en la figura 2.6:

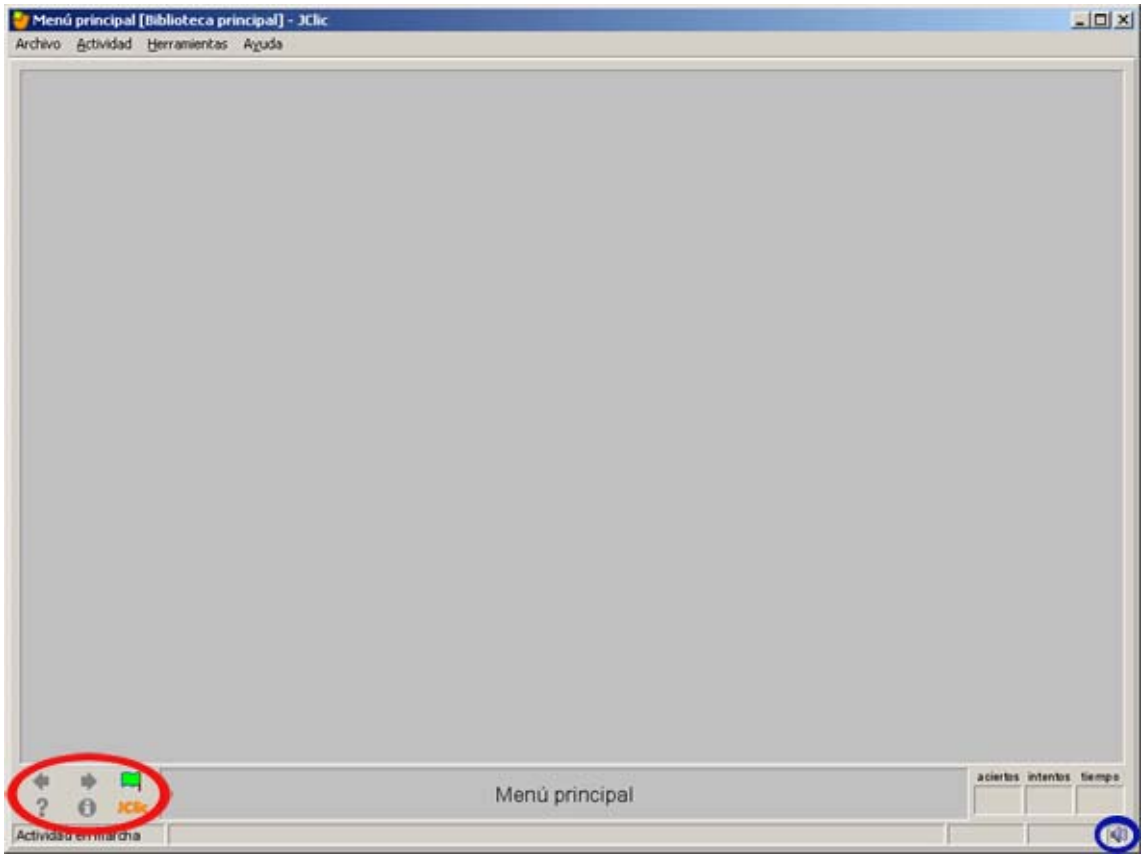

Figura 2.6 Pantalla de inicio de JClic.

Para empezar a realizar las actividades se debe cargar un proyecto (paquete de actividades). Una vez cargado el paquete de actividades nos podemos mover por las

actividades contenidas en el mismo mediante los botones que se tienen abajo a la izquierda (figura 2.7), con dichos botones nos podemos mover a la actividad siguiente y la anterior dentro del proyecto, podemos reiniciar la actividad en cualquier momento y tantas veces como se quiera, podemos obtener la solución del ejercicio (pulsando el botón de ayuda) y podemos ver la pantalla de información sobre JClic (figura 2.8.).

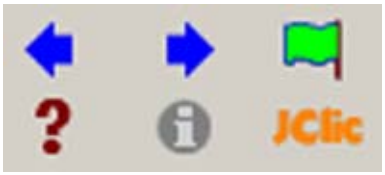

Figura 2.7 Botones de control de actividad en JClic.

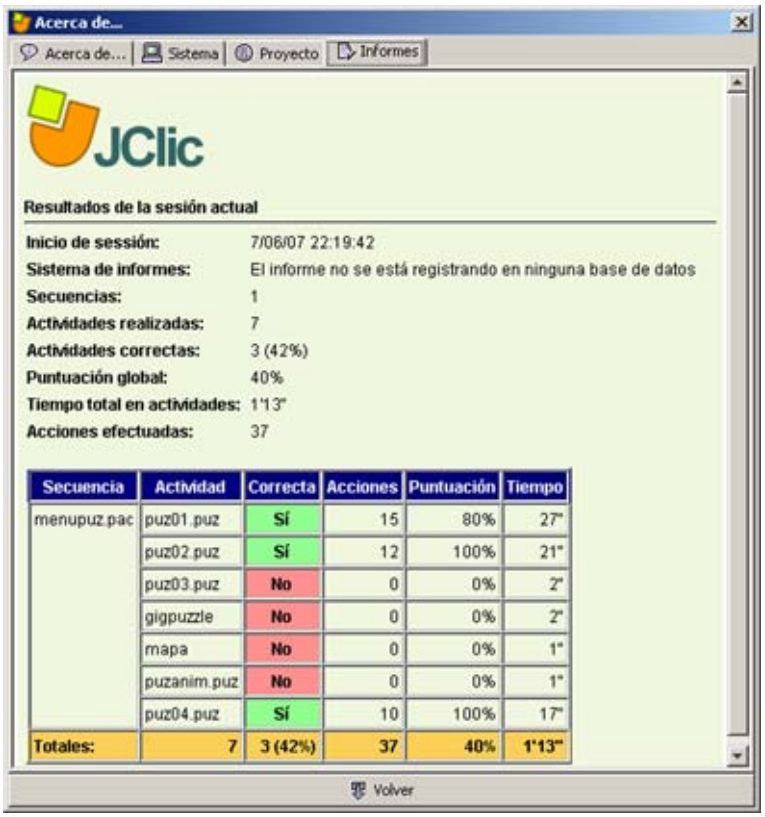

Figura 2.8 Informe de resultados de la sesión en JClic.

A continuación, en la figura 2.9 se muestran algunas de las actividades que se pueden realizar utilizando JClic.

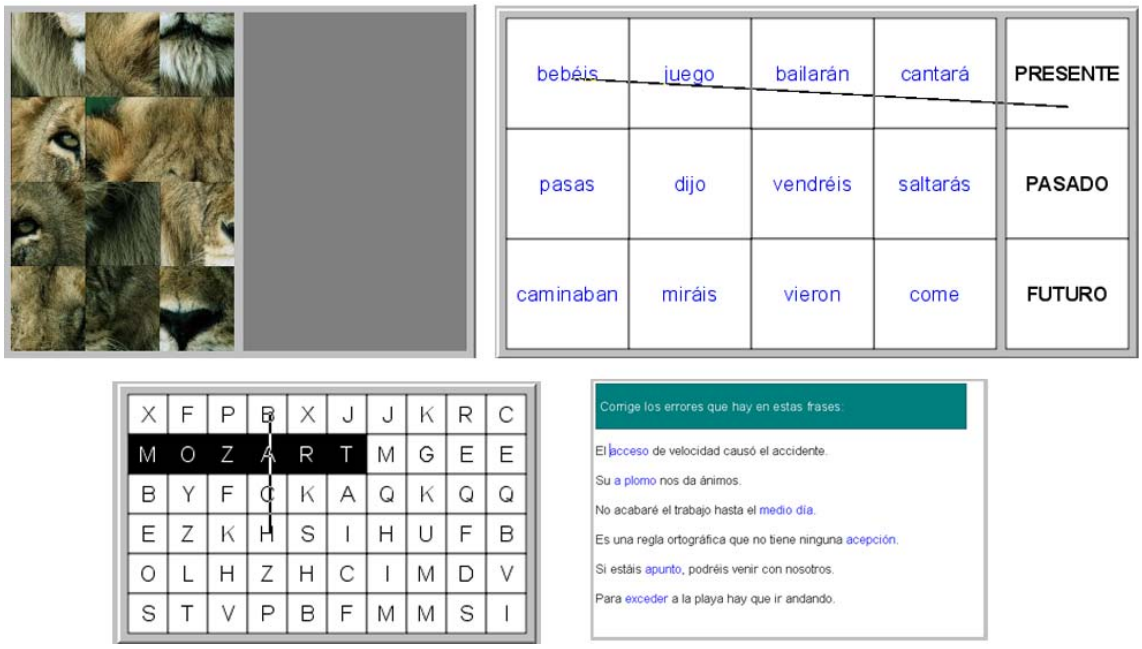

Figura 2.9 Actividades de JClic.

En la figura 2.9 se puede observar algunos ejemplos de la variedad de actividades que ofrece JClic, de izquierda a derecha y de arriba abajo se tiene un ejemplo de puzzle, otro de asociación, otro de sopa de letras y por último un ejercicio de texto.

2.5.2 Hot Potatoes, ¿Qué es y como funciona?

Hot Potatoes [http://hotpot.uvic.ca/] es un conjunto de seis herramientas para elaborar contenidos digitales. Las utilidades que posee esta herramienta permiten elaborar ejercicios interactivos de tipo página Web y de diferentes modalidades. Los ejercicios generados son páginas Web estándar que utilizan código XHTML 1.1 para la visualización, y JavaScript (ECMAScript) para la interactividad, los cuales son estándares W3C[2]. Estas herramientas funcionan también con Unicode[3], lo cual hace que se puedan crear ejercicios en cualquier idioma, o en una mezcla de idiomas.

Hot Potatoes es una herramienta que cuenta con una parte para editar las actividades y otra para realizarlas, pero ambas partes se encuentran en el mismo

[2] *W3C*: Consorcio World Wide Web, es un consorcio internacional donde las organizaciones miembro trabajan conjuntamente para desarrollar estándares web.

[3] *Unicode* es un estándar industrial cuyo objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático.

ejecutable. Por lo tanto con esta herramienta se pueden crear actividades y también realizarlas, una vez que se ha instalado la herramienta, se puede ejecutar como una aplicación normal.

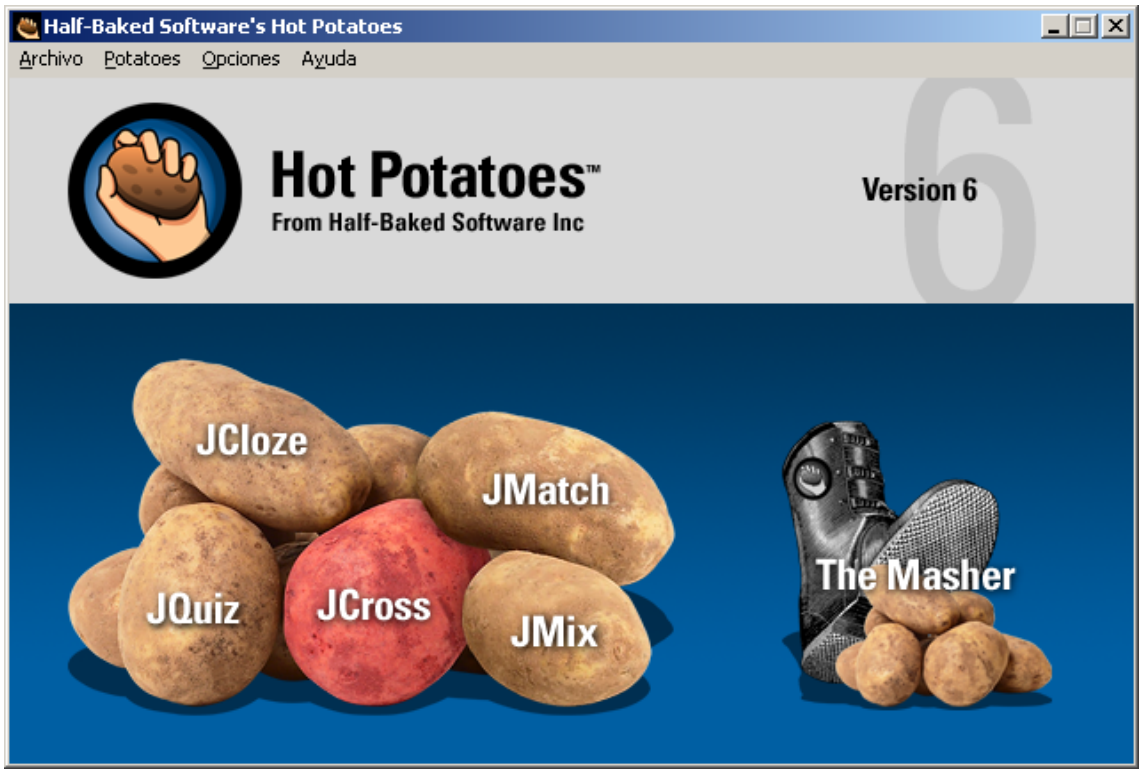

Figura 2.10 Pantalla inicial de Hot Potatoes.

En la pantalla que se muestra en la figura 2.10. se tiene el menú de entrada de la aplicación, en ella están a la izquierda los cinco botones que nos llevan a los diferentes editores de actividades que posee Hot Potatoes y un sexto botón “The Masher” que es el que nos permite acceder a las actividades para poder resolverlas. A continuación en la figura 2.11, se muestra una captura de las pantallas de los diferentes tipos de actividades.

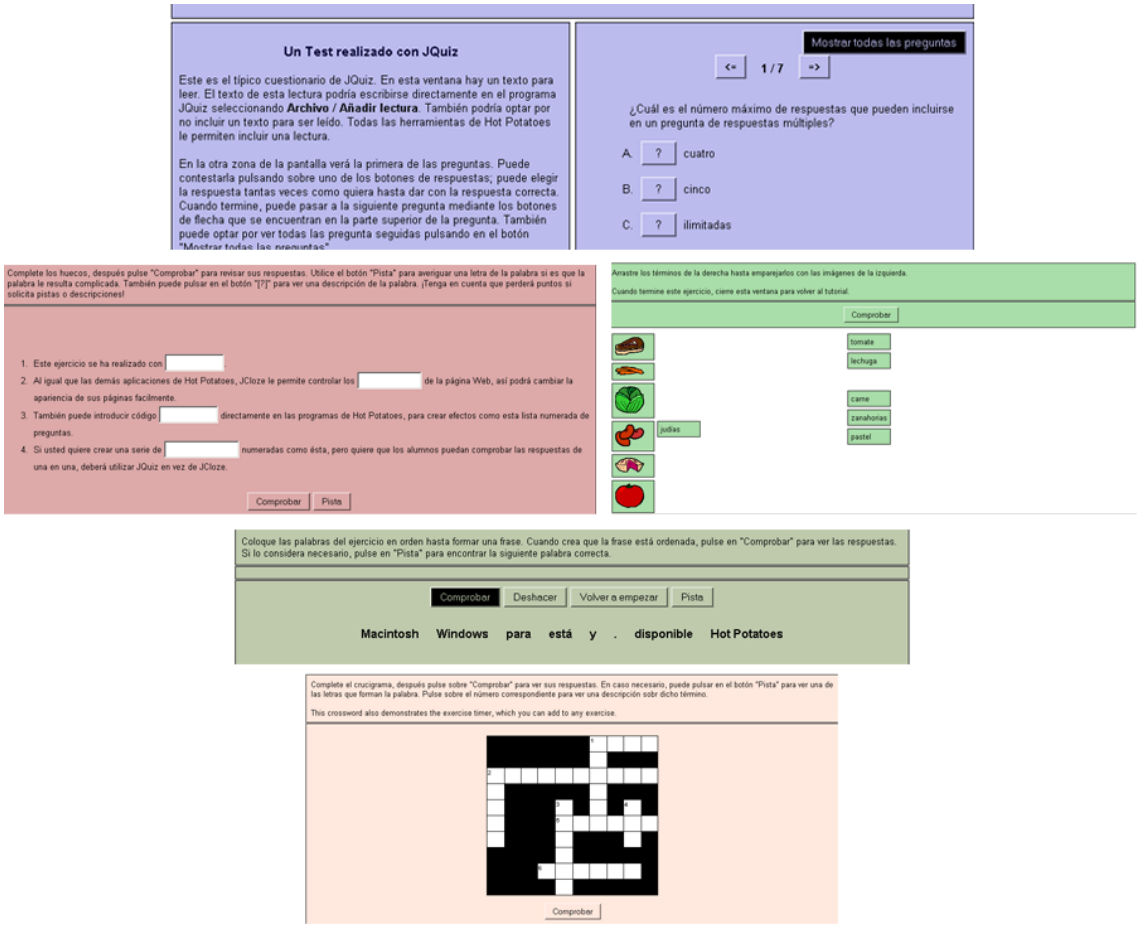

Figura 2.11 Actividades de Hot Potatoes.

En la figura 2.11 se puede ver una captura de cada uno de los tipos de actividades que ofrece Hot Potatoes, ordenadas de arriba abajo y de izquierda a derecha de la siguiente forma:

- JQuiz, que son ejercicios basados en preguntas.
- JCloze, que son ejercicios de rellenar huecos.
- JMatch, que son ejercicios tipo asociación.
- Jmix, que son ejercicios de ordenar tanto palabras como frases.
- Jcross, que son ejercicios de crucigramas.

2.5.3 Vitual Prismaker, ¿Qué es y como funciona?

Virtual Prismaker [http://www.spaincenter.org/b2b/toys/prismaker/virtual.htm] surge es la versión virtual de Prismaker [http://www.prismaker.com/], que es un juego de construcción que dispone de un número limitado de tipos de piezas, con la

posibilidad de asociarles un logotipo con el que dotarlas de significado y, por tanto, ampliar las posibilidades educativas del juego.

Un objetivo que tiene Virtual Prismaker es evaluar el juego Prismaker desde sus dos vertientes, la del juego físico, y la del juego virtual. Mediante Virtual Prismaker los profesores y los diseñadores gráficos y de software podrán actuar sobre la zona de desarrollo próximo del niño. De forma que en esa interacción social el niño sea capaz de realizar con ayuda del programa lo que no sería capaz de resolver por sí solo.

Esta versión virtual proporciona al niño un entorno de trabajo lo más parecido posible al que se encuentra en la vida real, creando un interfaz basado en metáforas. Con ello se pretende conseguir un modo más efectivo y eficiente de comunicarse con comunidades de usuarios diversas. [López et al.]

A continuación en la figura 2.12 se muestran un par de elementos de la interfaz de usuario, como son a la izquierda una escena de la sala de juegos y a la derecha el libro de actividades.

Figura 2.12 Elementos de la interfaz de usuario de Virtual Prismaker.

A continuación en la figura 2.13 se muestran unas capturas de algunas pantallas de Virtual Prismaker.

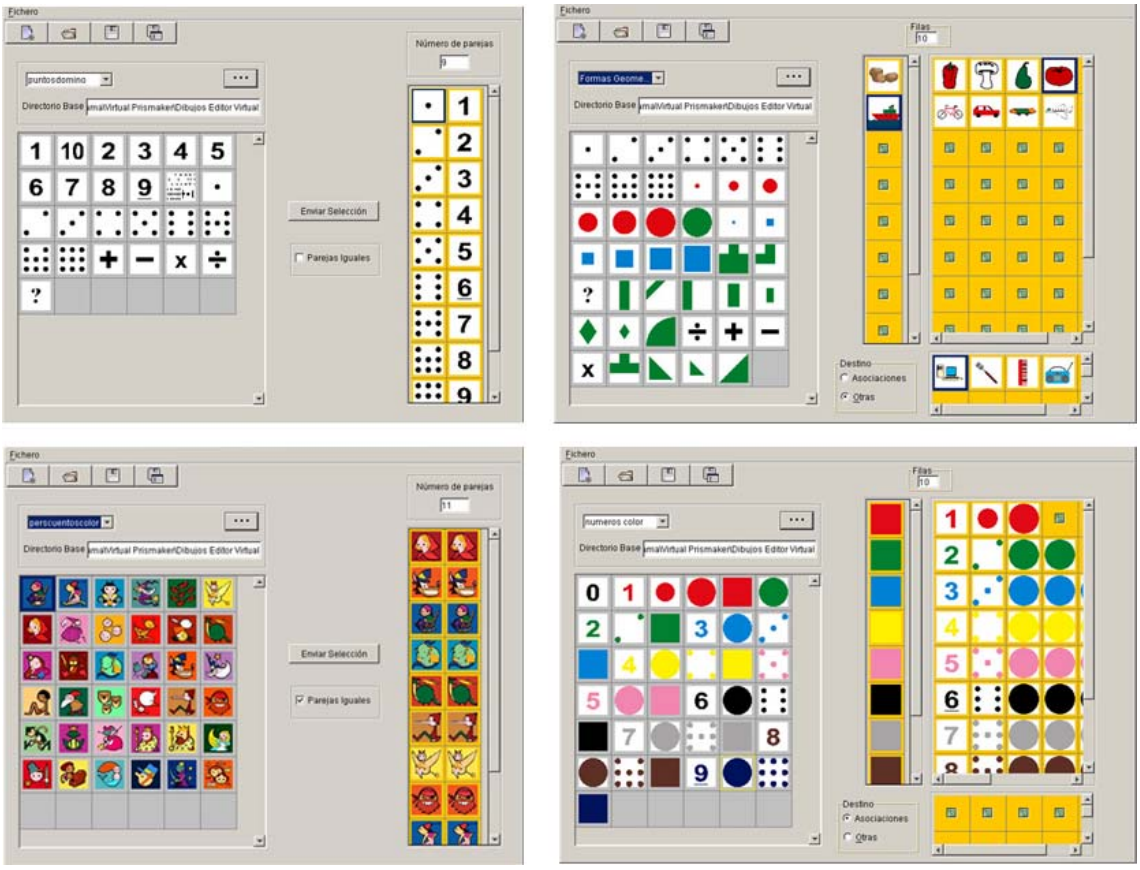

Figura 2.13 Capturas de pantallas de Virtual Prismaker.

2.5.4 Otras herramientas de juego

Además de las distintas herramienta que se han mostrado en los apartados anteriores, actualmente se tienen otras formas de aprender utilizando para ello la técnica del juego y la informática, este es el caso de por ejemplo el portal "*Portal de educación*" de la Junta de Castilla y León [http://www.educa.jcyl.es/], donde se tienen diversos juegos educativos adaptados a las diferentes edades de los usuarios, y cuya finalidad al igual que el resto de herramientas presentadas en este punto es el aprendizaje del niño por medio del juego llevándolo a cabo utilizando la informática.

A continuación, en la figura 2.14, se muestran unas pantallas de dicho portal.

Figura 2.14 Capturas del portal de educación Zona Infantil.

Además de este portal, del cual se muestran algunas pantallas en la figura 2.14, también hay en Internet otros portales del mismo tipo que intentan ayudar a aprender al niño por medio del juego.

2.6 Conclusiones

En resumen, lo que se puede deducir de los capítulos 1 y 2 es que lo que se pretende realizar es un producto software que cumpla las siguientes características:

- Que sea un producto en el que tengan cabida conocimiento procedente de la Ingeniería del Software y de la IPO (Interacción – Persona Ordenador), se desarrollará un producto software utilizando una metodología de clara inspiración en Ingeniería del Software, pero donde se desea considerar especialmente la consecución de determinadas características de calidad para cuya consecución se puede recurrir a propuestas provenientes de la disciplina de Interacción Persona-Ordenador.

- Así mismo se pretende llevar a cabo la realización de un producto que permita unas determinadas facilidades relacionadas con la evaluación de la interacción y la identificación de necesidades “educativas” adicionales.
- Qué se adapte a las características que según el currículo infantil del Ministerio de Educación y Ciencia, debe cumplir una herramienta software cuya finalidad es que sea utilizada para el aprendizaje en unas edades muy tempranas.
- También se pretende poner especial consideración en la usabilidad del producto software generado, integrando experiencia relacionada con el desarrollo de IU (interfaces de usuario) para niños. En este sentido, en el capitulo 3 se recogerá qué se entiende por usabilidad en base a un conjunto de criterios ergonómicos.

CAPÍTULO 3 USABILIDAD Y CRITERIOS ERGONÓMICOS

Este capítulo está dedicado a la usabilidad y a los criterios ergonómicos, tal y como su nombre indica, su finalidad es describir estos dos aspectos e indicar la importancia que tiene tenerlos en cuenta a la hora de desarrollar un software, así como la importancia que tiene que se tengan en cuenta desde las primeras etapas del desarrollo del software; también se indicará a lo largo del capítulo como se han integrado en el desarrollo de la aplicación que nos ocupa.

3.1 Definición de usabilidad

Sobre el término **usabilidad** nos encontramos varias definiciones que nos proporciona la *Organización Internacional de Estándares* (ISO), dichas definiciones son las siguientes:

Según la norma 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*". Y según la norma ISO/IEC 9241: "*Usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico*".

En la primera de las definiciones (ISO/IEC 9126) se pone más hincapié en las características internas y externas del producto, que contribuyen a su funcionalidad, usabilidad y eficiencia; mientras que en la segunda de las definiciones (ISO/IEC 9241) está más centrada en cómo el usuario realiza unas tareas específicas en unos escenarios

específicos con efectividad, eficiencia y satisfacción centrándose más, por tanto, en la calidad en el uso.

De las dos definiciones anteriores, tomándolas en conjunto, podemos concluir que la usablilidad es por tanto aquello que nos ayuda a utilizar, en nuestro caso, un software de la mejor manera posible para nosotros y de aquella forma que nos ayude a sacarle todo el partido posible; teniendo en cuenta que se potencia el uso de un software para una características y condiciones específicas, es decir, podemos hacer un software "usable" para un grupo específico de usuarios, pero esto no va a hacer que este software sea usable para todo el mundo, sino para aquellos individuos para los que haya sido pensado.

3.2 Atributos de la usabilidad

La usabilidad es una cualidad demasiado abstracta como para ser medida directamente. Para poder estudiarla se descompone en los siguientes cinco atributos básicos [Nielsen, 1993]:

- **Facilidad de Aprendizaje:** Cuán fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente la tarea que desea realizar el usuario. Se mide normalmente por el tiempo empleado con el sistema hasta ser capaz de realizar ciertas tareas en menos de un tiempo dado (el tiempo empleado habitualmente por los usuarios expertos). Este atributo es muy importante para usuarios noveles.
- **Eficiencia:** El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario. Cuanto mayor es la usabilidad de un sistema, más rápido es el usuario al utilizarlo, y el trabajo se realiza con mayor rapidez.
- **Recuerdo en el tiempo:** Para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero cada vez. Este atributo refleja el recuerdo acerca de cómo funciona el sistema que mantiene el usuario, cuando vuelve a utilizarlo tras un periodo de no-utilización.
- **Tasa de errores:** Este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza

una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja. Los errores reducen la eficiencia y satisfacción del usuario, y pueden verse como un fracaso en la transmisión al usuario del modo de hacer las cosas con el sistema.

- **Satisfacción:** Éste es el atributo más subjetivo. Muestra la impresión subjetiva que el usuario obtiene del sistema.

En el desarrollo de eAula, se han tenido en cuenta estos atributos para hacer que la aplicación sea usable; se ha intentado que tenga una gran facilidad de aprendizaje intentando hacer la interfaz lo más intuitiva posible, tanto en el caso del alumno donde se han utilizado botones con imágenes que el alumno pueda asociar con la función que realiza, como por ejemplo, un altavoz en lo referente a sonidos o una flecha para indicar siguiente; como en el caso del profesor donde se han indicado los pasos a realizar para hacer funcionar la aplicación.

Del mismo modo, se ha intentado que para realizar una determinada acción se lleve a cabo el menor número de pasos posible para hacer eficiente la aplicación; así mismo se ha intentado que sea fácil de recordar su funcionamiento haciéndolo sencillo, tal y como se ha indicado antes.

Teniendo en cuenta estos atributos, se ha intentado por tanto conseguir una pequeña tasa de errores y una gran satisfacción por parte de los usuarios.

3.3 Razones para tener en cuenta la usabilidad

La principal razón por la cual se debe tener en cuenta la usabilidad cuando se desarrolla un sistema software, es la obtención de un sistema que haga que el usuario sea más productivo, aumentando su eficiencia y satisfacción al utilizarlo.

La usabilidad es muy importante para que un sistema sea aceptado finalmente por sus usuarios, ya que este debe ser percibido como una herramienta que va a ayudar al usuario a realizar sus tareas y no a complicárselas, ya que de esto depende la aceptación del mismo. Puede llegar a ocurrir que el sistema no llegue a usarse en absoluto, o que se use muy de vez en cuando. Si las tareas de usuario no son respaldadas convenientemente por el sistema, entonces no se está respondiendo adecuadamente a las necesidades del usuario, y el equipo de desarrollo se está, por tanto, alejando del objetivo principal de la construcción de un sistema software.

También es importante invertir en la usabilidad del sistema que se construye de cara a la organización del desarrollo software; ya que puede ocurrir que se detecten errores graves de usabilidad en un momento cercano al final estimado del proyecto, y entonces el tiempo de desarrollo puede crecer demasiado para corregir tales problemas. Por otro lado, se deben considerar los elevados costes de servicio de atención al usuario que se pueden derivar de un sistema que tenga grandes deficiencias de usabilidad. Se debe tener también en cuenta que en un mercado altamente competitivo como es el de los productos software, el lanzamiento de un sistema con un alto nivel de usabilidad por parte de un competidor puede ser muy peligroso, por lo que es recomendable invertir en usabilidad, además de por las razones que se han dado anteriormente, para poder mantener o ampliar la cuota de mercado que se posee.

3.4 Como lograr la usabilidad

Para conseguir un producto con una gran usabilidad se debe seguir un ciclo de vida iterativo (un ciclo de vida iterativo es la iteración de varios *ciclos de vida en cascada*[4], al final de cada iteración se obtiene una versión mejorada o con más funcionalidad del producto). En una fase previa se establecen unas especificaciones de usabilidad que van a servir de patrón con el que comparar el nivel de usabilidad del sistema. A continuación comienza un ciclo diseño-evaluación-rediseño que finaliza cuando se alcanzan los niveles detallados en las especificaciones de usabilidad. [Ferré]

Las técnicas que se exponen a continuación van a agruparse según su uso en dicho ciclo:

- **Especificaciones:** Análisis de usuarios, análisis de tareas y especificaciones de usabilidad.
- **Diseño:** Diseño de la interacción, prototipado y participación de usuarios.
- **Evaluación:** Test de usabilidad y evaluación heurística.

A continuación se van a definir cada una de las técnicas, así como el procedimiento que siguen y cuando corresponda, las técnicas relacionadas.

[4] *Ciclo de vida en cascada*: Fue propuesto por Winston Royce en 1970, es un ciclo de vida que admite iteraciones; después de cada etapa se realiza una o varias revisiones para comprobar si se puede pasar a la siguiente.

3.4.1 Especificaciones

Al principio del proyecto se elaboran unas especificaciones de usabilidad intentando que realmente reflejen el nivel de usabilidad del sistema en los aspectos específicos que más interesen. Estas especificaciones dirigirán el proceso iterativo de desarrollo, pero para crearlas será necesario identificar previamente a los usuarios y las tareas que van a desarrollar con el sistema. Esta parte está muy relacionada con la Ingeniería de Requisitos[5].

3.4.1.1 Análisis de usuarios: Si se desea construir un sistema software usable, se debe primero conocer a fondo a qué usuarios específicos está destinado, cuáles son sus características principales [Hix et al., 1993].

Motivación: Para conocer a los usuarios, las tareas que desarrollan y cómo las llevan a cabo. Es importante conocer cómo piensa el usuario para desarrollar un sistema que trabaja según ese esquema.

Procedimiento: Depende del sistema específico a desarrollar. Algunos procedimientos que pueden llevarse a cabo son los siguientes:

- **Análisis de mercado**: Adecuado para el software comercial.
- **Visitas de campo**: Cuando se desarrolla software para una empresa u organización es muy útil observar al usuario en su entorno de trabajo habitual, más si cabe en el caso de que el usuario esté utilizando un sistema que será reemplazado por el que se va a construir.
- **Cuestionarios**: Sin ser tan útil como hablar a los usuarios en su entorno habitual, la información acerca de los usuarios puede ser recogida mediante cuestionarios.

Técnicas relacionadas: El análisis de usuarios puede proporcionar una categorización de usuarios, la cual puede ser útil a la hora de obtener una muestra de usuarios con los que realizar test de usabilidad.

[5] Ingeniería de Requisitos: Todas las actividades relacionadas con la identificación y documentación de las necesidades de clientes y usuarios.

3.4.1.2 Análisis de tareas: El término *Análisis de Tareas* se usa para describir un conjunto de técnicas que se preocupan de cómo hace la gente para realizar una determinada tarea [Preece et al.,1994]. Una tarea es una actividad con sentido para el usuario, algo que el usuario considera necesario o deseable que se realice.

Motivación: Descomponer la interacción con el sistema en unidades con sentido para el usuario. Estas unidades serán el punto de partida a la hora de desarrollar el sistema.

Procedimiento: Si en la actualidad la población de usuarios ya desarrolla una serie de tareas, se realiza un análisis de tareas actuales durante el análisis de usuarios, para identificar dichas tareas y el modo en que el usuario las percibe. Tras este primer análisis opcional, se identifican las tareas que el sistema a desarrollar va a ofrecer, en base a los objetivos que el usuario quiere satisfacer. Las tareas son descompuestas entonces en subtareas y éstas en acciones que el usuario realizará en su interacción con el sistema.

Técnicas relacionadas: El análisis de usuarios se toma como entrada al análisis de tareas, y ambos se realizan conjuntamente en determinadas ocasiones. El conjunto de tareas que se obtiene sirve como base para los participantes en los test de usabilidad[6].

3.4.1.3 Especificación de Usabilidad: Se establecen especificaciones de usabilidad como objetivos cuantitativos de usabilidad [Whiteside et al.,1988], los cuales se definen antes de comenzar con el diseño del sistema. Se basan en los cinco atributos de usabilidad básicos descritos anteriormente, o en subatributos de los mismos.

Motivación: Si se quiere poder medir la usabilidad del sistema que se está construyendo, es preciso tener un conjunto de especificaciones de usabilidad que puedan ser verificadas.

Procedimiento: Se define un conjunto de especificaciones de usabilidad para cada atributo de usabilidad que se quiera medir (que sea importante para aquel sistema que se quiera construir). Deben definirse de modo que puedan medirse, bien mediante test de usabilidad, bien mediante cuestionarios.

---

[6] Test de usabilidad: en el apartado 3.4.3.1. de esta memoria se define que son y en que consisten los test de usabilidad.

Técnicas relacionadas: La mayor parte de las especificaciones de usabilidad están asociadas a una tarea específica de las obtenidas en el análisis de tareas. Los test de usabilidad aportan los valores de las especificaciones para la versión en curso del sistema que se está desarrollando.

3.4.2 Diseño

Una vez identificadas las tareas a las que el sistema va a dar soporte, se puede empezar a diseñar la interacción del sistema, como una primera aproximación que será evaluada y, eventualmente, mejorada en posteriores iteraciones.

3.4.2.1 Diseño de la Interacción: El diseño de la interacción se puede dividir en dos etapas: Diseño del concepto del sistema y diseño de la parte visual de la interacción.

Diseño del concepto del sistema: Es la actividad más importante del desarrollo de software, pues definirá de qué modo va a funcionar el sistema. Es importante crear un concepto del sistema que pueda ser comprendido y asimilado fácilmente por el usuario. Para ello se pueden usar metáforas (como por ejemplo la metáfora del escritorio usada por algunos sistemas operativos) basadas en conceptos familiares para el usuario, o bien imitar sistemas ya conocidos por el usuario.

Diseño de la parte visual de la interacción: Hay una serie de normas pertenecientes al campo del diseño gráfico sobre cómo escoger colores, tipos de letra, la disposición de los elementos en una ventana, etc. Esta parte suele realizarla un diseñador gráfico profesional.

3.4.2.2 Prototipado: El prototipado es una técnica muy valiosa en las primeras fases del desarrollo para representar el diseño de la interacción y evaluar su usabilidad.

Motivación: No es posible conocer la opinión de los usuarios mostrándoles especificaciones técnicas a un nivel abstracto. Los usuarios entenderán mucho mejor prototipos concretos del sistema.

Procedimiento: Algunas técnicas de prototipado ayudan a representar la interacción con un esfuerzo mínimo de implementación:

- *Borradores en papel:* Al principio del proceso de diseño se pueden crear prototipos sobre papel para mostrarlos al usuario. Normalmente son

representaciones de las ventanas de la aplicación. El diseñador actúa como sistema, presentando al usuario el siguiente elemento cuando ocurre una transición entre ventanas [Nielsen, 1993].

- *Técnica del Mago de Oz [Preece et al.,1994]:* Un experto humano actúa como sistema, dando las respuestas a las peticiones del usuario. El usuario interactúa normalmente con el terminal, pero en vez de haber un software detrás, un desarrollador está frente a otro terminal conectado al primero a través de la red, y va tecleando la supuesta respuesta del sistema. El usuario normalmente no sabe esto, para conseguir así la sensación de que está tratando con un sistema real.
- *Escenarios, storyboards y viñetas:* Un escenario describe una historia de ficción de un usuario interactuando con el sistema en una situación concreta. Las viñetas son representaciones que capturan la interacción que ocurre en un escenario. *Storyboards* son secuencias de viñetas que se centran en las principales acciones en una situación dada [Preece et al.,1994]. Estas técnicas posibilitan que el equipo de diseño tenga en cuenta lo apropiado o no que es el diseño actual para las necesidades del usuario, favoreciendo de esta manera un proceso de diseño más centrado en el usuario.

3.4.2.3 Participación de los Usuarios: Para que el sistema sea realmente como quieren y/o necesitan los usuarios se puede involucrar a éstos para que participen en el proceso de diseño. Esta filosofía de diseño se conoce como *Diseño Centrado en el Usuario*. Por otro lado está el *Diseño Participativo* que va un paso más allá y pone a representantes de usuarios como responsables de decisiones de diseño.

- *Diseño Centrado en el Usuario:* Es la aplicación al diseño de la filosofía en la que se basa la Ingeniería de Usabilidad. Se trata de centrar el proceso de diseño en el usuario, implicando a los usuarios en el diseño para confirmar que se está desarrollando un sistema que en realidad satisface sus necesidades. La interacción debe ser diseñada desde el punto de vista del usuario [Hix et al., 1993].

Con esto lo que se pretende es que el sistema que se está desarrollando sea fácil de entender, aprender y de utilizar para su usuario final y que al mismo tiempo realice las tareas para la que ha sido desarrollado de una manera correcta. La participación del usuario, es por tanto un punto clave para el desarrollo del sistema y para tener su punto de vista se le debe involucrar en el proceso de desarrollo. Esta participación de los usuarios se lleva a cabo mediante la creación de diversos prototipos de la aplicación a lo largo del proceso de desarrollo.

- *Diseño Participativo:* También es conocido como *Diseño escandinavo*, debido a que surgió en los países escandinavos. Es una aproximación al diseño que pertenece no sólo al diseño de sistemas software, sino también al diseño industrial. Este enfoque acepta la idea de que el usuario es el máximo experto en diseñar un sistema que satisfaga sus necesidades. Los usuarios son los diseñadores del sistema, actuando los ingenieros software como consejeros técnicos que indican qué se puede y qué no se puede hacer [Schuler et al.,1993]. El papel de los usuarios no va a ser el de un elemento pasivo al que se consulta sobre temas específicos, sino como núcleo central del equipo de desarrollo.

Para llevar a cabo esta técnica se debe tener en cuenta que cuando los usuarios forman parte del equipo de desarrollo se deben utilizar medios y

lenguajes no técnicos, es decir, que sean fácilmente entendibles para los usuarios, por ejemplo diseñar la interfaz con papel y lápiz.

3.4.3 Evaluación

La evaluación de usabilidad permite conocer el nivel de usabilidad que alcanza el prototipo actual del sistema, e identificar los fallos de usabilidad existentes.

La evaluación se realiza usualmente mediante test de usabilidad, complementados con evaluaciones heurísticas.

3.4.3.1 Test de Usabilidad: Consisten en presentar al usuario una serie de tareas a realizar, y pedirle que las realice con el prototipo del sistema. Las acciones y comentarios del usuario se guardarán para un análisis posterior.

Para conseguir unos test de usabilidad con resultados fiables, las condiciones del test y de lugar donde éste se realiza deben ser lo más parecidas posibles al entorno de uso previsto para el sistema.

Motivación: Se basa en que no es posible conocer el nivel de usabilidad de un sistema si no se prueba con usuarios reales.

Procedimiento: Es preciso decidir con qué distintos grupos de usuarios se va a probar el sistema, y cuántos participantes de cada grupo se van a obtener para realizar los test. A continuación, se deben diseñar las tareas de test cuya realización se va a pedir a los participantes; normalmente se sacan del resultado del análisis de tareas, intentando enmarcarlas en un contexto de uso real. Hay que decidir también la posibilidad de pedir ayuda al evaluador, qué tipo de información se dará a los participantes acerca del sistema antes de comenzar con el test en sí, o si se dará la posibilidad al participante de acceder libremente al sistema para observar los comentarios que intercambian.

Una vez que se han realizado todos los test, los resultados obtenidos serán analizados y se aplicarán en el siguiente ciclo de diseño.

3.4.3.2 Evaluación Heurística: Este tipo de evaluación la realiza un experto en usabilidad (o en HCI[7]), dicho experto realizará una crítica basada en su experiencia de diseño de la interacción, o en guías de diseño de usabilidad ampliamente aceptadas, como las descritas por Shneiderman [Sheiderman, 1998] o Nielsen [Nielsen, 1993].

Los expertos proporcionan una información distinta a la obtenida de usuarios finales mediante test de usabilidad. Las sugerencias de modificaciones por parte de un experto suelen tener más valor que las realizadas por usuarios, por ser más viables y más precisas acerca de los problemas subyacentes de usabilidad, como puede ser la falta de consistencia o una navegabilidad pobre. Por otra parte, para identificar problemas de usabilidad específicos es preciso realizar test de usabilidad con usuarios reales.

Por tanto, la evaluación heurística no se debe utilizar para sustituir los test de usabilidad sino que debe usarse como un complemento a ellos.

Motivación: Los expertos identifican ciertos problemas de usabilidad que requerirían un gran número de test de usabilidad para ser correctamente identificados y solucionados.

Procedimiento: Se explica al experto el modo de funcionamiento del sistema, y se le entrena en las funciones principales. Se le informa acerca del dominio de la aplicación y el experto revisa el sistema según la conformidad del mismo a guías de diseño. Una vez finalizada la revisión el experto elabora un informe con los problemas identificados, también puede aportar sugerencias de diseños alternativos, esto será entregado al equipo de desarrollo.

3.5 Logro de la usabilidad mediante modelos de calidad

La elaboración de un modelo de calidad pasa por la identificación de aquellas características, tanto internas como externas, que influyen en la calidad de un producto software. Un modelo de calidad es una descomposición jerárquica, y su elaboración puede abordarse de diferentes formas, como puede ser la identificación de objetivos y métricas que permitirían la caracterización de la calidad.

En Ingeniería del Software (disciplina que se tratará en el capitulo 4 de esta memoria) se han elaborado diferentes modelos de calidad y estándares relacionados con

[7] HCI: *Human-Computer Interaction*, en español IPO (Interacción Persona-Ordenador)

la calidad que ofrece un producto software. Entre estos modelos se pueden identificar dos tendencias: por un lado están los que persiguen identificar características del producto (McCall, Boehm...) y, por otro, los que tratan de identificar características de calidad en el proceso de desarrollo del producto software (Cocomo, CMM,...).

A continuación se definirán algunos de estos modelos:

3.5.1 Modelos de calidad orientados al producto

- **McCall:** En este modelo de calidad se identifican 11 factores de calidad orientados al producto y agrupados entorno a tres factores: *operación, revisión y reutilización*. (Figura 3.1.).

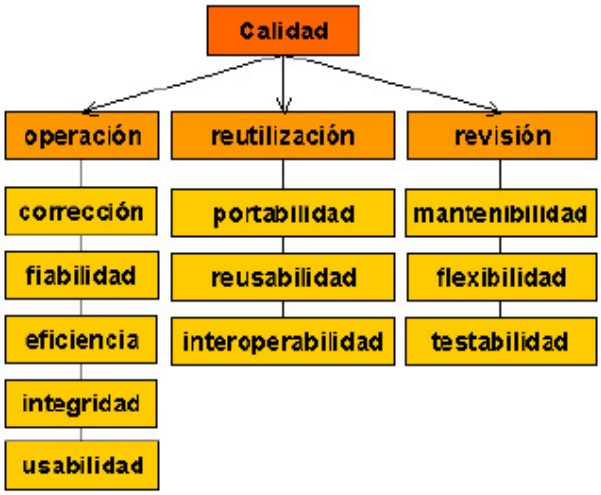

Figura 3.1 Modelo de calidad de McCall.

La gran aportación de este modelo fue el establecimiento de relaciones entre características de calidad y métricas, aunque haya sido criticado por la objetividad de algunas de las métricas propuestas. [McCall et al., 1977]

- **Bohem:** El modelo de Bohem es similar al de McCall (figura 3.1) al representar la estructura de características de calidad de forma jerárquica, contribuyendo cada una de ellas a la calidad total. En este modelo también están consideradas las necesidades del usuario, pero también encontramos características de calidad en el terreno del hardware que no están presentes en el otro modelo propuesto. [Boehm et al., 1978]

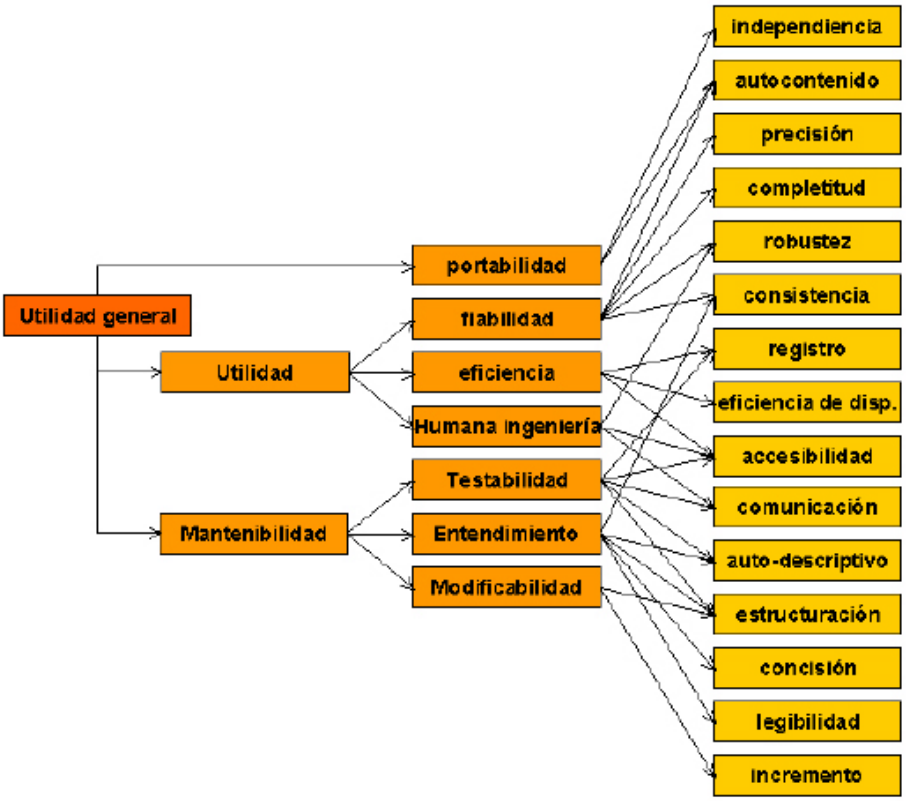

Figura 3.2 Modelo de calidad de Boehm.

- **FURPS:** Este modelo establece cinco características como factores de calidad que son los que le dan nombre: *Functionality, Usability, Reliability, Performance y Supportability.* (figura 3.3.).

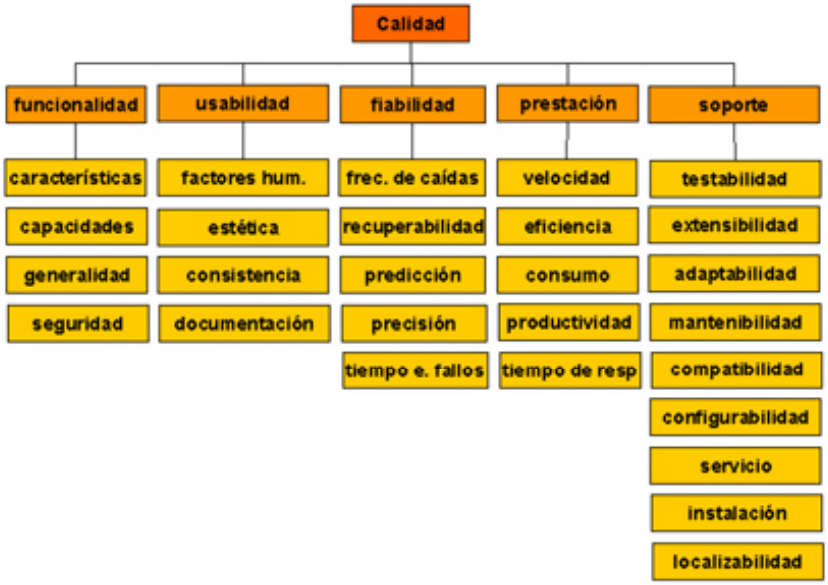

Figura 3.3 Modelo de calidad FURPS.

Una limitación de este modelo de calidad es que no tiene en cuenta la portabilidad de los productos software que se estén considerando, factor digno

de consideración en función de las exigencias actuales que recaen sobre el proceso de desarrollo del software [Grady, 1987].

- **ISO 9126:** Este estándar define la calidad de un producto software como un conjunto de características dependientes del producto y cuenta con gran aceptación y extensión, pese a estar imitado al describir la calidad a nivel de factores y, únicamente, introducir una serie de criterios que también pueden considerarse al tratar cada uno de esos factores. (figura 3.4.) [ISO,1991]

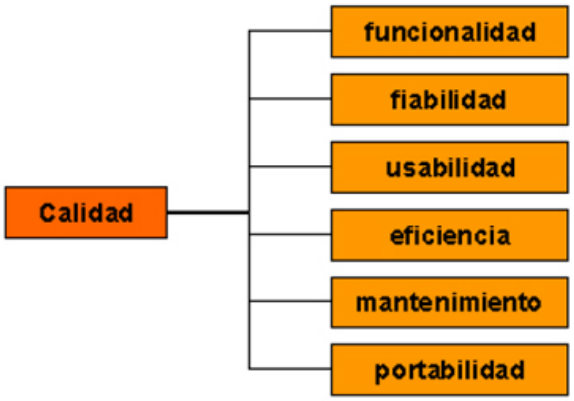

Figura 3.4 ISO/IEC 9126.

- **Dromey:** En este modelo se estudian las relaciones entre las características y las subcaracterísticas asociadas con la calidad de un producto software, concluyendo que existen diferentes características, ya sean éstas factores o criterios, que debido al nivel que ocupan en el modelo de calidad, no pueden ser consideradas en su totalidad dentro del software. De esta forma Dromey aboga por la elaboración de modelos de calidad abiertos o mixtos en los que partiendo de atributos identificados en otros estándares, se enriquecen con criterios propios del producto software concreto que se pretenda desarrollar. En la figura 3.5. aparece recogida parte de su propuesta para un modelo de calidad asociado a la fase de implementación, donde podemos ver las siguientes cualidades tangibles:
    - **Correctitud:** Pueden ser internas (asociadas con los componentes individuales) o contextuales (asociadas con la manera en que los componentes son utilizados en el contexto).
    - **Internas:** miden lo bien que un componente ha sido entregado de acuerdo a su objetivo, implementación o lo bien que ha sido compuesto.

o **Contextuales:** cómo los componentes son compuestos y las influencias que ejercen sobre la calidad del producto.

o **Descriptivas:** Para ser útil un software debe ser fácil de entender y utilizar de acuerdo a su propósito. Estas propiedades descriptivas aplican a requerimientos, diseños, implementación y a las IU.

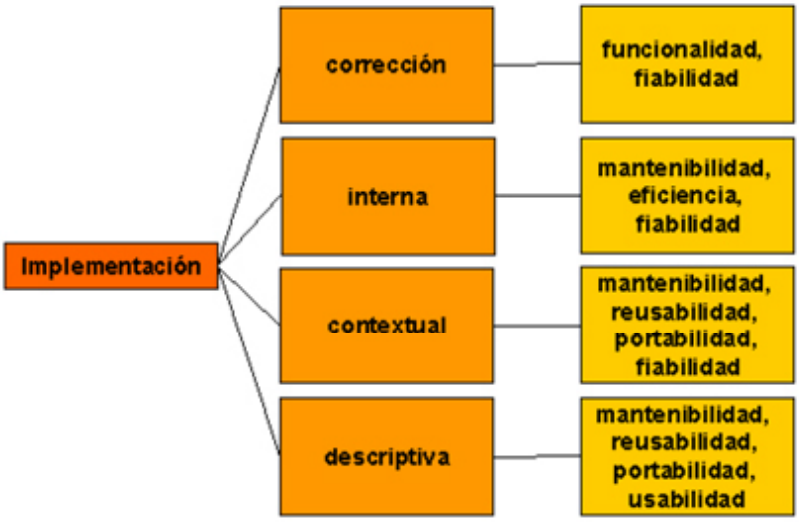

Figura 3.5 Modelo de calidad propuesto por Dromey.

3.5.2 Modelos de calidad orientados al proceso

- **CMM:** *(Capability Maturity Model)* Para Le Manh (Le Mahn, 1998), el CMM. provee a las organizaciones de software de una guía sobre cómo controlar el desarrollo y mantenimiento de sus procesos de software, y cómo evolucionar hacia una cultura de ingeniería de software y administración excelente. Fue diseñado para guiar a las organizaciones en la selección de estrategias de mejoramiento de los procesos, determinando la madurez del proceso actual e identificando los problemas más críticos para la calidad y el mejoramiento del proceso.

El **CMM** está estructurado en 5 niveles de madurez (figura 3.6.) que proporcionan las bases para el mejoramiento continuo del proceso. Estos niveles definen una escala ordinal para medir la madurez de un proceso[8] evaluar su capacidad[9]. [Mendoza]

[8] La "*madurez de un proceso de software*" es el grado para el cual un proceso específico está definido, manejado, medido, controlado y es efectivo.

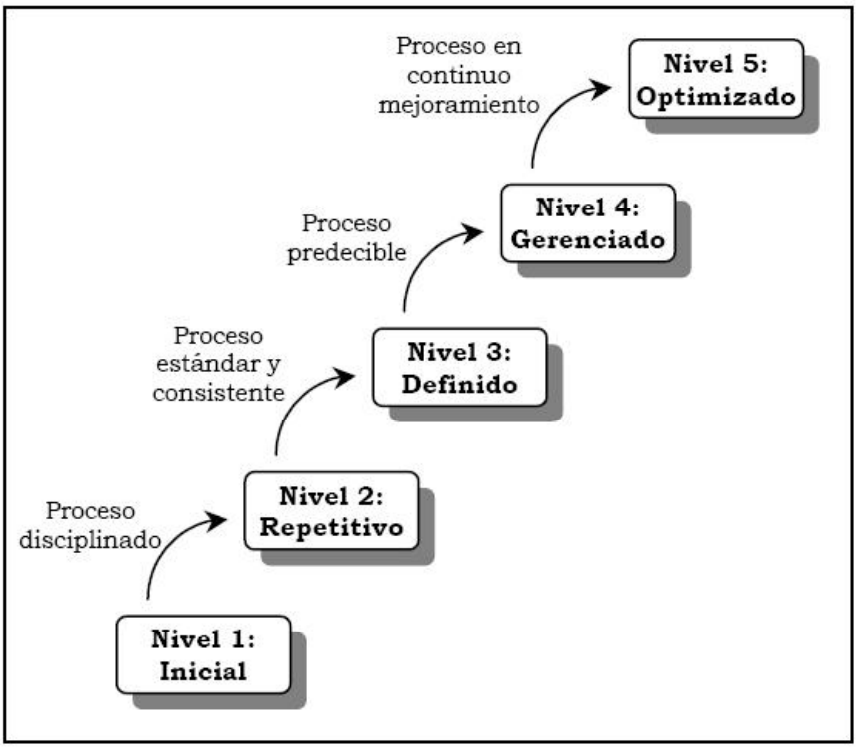

Figura 3.6 Niveles de madurez del modelo de calidad CMM.

- **SPICE:** *Software Process Improvement and Capacitability dEtermination* [El Eman et al., 1997]. Según la ISO/IEC (ISO/IEC, 1997), SPICE es un modelo para la evaluación de procesos de software que se encuentra dentro de los documento de la ISO y ha ido evolucionando hacia un proyecto de estándar ISO 15504. La arquitectura del modelo contiene dos jerarquías:

    - El lado izquierdo, en la figura 3.7, consiste en la categoría de procesos, compuestos por procesos y éstos están compuestos por prácticas bases.

    - Los procesos son evaluados en términos del lado derecho, figura 3.7. Los procesos pueden ser evaluados a un nivel de capacidad, los niveles de capacidad están compuestos por características comunes y estas a su vez están compuestas por prácticas genéricas.

[9] La "*capacidad del proceso de software*" describe el rango de resultados esperados que se pueden alcanzar siguiendo tal proceso.

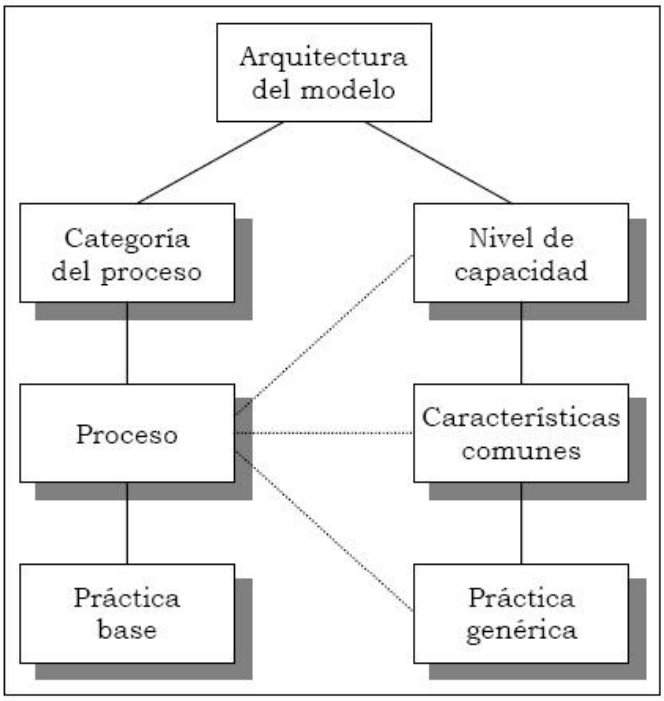

Figura 3.7 Modelo de calidad SPICE.

El Modelo SPICE fue ideado pensando en las particularidades que implica el desarrollo de software, es decir, fue diseñado especialmente para software.

- **ISO 9000:** La serie de normas ISO 9000 es un conjunto de documentos que pueden usarse para los propósitos de aseguramiento de la calidad de casi cualquier cosa. Esta norma especifica los requisitos de los sistemas de calidad para ser usados en un contrato entre dos partes que requieren la demostración de la capacidad de un proveedor para diseñar y suministrar un producto determinado.
    - ISO 9000: Normas para la gerencia y el aseguramiento de la calidad. Guía para la selección y uso.
    - ISO 9001: Sistemas de Calidad - Modelo para el aseguramiento de la calidad en el diseño, desarrollo, producción, instalación y servicio.
    - ISO 9002: Sistema de Calidad - Modelo para el aseguramiento de la calidad en la producción e instalación.
    - ISO 9003: Sistemas de Calidad - Modelo para el aseguramiento de la calidad en la inspección final y prueba.

- ISO 9004: Gerencia de la calidad y elementos del sistema de calidad - Pautas.
- ISO 9000-3: Guía para la aplicación de la ISO 9001 al desarrollo, suministro y mantenimiento del software.

Dentro de las ventajas que presenta, se puede mencionar que es bastante conocido por las organizaciones y sus clientes, y sirve de apoyo a los demás estándares.

La crítica más importante es que descuida la etapa de análisis, haciendo énfasis en el diseño y el desarrollo. No fue creado para el tratamiento del software específicamente, por cuanto es una adaptación de las ideas de los procesos de manufactura.

3.6 Criterios ergonómicos

La ergonomía es el conjunto de estudios, métodos y disposiciones para hacer el trabajo más humano en función de las capacidades fisiológicas y psicológicas del individuo. Algo tan simple como el empleo de fuentes gráficas o colores uniformes tiene un peso muy importante en la productividad que el usuario final es capaz de alcanzar con las aplicaciones

Para conseguir que un software sea usable se pueden tener en cuenta diversos “criterios ergonómicos” [Bastien et al., 1993], con los criterios ergonómicos lo que se pretende es tener en consideración determinados factores humanos a la hora de desarrollar interfaces de usuario, haciendo de este modo que dichas interfaces sean más fáciles de utilizar por aquellos usuarios a los que está destinada.

A continuación se indican una serie de criterios ergonómicos.

- **GUIDANCE:**

Con una buena guía se puede facilitar el uso de un sistema permitiendo al usuario que en poco tiempo conozca los pasos necesarios para llevar a cabo una tarea específica en un sistema dado con unas condiciones determinadas. Con esto, se consigue por tanto que haya una facilidad en el aprendizaje del usuario del sistema, lo que hará que se lleve a cabo un mejor manejo del mismo.

- **PROMPTING:**

Con esto nos referimos a mostrar en cada momento al usuario las diferentes opciones que tiene en un momento determinado, así como los pasos que debe seguir para realizar alguna función. Se ayuda por tanto al usuario mostrándole das distintas alternativas que posee. El que la interfaz muestre las cosas de una manera determinada hace que su uso sea más sencillo, ayudando en cada momento al usuario, por ejemplo, una manera de ayudar al usuario sería al darle varias opciones para llevar a cabo una determinada tarea, señalar de algún modo la más común, o la que se debería elegir normalmente en ese caso, de este modo, estamos ayudando al usuario de una manera sugestiva.

- **GROUPING/DISTINCTION OF ITEMS:**

Con este criterio nos referimos a la agrupación de la información de tal manera que si mostramos varias cosas relacionadas entre sí, estas se encuentren en el mismo sitio, con esto hacemos que cuando el usuario busque alguna cosa encuentre cerca de ella lo que esté directamente relacionado, y del mismo modo podrá distinguir mejor aquellas cosas que son diferentes ya que estarán colocadas en diferentes lugares.

Ayudamos por tanto al usuario a distinguir grupos, por ejemplo, imágenes por un lado, sonidos por otro; relación de los unos con los otros...

- **INMEDIATE FEEDBACK:**

Este criterio concierne a la respuesta inmediata del sistema ante una acción del usuario, lo que se pretende por tanto, es que cuando el usuario introduzca un dato u orden en el sistema, éste le responda de manera inmediata. De este modo, el usuario podrá asociar sus acciones con la reacciones correspondientes por parte del sistema y, en nuestro caso, será cuando el niño aprenda, ya que sabrá que cada vez que hace una determinada cosa se le responde de la misma forma.

- **LEGIBILITY:**

Con esto nos referimos a que la información mostrada en pantalla en todo momento tiene que ser legible y ayudar al usuario (al niño) a entender lo que se está haciendo en cada momento y no abrumarlo con demasiada información o con datos que en un momento determinado no le sirvan de ninguna ayuda.

- **WORKLOAD:**

Con esto nos referimos a todos aquellos elementos que juegan un rol en la percepción del usuario, es decir, todo aquello que se utiliza para que la aplicación “se comunique” con el usuario; este criterio se divide a su vez en dos: **Brevedad**, la capacidad del usuario es limitada por lo que no debemos abrumarle con demasiados datos y **Concisión**, representar los elementos de una manera clara, para que el usuario pueda recordarlos.

- **MINIMAL ACTIONS:**

Para realizar cada tarea de la aplicación se debe utilizar el número mínimo de pasos, de este modo hacemos que sea más fácil para el usuario recordar que tiene que hacer en cada momento, en el caso que nos ocupa, al tratarse de niños esto es todavía más importante ya que al ser además de corta edad (3-5 años), debemos tener en cuenta que el uso de la aplicación debe ser sencillo para que ellos lo recuerden y no se aburran antes de terminar una tarea determinada.

- **EXPLICIT USER ACTION:**

Con esto, nos referimos a la relación entre la aplicación y las acciones del usuario, de tal modo que cuando el usuario pulse un determinado botón o haga una determinada cosa se lleve a cabo siempre la misma acción de tal forma que el usuario sepa exactamente que conlleva cada una de sus acciones.

- **ADAPTABILITY:**

Con la adaptabilidad nos referimos a que la aplicación debe estar adaptada para el usuario a la que está dirigida; es decir, por ejemplo en nuestro caso, al tratarse de una aplicación para niños debe estar destinada específicamente a ellos, con una interfaz atractiva para los mismos y un manejo de la aplicación que sea fácil e intuitiva para los niños.

- **FLEXIBILITY:**

Con la flexibilidad lo que se pretende es que la aplicación tenga una interfaz que se pueda adaptar a las necesidades particulares del usuario; en nuestro caso esto se puede llevar a cabo dando a cada alumno una tarea en concreto, es decir hacer que en cada momento cada uno de los alumnos pueda estar realizando una tarea diferente dependiendo de las carencias de cada uno y de las necesidades que tenga el alumno en cada momento.

En la aplicación que nos ocupa, lo criterios que se han pretendido tener más en cuenta son los siguientes:

Calidad	Factor	Criterio	Ejemplos
Usability	**Understability**	**conpatibility**	Mediante la utilización de símbolos sencillos para que puedan ser identificados por los usuarios, y que cada símbolo signifique siempre lo mismo. Por ejemplo, en la aplicación del alumno cada vez que se utiliza un audio se tiene el icono del altavoz, tanto en el botón que nos da la opción de apagar y encender las locuciones, como en aquellas asociaciones en las que tengamos que unir sonidos; y en la aplicación del profesor se ha incluido un icono en cada uno de los botones para que el profesor asocie la imagen con la función que se realiza al pulsarlo.
		Legibility	Con una combinación de colores que posibilite que el usuario lleve a cabo las acciones deseadas. En la aplicación se ha intentado cumplir esto utilizando colores compatibles entre sí y que permitan una completa legibilidad.
		Prompting	Resaltando aquellas imágenes o botones asociados a acciones que el usuario puede o debe realizar en cada momento. Por ejemplo, en la aplicación del alumno, cuando terminamos tanto una asociación como un puzzle, se lleva a cabo una animación que nos descubre el botón que debemos pulsar, dicho botón no es visible hasta ese momento, es decir, hasta que debe ser pulsado.
		Immediate feed-back	Cualquier acción que el usuario lleve a cabo debe provocar una reacción inmediata y automática. En toda la aplicación se tiene que cada vez que el usuario pulsa alguno de sus elementos esto conlleva una reacción automática de la aplicación, por ejemplo si en la aplicación del alumno se pulsa el botón de salir, automáticamente se va a la pantalla de salida.

		significance of codes a behaviour	Las etiquetas que se utilicen en cada momento deben ser significativas para los usuarios a los que está dirigida la aplicación. En el aula virtual se han intentado incluir unos iconos que sean significativos para los usuarios, por ejemplo, en la aplicación del alumno cuando se va a la pantalla de salida, se tiene un niño que mueve la cabeza de arriba abajo en el botón NO.
		helpfulness	Proporcionar ayuda para que el usuario pueda realizar cada una de las tareas que se le proponen. Cada vez que en la aplicación del alumno se entra en una actividad además de aparecer un texto de ayuda se ejecuta una locución que indica al alumno que debe hacer para realizar dicha actividad y en la aplicación del profesor se tiene un botón de ayuda que al pulsarlo muestra una leyenda sobre los pasos que se deben realizar para enviar las actividades a los alumnos.
	Learnability	**grouping**	Agrupando en cada momento aquellas imágenes o botones relacionados entre sí. Por ejemplo todos los botones de navegación general se encuentra agrupados en la misma posición en la pantalla.
		minimal actions	El número de acciones será el mínimo posible en cada caso. El número de clicks que el usuario debe realizar para pasar de una acción a otra son mínimos, por ejemplo, en el caso de la aplicación del alumno, para apagar las locuciones solo tiene que pulsar un botón y para volver a encenderlas, pulsará el mismo botón.
		conciseness	Las explicaciones serán claras y concisas para que los usuarios a los que están dirigidas puedan entenderlo sin ningún problema. En el caso de la aplicación destinada a los alumnos, al iniciar una actividad se explica de manera clara lo que el alumno debe hacer para poder realizar dicha actividad.

		information density	La interfaz estará lo menos cargada posible para no abrumar a los usuarios con demasiados datos. La interfaz no posee información innecesaria, en todo momento el usuario solo tiene disponibles aquellos elementos que necesita para llevar a cabo la ejecución de la función correspondiente.
		consistency	Las tareas y las combinaciones de colores deben seguir cierta homogeneidad y coherencia. La disposición de los botones será siempre la misma en las diferentes actividades. Los botones que realizan la misma tarea en las distintas actividades, están situados en la misma posición.
	Operability	**explicit user action**	Las acciones que se pueden llevar a cabo estarán visibles para el usuario. El usuario tiene visibles en cada momento aquellos botones que puede pulsar.
		user control	Al usuario aunque se le guíe debe creer que tiene control sobre la aplicación. Esto se puede hacer destacando en cada momento aquel botón que es recomendable pulsar. En la aplicación del alumno, en el caso de las asociaciones, por ejemplo, el botón que la corrige aparece cuando se debe pulsar.
		flexibility	Haciendo que con la misma aplicación dos usuarios puedan realizar actividades diferentes en un mismo momento. Un alumno puede estar realizando una asociación, mientras otro alumno completa un puzzle y otro alumno esta esperando a recibir datos del profesor. Lo que cada alumno esté haciendo en cada momento puede ser totalmente independiente de lo que están haciendo sus compañeros de clase.

Tabla 3.1 Criterios ergonómicos tenidos en cuenta en las aplicaciones.

3.7 Aplicación de la usabilidad

En los puntos anteriores se han dado varias formas para conseguir la usabilidad, en el caso de seguir lo indicado en el punto 3.4. si un Ingeniero software quiere aplicar los conceptos y técnicas que se han descrito en dicho punto para hacer más usable el software que está desarrollando no debe intentar aplicarlo todo de una vez. Deberá tener en cuenta que la evaluación de la usabilidad se deberá llevar a cabo a lo largo de todo el proceso de desarrollo.

Del mismo modo en los otros puntos se han mostrado un conjunto de factores, criterios y modelos para aplicar al desarrollo pero no se tienen unas pautas fijas que nos indiquen exactamente como debe realizarse, faltaría por tanto integrar dichos elementos en el propio proceso de desarrollo. La aplicación de unos factores de usabilidad u otros dependerá tanto del punto de vista del desarrollador como de la experiencia del mismo.

Para aplicar usabilidad al desarrollo del producto se tendría que ver que estándares internacionales se quieren obtener, y en función de ellos ver que modelos se tienen que aplicar para obtenerlos.

De todo esto se puede deducir que no hay un modelo de calidad cerrado, pero si que existe un primer nivel de descomposición de calidad extendido y aceptado, el estándar 9126, a partir de ahí se apuesta por la utilización de los modelos de calidad abiertos donde cada uno debe considerar aquello que más le interese para su caso en particular.

A continuación se muestra la propuesta realizada por Francisco Montero [Montero, 2005], que apuesta por una mezcla entre los estándares internacionales y los criterios ergonómicos.

Calidad	Factor	Criterio	Contribución
Usability	**Understandability**	compatibility	Alta
		Legibility	Media
		Prompting	Media
		immediate feed-back	Alta
		significance of codes and behaviour	Alta
		helpfulness	Media
	Learnability	Grouping	Alta
		minimal actions	Alta
		conciseness	Baja
		information density	Media
		consistency	Alta
	Operability	explicit user action	Alta
		user control	Alta
		user experiencie´s	Alta
		flexibility	Media
		error protection	Alta
		quality of error messages	Baja
		error correction	Media
		privacy policies	Baja
		accesibility	Alta

Tabla 3.2 Propuesta de modelo de calidad centrado en la usabilidad.

Las principales características que ofrece el modelo de calidad recogido en la Tabla 3.2. son:

- **Está basado en estándares internacionales:** como la ISO 9126 o la ISO 9241 relacionados con la calida y en otros como la ISO 12407 y la 18529.
- **Es abierto:** Pueden incorporarse nuevos criterios de calidad, ligándose a los ya identificados en el mismo.
- **Es cerrado,** y puede utilizarse para dar asistencia a la puesta en práctica de técnicas de evaluación de interfaces de usuario.
- Utiliza los **criterios ergonómicos.**
- **Permite la puesta en práctica de técnicas de diseño centradas en el usuario.**

3.8 Conclusiones

En este capítulo se puede concluir que la usabilidad es muy importante a la hora de llevar a cabo el desarrollo de un software; en todos los puntos que se han tratado en este capitulo se han dado razones para integrar la usabilidad dentro del proceso de desarrollo software; ya que se considera que la usabilidad debe formar parte de dicho proceso y no ser un proceso aparte que sea aplicado por personas ajenas al desarrollo del mismo, sino que deben ser los propios desarrolladores los que deben tener en cuenta la usabilidad.

También tenemos en todos los puntos que la usabilidad debe aplicarse desde el principio del proceso de desarrollo ya que esto reducirá considerablemente el esfuerzo y los costes económicos del mismo.

Del mismo modo nos encontramos con que no se tiene un proceso definido de cómo debe aplicarse la usabilidad ni de que pasos se deben seguir, sino que queda reservado a lo que el ingeniero del software crea oportuno, ya que es este quien al final decide que proceso llevar a cabo para aplicar la usabilidad en su proyecto.

Además se tiene que tanto en la ingeniería del software como en la interacción persona-ordenador se nos indican unos procesos para llevar a cabo la usabilidad, pero en ambos casos no coinciden, estas dos disciplinas (IS e IPO) serán tratadas posteriormente en esta memoria.

CAPÍTULO 4 DESARROLLO CONCEPTUAL DE eAula

En este capitulo se va a explicar en que consiste la aplicación eAula, que partes la forman y como interactúan entre ellas teniendo en cuenta los conceptos que se han visto en capítulos anteriores. eAula es una aplicación para impartir clases a niños de edades comprendidas entre los 3 y los 5 años, por lo tanto, se tiene que el ámbito de aplicación de la misma será un colegio de primaria.

Para el desarrollo de esta aplicación se han tomado como referencia otras herramientas con funcionalidad parecida, tal y como se comentó en el capítulo 2 de esta memoria; incluyendo algunos de los funcionamientos que parecían adecuados e intentando corregir algunos otros que no parecían idóneos, sobre todo teniendo en cuenta el rango de edad en el que se mueve este proyecto, y que es una característica distintiva frente al resto de aplicaciones de este tipo.

Para llevar a cabo el análisis de la aplicación se aplicarán técnicas de Ingeniería del Software de análisis de requisitos funcionales, diagramas de clases y casos de uso; también se incluirá algún diagrama de secuencia para especificar mejor el comportamiento de la aplicación. Sólo con los casos de uso no se pueden describir las tareas del usuario desde el punto de vista final, por lo que, se usarán técnicas de Interacción Persona-Ordenador como el análisis de tareas, los diagramas de tareas, los diagramas de presentación y técnicas de prototipado de distinto nivel. Todas estas técnicas se aplicarán en un proceso iterativo e incremental. Las fases serán por tanto, fase de análisis de requisitos, fase de diseño abstracto y fase de implementación, tal y como queda reflejado en la figura 4.1., aunque dicha metodología no se seguirá de una

manera estricta, ya que el paso de una fase a la siguiente no se realiza de manera automática, especialmente en el último paso.

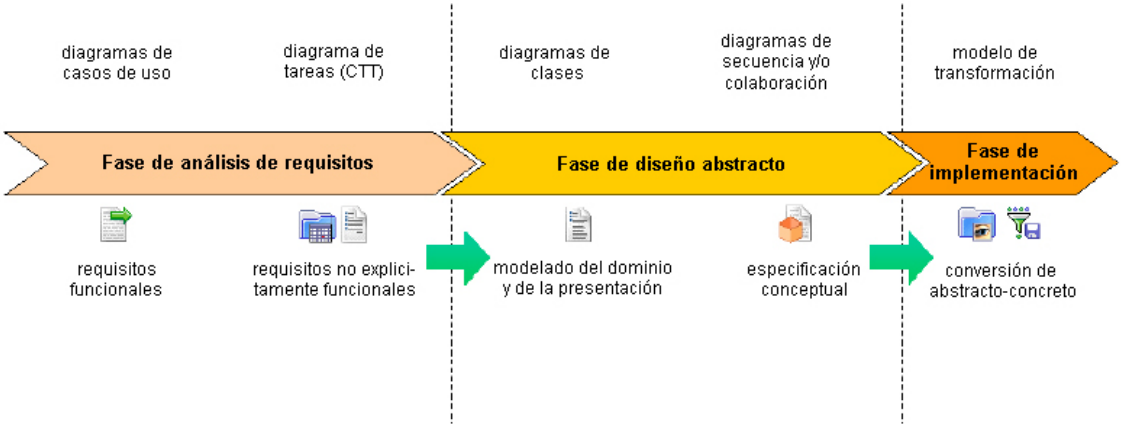

Figura 4.1 Metodología seguida en el desarrollo de la aplicación

4.1 Ámbito

El ámbito del aula virtual se centra en el funcionamiento de una clase de primaria (3-5 años), centrándose en la gestión del desarrollo de las actividades realizadas por los alumnos en unas determinadas asignaturas, es decir, con eAula se gestionan todas aquellas actividades que se realicen en el ordenador; en el caso que nos ocupa se encarga de actividades de tipo puzzle y asociaciones, puesto que son éstos los dos tipos de actividad que ofrece eAula actualmente.

A continuación se numeran unas consideraciones previas sobre el sistema que se han tenido en cuenta antes de realizar el análisis:

- Cada vez que un alumno realice una actividad el resultado de la misma quedará almacenado para poder ser consultado posteriormente.
- Cada alumno tendrá un nombre de usuario único de tal forma que en un mismo ordenador pueden sentarse diferentes alumnos siempre y cuando se inicie sesión con su correspondiente nombre de usuario.
- Varios alumnos de un mismo aula pueden estar realizando actividades distintas al mismo tiempo, ya que el envío de actividades puede ser totalmente personalizado.

El análisis del caso de estudio comienza con la fase de análisis de requisitos, donde se especifican los requisitos del sistema partiendo de los requisitos desde un

punto de vista más general, aumentando posteriormente el nivel de concreción de los mismos.

4.2 Fase de análisis de requisitos

En esta fase se llevará a cabo la familiarización con el entorno del caso práctico, para ello se hará la captura de requisitos, tanto funcionales como no explícitamente funcionales; para lo que se utilizarán los diagramas de casos de uso, los diagramas de secuencia y la metodología RUP para los requisitos funcionales, y criterios ergonómicos y CTT para el análisis de tareas, prestándose especial atención a la consideración de la calidad concretada en criterios ergonómicos (véase el Capítulo 3)

4.2.1 Estructura de “eAula”

La aplicación se divide en tres partes, tal y como se mostrará en la figura 5.1. del siguiente capítulo. Por un lado se tiene la aplicación del alumno con la que éste interactúa con el sistema y realiza las tareas que le encomienda el profesor, éste a su vez tiene una aplicación que le permite enviar actividades a los alumnos de manera individual y en grupo tantas actividades como quiera y a tantos alumnos como crea necesario, teniendo en cuenta que de este modo se permite una enseñanza totalmente personalizada atendiendo en cada momento a las carencias de cada alumno de manera individual.

También se tiene el servidor que interactúa a su vez con la aplicación del profesor y con la aplicación del alumno, de una manera totalmente transparente para ellos, de tal forma, que ambos pueden enviar datos en tiempo real al otro sin tener que preocuparse de cómo se lleva a cabo esta tarea, éstos datos se envían en formato XML.

En los siguientes puntos de este capítulo se verá detenidamente cada una de estas fases del desarrollo de la aplicación siguiendo la metodología indicada en la figura 4.1.

4.2.2 Requisitos funcionales

Aunque para la adquisición de requisitos se ha seguido la metodología de RUP, se debe indicar que al no ser el objetivo principal seguir estrictamente toda la metodología, ésta se ha utilizado con cierta flexibilidad. Los requisitos principales de la aplicación se pueden resumir en los siguientes:

- Dentro de una misma clase, diferentes alumnos pueden estar realizando diferentes actividades al mismo tiempo.
- Los alumnos pueden realizar las actividades a diferente ritmo, sin que la velocidad de uno interfiera en el otro.
- El profesor puede saber en cada momento qué alumnos están conectados a la aplicación.
- El profesor recibe los resultados de las actividades de cada alumno de manera unívoca y de forma transparente para él, sin tener que ir revisando uno a uno los ordenadores de los alumnos.
- El alumno recibe de manera automática las actividades que le envíe el profesor
- El servidor se encarga de guardar los resultados obtenidos por los alumnos.

Teniendo en cuenta que en este proyecto, la usabilidad juega un papel destacado se debe resaltar otro requisito muy importante para la usabilidad de la aplicación, que tanto la interfaz del profesor como, especialmente, la del alumno debe ser intuitiva y fácil de utilizar, es decir, debe ser una interfaz con alto grado de usabilidad que facilite que se puedan llevar a cabo las funciones para las que está pensada de una manera intuitiva para una persona sin conocimientos previos en este tipo de aplicaciones; sobre todo, teniendo en cuenta que va dirigida a niños cuya primera experiencia con un ordenador, debido a su edad, sería prácticamente el uso de esta aplicación.

4.2.3 Diagramas de casos de uso

A continuación se muestran dos diagramas de casos de uso, en el primero que es el diagrama correspondiente al alumno, se muestra la funcionalidad de la aplicación del alumno, en la figura 4.2 y en el diagrama de casos de uso de la figura 4.3. se muestra la funcionalidad correspondiente a la aplicación del profesor; se han realizado los diagramas del Profesor y del Alumno al considerarse éstos los más relevantes en el funcionamiento del sistema.

Para la realización de estos diagramas se ha utilizado la herramienta **Rational Rose** [http://www-306.ibm.com/software/rational/] que es una herramienta CASE de referencia para la especificación gráfica con la notación UML. La aplicación eAula ofrece dos caras que se corresponden con sendos conjuntos de actividades asociadas con

el rol de profesor y de los alumnos. Seguidamente se caracterizarán dichos puntos de vista a través de los diagramas de casos de uso y utilizando actores y casos de uso.

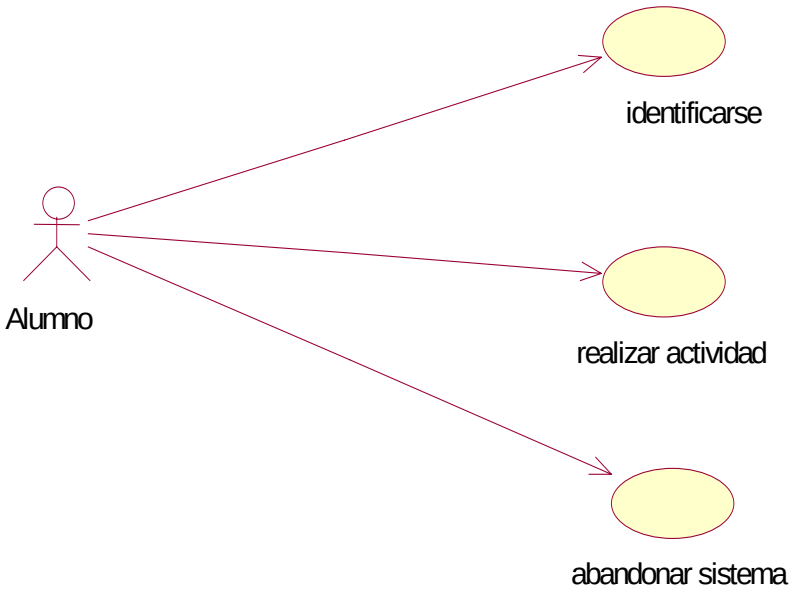

Figura 4.2 Diagrama de casos de uso "Alumno"

En el diagrama de la figura 4.2. se observa un único actor "*Alumno*" que interactúa con el sistema (vista del alumno). En alumno es el responsable de la manipulación de la aplicación.

La descripción de los casos de uso que componen el diagrama citado se realizará en las tablas 4.1., 4.2. y 4.3. respectivamente.

CU –01	**identificarse**
Descripción:	El usuario se identifica en la aplicación para acceder al sistema.
Precondición:	En este caso no se tiene ninguna precondición ya que es lo primero que se debe hacer para entrar en el sistema.
Flujo de eventos:	1. El usuario introduce un identificador.
	2. El usuario acepta
	3. El servidor introduce al alumno en el sistema.
	4. Devuelve ACK.
Flujo de eventos alternativo:	1.1.Si se pulsa a “Aceptar” sin haber introducido ningún dato, se muestra un mensaje de aviso para el alumno.

Tabla 4.1 CU-01: identificarse (alumno).

CU –02	**realizar actividad**
Descripción:	El usuario recibe una actividad o actividades y las realiza mediante un determinado número de pasos, devolviendo después un resultado.
Precondiciones:	1.El usuario debe haberse identificado para acceder al sistema. 2.El profesor debe haberle enviado al menos una actividad.
Flujo de eventos:	1. El usuario recibe la actividad
	2. Realiza la actividad siguiendo unos determinados pasos dependiendo de que tipo de actividad se trate.
	3.Los resultados se envía al servidor.
	4. El servidor envía los resultados al profesor y los almacena en el histórico.
Flujo de eventos alternativo:	2.a.Si el usuario va a la pantalla de salida y luego vuelve a la actividad, seguir con el paso 2. 2.b.Si el usuario sale de la aplicación sin terminar la actividad, se interrumpe la realización de la actividad y los resultados obtenidos no se envían al servidor.

Tabla 4.2 CU-02: realizar actividad.

CU –03	**abandonar sistema**
Descripción:	El usuario decide abandonar el sistema y pulsa salir.
Precondición:	El usuario debe encontrarse dentro del sistema.
Flujo de eventos:	1. El usuario pulsa cerrar para salir del sistema.
	2. La aplicación muestra la pantalla de salida y pide confirmación al usuario.
	3.El usuario confirma que desea salir de la aplicación.
	4. La aplicación se cierra y el servidor informa al profesor de que el alumno ha abandonado el sistema.
Flujo de eventos alternativo:	3.1. Si el usuario no confirma, volver al paso 2.

Tabla 4.3 CU-03: abandonar sistema (alumno).

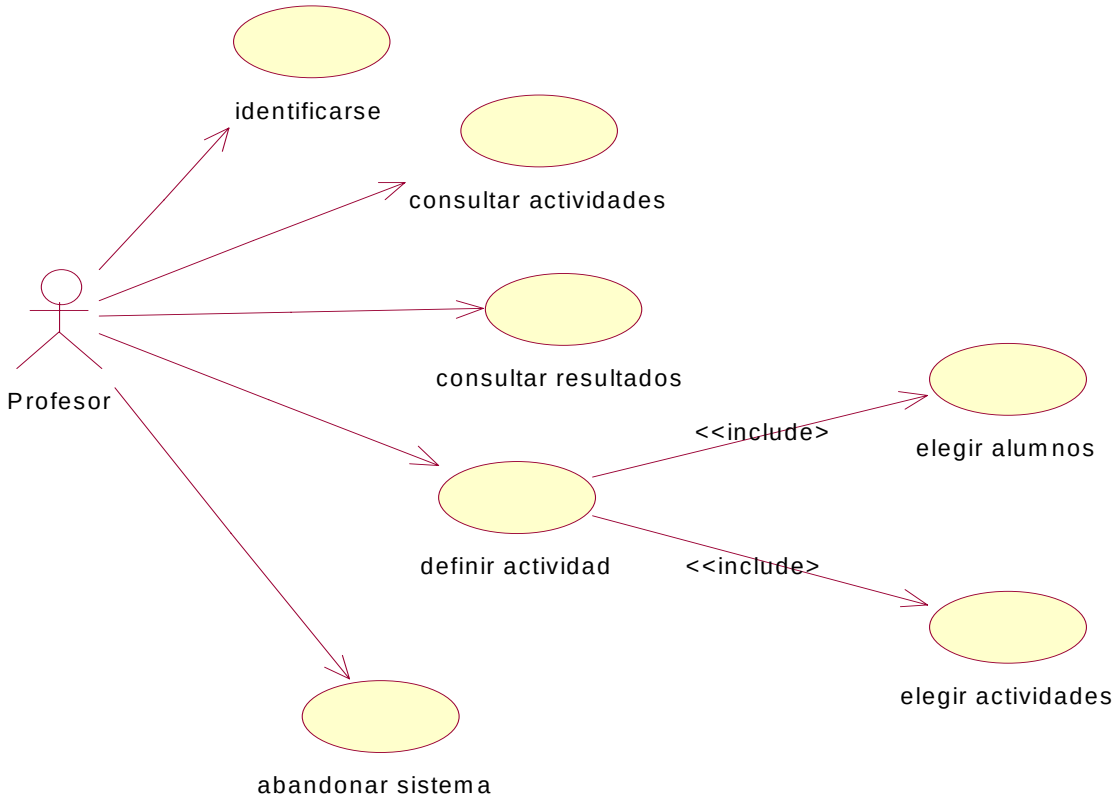

Figura 4.3 Diagrama de casos de uso “Profesor”

En el diagrama de la figura 4.3. al igual que pasaba en el de la figura 4.2., se observa un único actor, en este caso es el "*Profesor*" quien interactúa con el sistema desde el punto de vista del profesor. Es por tanto el profesor el responsable de la manipulación de la aplicación.

La descripción de los casos de uso que componen el diagrama citado se realizará en las tablas 4.4., 4.5. , 4.6., 4.7. y 4.8. respectivamente.

CU –04	**identificarse**
Descripción:	El usuario se identifica en la aplicación para acceder al sistema.
Precondición:	En este caso no se tiene ninguna precondición ya que es lo primero que se debe hacer para entrar en el sistema.
Flujo de eventos:	1. El usuario introduce un identificador.
	2. El usuario acepta
	3. El servidor introduce al profesor en el sistema.
	4. Devuelve ACK.
	5.El servidor envía al profesor datos sobre las actividades y los alumnos disponibles en el sistema.
Flujo de eventos alternativo:	1.1.Si se pulsa a "Aceptar" sin haber introducido ningún dato, se muestra un mensaje de aviso para el profesor.

Tabla 4.4 CU-04: identificarse (profesor).

CU –05	**consultar actividades**
Descripción:	El usuario puede en cualquier momento actualizar las actividades, para ver las que están disponibles en el servidor.
Precondición:	El usuario debe haberse identificado para acceder al sistema.
Flujo de eventos:	1. El usuario pulsa sobre el botón de actualizar actividades.
	2.El servidor envía al profesor la lista actual de actividades disponibles en el servidor.
Flujo de eventos alternativo:	En este caso no hay flujo alternativo.

Tabla 4.5 CU-05: consular actividades.

CU –06	**consultar resultados**
Descripción:	El usuario puede en cualquier momento consultar los resultados generales almacenados en el servidor (histórico).
Precondición:	El usuario debe haberse identificado para acceder al sistema.
Flujo de eventos:	1. El usuario pulsa sobre el botón de ver resultados generales.
	2.El servidor envía al profesor los datos almacenados en el histórico.
	3.El usuario vuelve a la aplicación general.
Flujo de eventos alternativo:	3.1.El usuario borra el histórico. 3.2El usuario vuelve a la aplicación general.

Tabla 4.6 CU-06: consultar resultados.

CU –07	**definir actividad**
Descripción:	El usuario puede definir una actividad/es en concreto para un alumno/s concretos.
Precondiciones:	1.Debe haber actividades definidas en el servidor. 2.Debe haber alumnos disponibles dentro del sistema.
Flujo de eventos:	1. El usuario elige una actividad o un grupo de actividades de la lista que se le ofrece.
	2.El usuario elige un alumno o un grupo de alumnos de la lista que se le ofrece.
	3.El usuario pulsa sobre enviar.
	4.El servidor envía la actividad o actividades correspondientes al alumno o alumnos indicados.
Flujo de eventos alternativo:	En este caso no se tiene flujo de eventos alternativo.

Tabla 4.7 CU-07: definir actividad.

CU –08	**abandonar sistema**
Descripción:	El usuario decide abandonar el sistema y pulsa salir.
Precondición:	El usuario debe encontrarse dentro del sistema.
Flujo de eventos:	1. El usuario pulsa cerrar para salir del sistema.
	2. La aplicación se cierra.
Flujo de eventos alternativo:	En este caso no se tiene flujo de datos alternativo.

Tabla 4.8 CU-08: abandonar sistema (profesor).

4.2.4 Requisitos no explícitamente funcionales

Tal y como se comentó en el capitulo 3 de esta memoria, la usabilidad se quiere tener en cuenta desde el comienzo del proceso de desarrollo, a lo largo de los puntos que componen dicho capitulo se han comentado diversas formas de conseguir que un software sea usable. También se hablaba en el capitulo 3 del uso de criterios ergonómicos para caracterizar la usabilidad, en ese mismo capítulo se indicó en que medida se han utilizado para el desarrollo del software que nos ocupa, por lo que no se volverá a repetir lo mismo en este punto, sino que se remite al lector a la tabla 3.1. donde se indican dichos criterios, en que consisten y de que modo se ha llevado a cabo su uso en el desarrollo de eAula.

Lo que si se puede añadir en este punto, es que después de llevar a cabo la realización de los casos de uso se ha notado cierta carencia a la hora de tener en cuenta la usabilidad tal y como se indicaba en el capítulo 3, por eso en este punto también se llevó a cabo un prototipado que se muestra a continuación en las figuras 4.4, 4.5 y 4.6. y que complementó la especificación ingenieril descrita previamente.

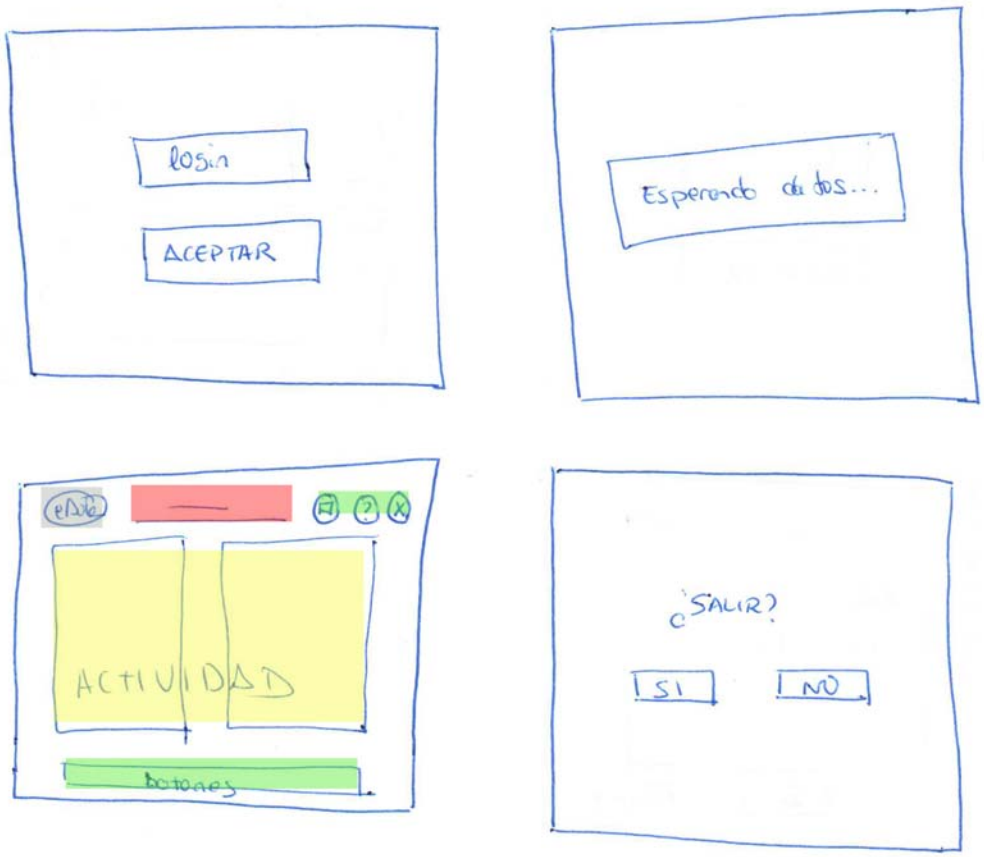

Figura 4.4 Prototipo de la interfaz del alumno.

En el dibujo que se muestra en la figura 4.4. se tiene un primer prototipo que se realizó para la aplicación del alumno, en dicho prototipo se puede observar arriba a la izquierda lo que será la pantalla de identificación del usuario, arriba a la derecha está lo que será la pantalla de espera del alumno, es decir en esa pantalla se quedaría la aplicación mientras estuviese esperando recibir datos por parte del profesor. Abajo a la izquierda se tiene la pantalla de la actividad, donde se ha reservado un determinado espacio para la actividad en sí (señalado en amarillo), otro espacio para la ayuda contextual (señalado en rojo), así como el espacio para los diferentes botones (ayuda, sonido, salir..., señalado en verde) y para el logo (señalado en gris) de la aplicación. Por último abajo a la derecha está lo que sería la pantalla de salida de la aplicación para que el alumno pueda confirmar que realmente quiere salir o volver atrás si ha pulsado el botón de salir por equivocación.

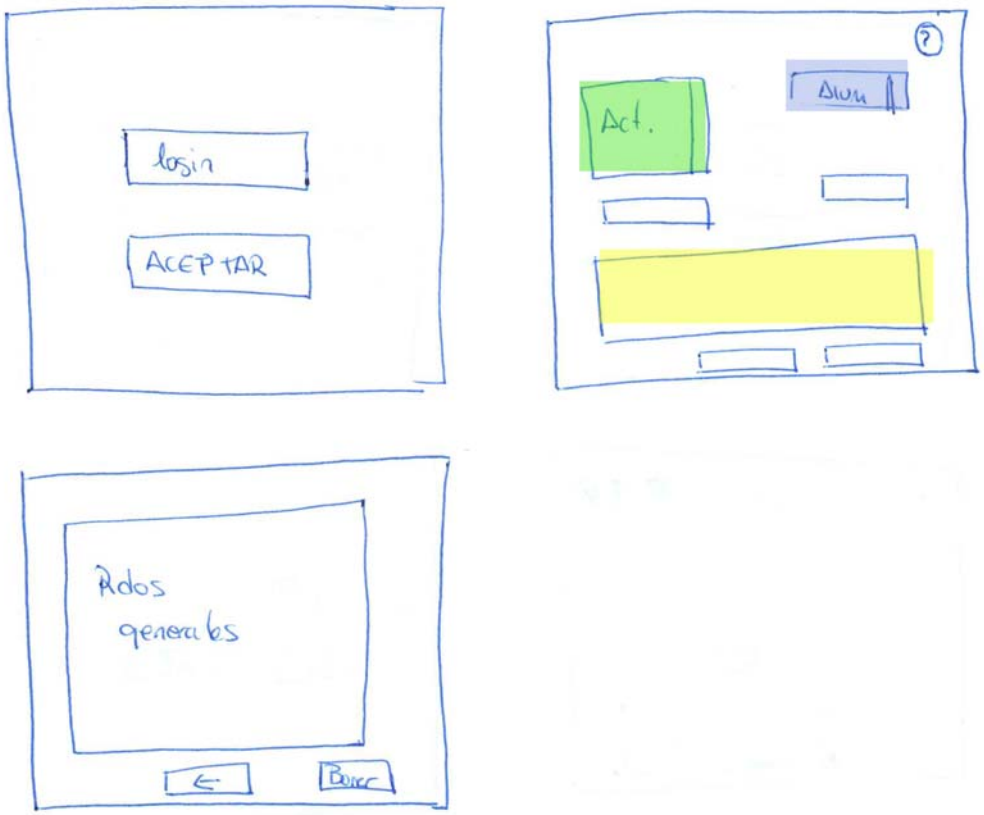

Figura 4.5 Prototipo de la interfaz del profesor

En la figura 4.5. se tiene un primer prototipo de lo que será la interfaz de la aplicación del profesor, arriba a la izquierda se tiene la imagen de lo que sería la pantalla de identificación del profesor, a la derecha se tiene lo que sería la pantalla principal de la aplicación, en ella se puede ver una zona reservada para la lista de las actividades (señalada en verde), una parte reservada para el alumno (señalada en azul) y una zona (señalada en amarillo) reservada para mostrar los resultados que irán obteniendo los alumnos en las diferentes actividades que realicen. Por último, abajo a la izquierda se tiene lo que será la pantalla para mostrar los resultados generales almacenados en el servidor, esta pantalla tendrá en el centro los resultados, ordenados de una determinada forma, y abajo dos botones, uno que le permitirá volver a la pantalla principal y otro que le permitirá al profesor borrar el histórico de resultados.

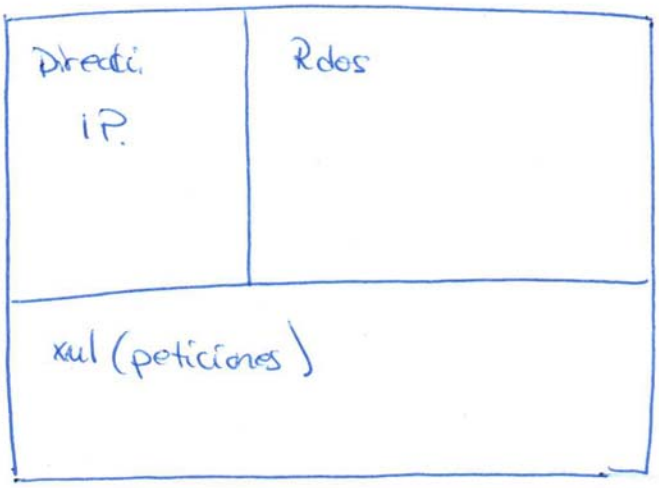

Figura 4.6 Prototipo de la interfaz del servidor.

En la figura 4.6. se muestra un prototipo inicial de lo que será la aplicación del servidor, en este caso se tiene una pantalla dividida a su vez en tres, una (arriba a la izquierda) para mostrar las direcciones IP de los usuarios que se encuentren conectados al servidor, otra (arriba a la derecha) que mostrará en todo momento el histórico actualizado en tiempo real y una tercera (abajo) que mostrará las peticiones xml que se le hagan al servidor.

4.2.5 Análisis de tareas

Después de llevar a cabo la definición de requisitos funcionales y con el fin de completar los escenarios de uso se utilizó una técnica de uso extendido en IPO como es el análisis de tareas. En este punto se llevará a cabo la definición de requisitos no explícitamente funcionales, para ello se llevará a cabo un análisis de tareas, ya que el análisis de tareas proporciona información muy relevante acerca del sistema a diseñar que a menudo no se recoge con las técnicas de requisitos tradicionales de la ingeniería del software.

El análisis de tareas es un término que cubre diferentes técnicas orientadas a describir las interacciones entre las personas y los entornos de una manera sistemática. Se puede definir como *el estudio de lo que un usuario tiene que realizar en términos de*

acciones y/o procesos cognitivos para conseguir un objetivo. Lo que hace, por tanto es ayudar en el proceso analítico de recogida de información, organizarlo y usarlo para realizar valoraciones o decisiones de diseño.

Para el desarrollo de los diagramas de tareas, se ha utilizado la notación ConcurTaskTrees (CTT), pero se han encontrado ciertas limitaciones para el modelado de tareas cuando las tareas que se pretenden modelar tienen un alto grado de libertad como es el caso que nos ocupa.

4.2.5.1 Notación Concur Task Trees (CTT)

CTT es una notación cuyo principal objetivo es el de representar las relaciones temporales existentes entre las actividades y usuarios que son necesarios para llevar a cabo las tareas. Se trata de una notación gráfica, descriptiva y simple en forma de árbol, donde se representa mediante una descomposición jerárquica las tareas existentes en un sistema. CTT permite la utilización de un conjunto de operaciones, tomadas de LOTOS, para describir las relaciones temporales entre tareas y representar operaciones secuenciales, concurrentes, recursivas, iterativas, opcionales, etc., algunas de las que se han identificado delicadas a la hora de ser representadas utilizando los diagramas facilitados con UML. En CTT se pueden distinguir cinco tipos diferentes de tareas en función del actor o actores que las lleven a cabo, dichas tareas se pueden consultar en la tabla 4.9.

CTT	IdealXML	Descripción
		Tareas del usuario
		Tareas de aplicación
		Tareas de interacción
		Tareas abstractas
		Tareas de cooperación

Tabla 4.9 Tipos de tareas de CTT.

En CTT, junto a los tipos de tareas definidos, se dispone de una serie de operadores para la descripción de relaciones temporales entre las tareas especificadas. Los operadores definidos son los recogidos en la tabla 4.10.

Operador	**Descripción**
T1 ||| T2	*Concurrencia*. Las acciones pueden realizarse en cualquier orden sin ningún tipo de restricción
T1 |[ ]| T2	*Sincronización*. Las acciones pueden realizarse en cualquier orden, pero deben sincronizarse para intercambiar información
T1 >> T2	*Activación*. Una tarea permite la ejecución de otra cuando termina
T1 [ ]>> T2	*Activación con paso de información*. T1 proporciona información a T2 y permite su ejecución cuando finaliza.
T1 [ ] T2	*Elección*. Es posible la elección entre un conjunto de tareas
T1 [> T2	*Desactivar*. La primera tarea es desactivada cuando comienza la ejecución de la segunda
T1 |> T2	*Suspender/Resumir*. T2 tiene la posibilidad de interrumpir a T1 que podrá ser retomada cuando aquella finalice
T1 |=| T2	Independencia en cuanto al orden. Ambas tareas pueden ser realizadas, pero una vez comenzada una debe finalizar antes de que comenzar con la otra T1 |=| T2 [T] La realización de la tarea es opción
T*	Realización reiterada de una tarea

Tabla 4.10 Operadores temporales de CTT.

Para realizar los diagramas de tareas se ha utilizado la herramienta IdealXML [http://www.info-ab.uclm.es/personal/FranciscoMSimarro/IdealXML.htm]. Esta herramienta es un entorno que permite modelar los diagramas de tareas y las especificaciones abstractas de la interfaz de usuario.

En IdealXML, los iconos de las tareas son diferentes a los de la notación CTT, la comparación entre ambos se puede ver en la tabla 4.9, donde la columna central está ocupada por los iconos que utiliza IdealXML.

Una vez hechas las aclaraciones pertinentes, se mostrarán los diagramas de tareas. Todas las tareas que surgen de la aplicación siguen una misma estructura, que se muestra a continuación en la figura 4.7.

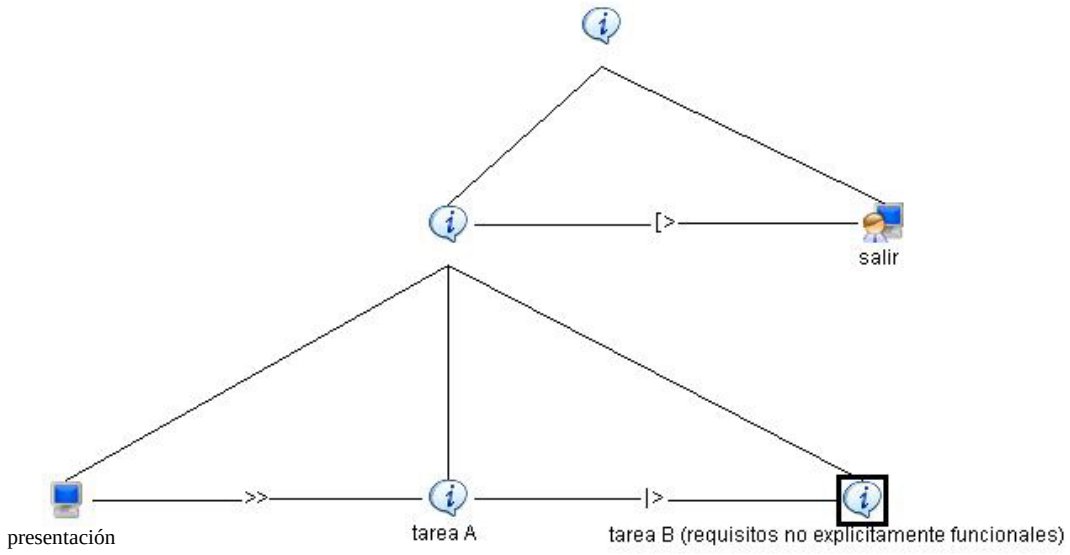

Figura 4.7 Diagrama general de tareas.

Las tareas más interesantes respecto a considerar un estudio relacionado con la usabilidad son las que tiene que ver con los casos de uso etiquetados con:

- realizar actividad (actor alumno).
- definir actividad (actor profesor) .

4.2.5.2 Tarea Realizar Actividad

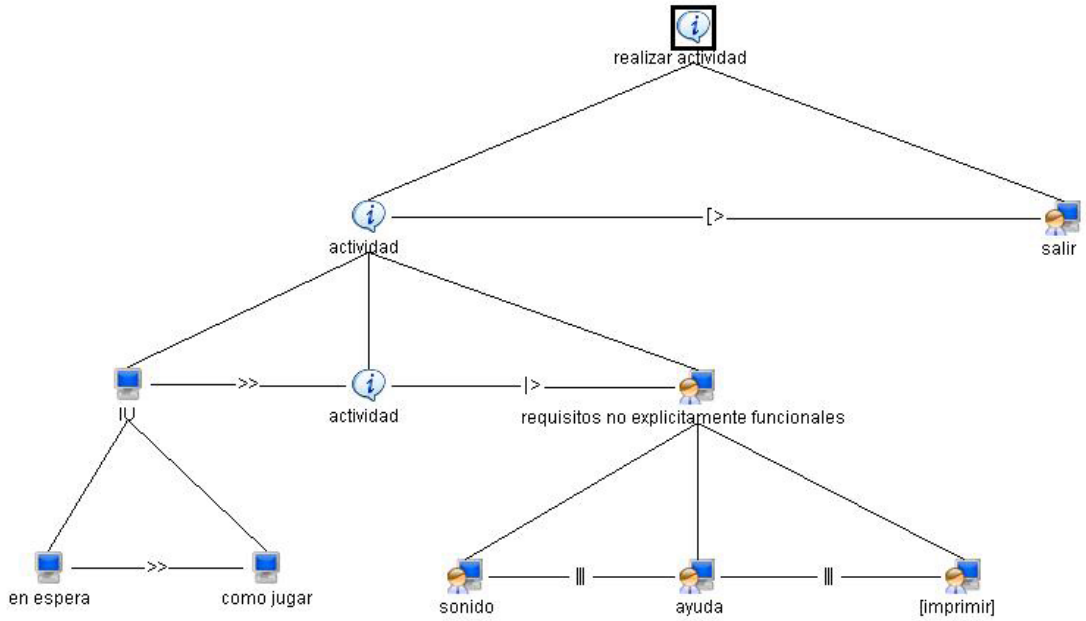

Figura 4.8 Tarea: Realizar actividad

En el diagrama de la figura 4.8. se muestra la tarea “Realizar Actividad”, esta tarea se compone a su vez de varias subtareas, de tipo aplicación nos encontramos con “en espera” y “como jugar” que son aquellas tareas que realiza la aplicación de manera automática, de tipo interacción tenemos “salir”, “sonido”,”ayuda” e “imprimir”, esta última con carácter opcional ya que depende de en que tipo de actividad estemos se da o no. Por último nos encontramos con la tarea abstracta “actividad”, la cual a su vez contendría varias tareas, de pendiendo de que tipo de actividad se tratase, por eso se deja como tarea abstracta.

Este diagrama de tareas se encuentra directamente relacionado con el caso de uso “realizar actividad” (tabla 4.2).

4.2.5.3 Tarea Definir Actividad

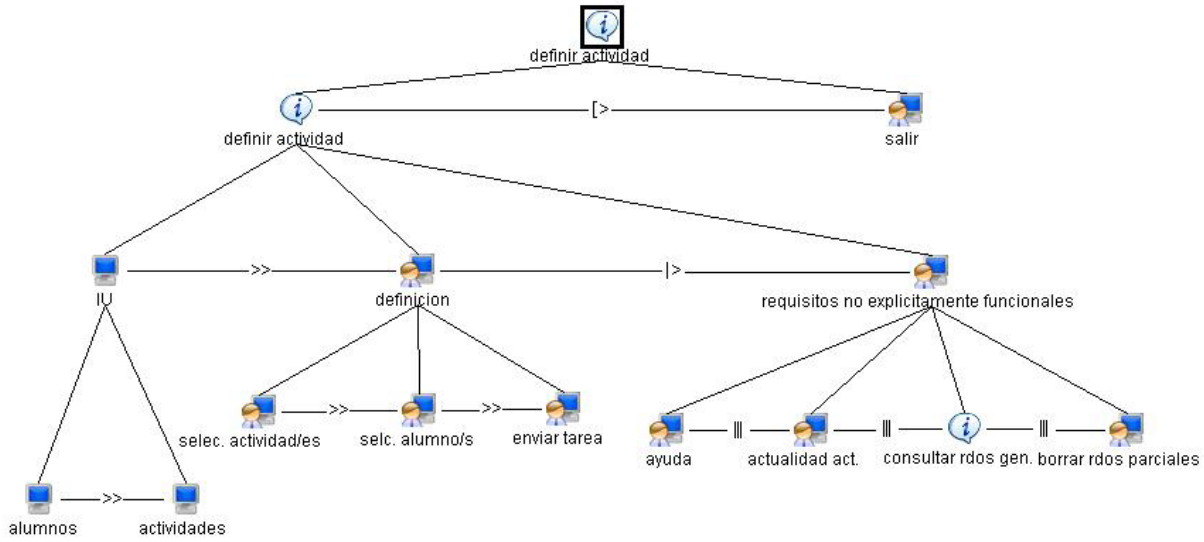

Figura 4.9 Tarea: Definir actividad.

Se debe tener en cuenta que la presencia o ausencia de los requisitos no explícitamente funcionales en la interfaz no impide la realización de la tarea, pero si que influye en la facilidad de uso y en la satisfacción del usuario.

En el diagrama de la figura 4.9. se muestra la tarea “Definir Actividad”, esta tarea se descompone a su vez en varias subtareas de entre las cuales se pueden definir tres grupos claramente diferenciados: previas, actividad en sí y colaterales.

El primer grupo sería aquel que engloba las tareas “IU”,”alumnos” y “actividades”, estas subtareas se podrían definir como previas ya que se llevan a cabo por la aplicación antes de que el usuario realice ninguna tarea, dichas tareas tienen que ver directamente con la interfaz y con la usabilidad, ya que constan de la interfaz en sí, mostrar inicialmente las actividades que se encuentran disponibles en el servidor y mostrar los alumnos conectados al sistema respectivamente.

El segundo grupo lo comprenderían las tareas de interacción de “definición”,“selección de actividad/es”,”selección de alumno/s” y “enviar tarea”, estas tareas comprenderían la actividad en sí, es decir aquellas funciones que son realizadas por el usuario, en este caso por el profesor. La primera sería la definición en sí de la tarea que se va a encomendar a el/los alumno/s, después se tiene la selección de actividades de entre un grupo que se ofrece al usuario previamente, la selección de uno

o varios alumnos para que realicen dichas actividades y el envío en sí de la tarea a el/los alumnos destinados a su realización.

En el tercer grupo, por último se encuentran todas aquellas tareas colaterales relacionadas también con la usabilidad, entre las que se encuentran las tareas de interacción “ayuda”,”refrescar actividades” y “borrar” formando el conjunto de ellas los “requisitos no explícitamente funcionales”, también se encuentra dentro de este grupo la tarea abstracta “consultar resultados generales”, dicha tarea se ha definido como abstracta porque a su vez englobaría dentro de ella otras tareas, pero que no se indican aquí al no estar relacionadas con la tarea principal que nos ocupa, es decir “definir actividad”.

4.3 Fase de diseño abstracto

En esta fase, con la información que se obtiene de la fase anterior, se diseñará un modelo conceptual del caso de estudio. Para ello en este punto se van a utilizar los diagramas de clases y secuencia para la parte referente a la ingeniería del software y los diagramas de presentación para abarcar el punto de vista de los distintos usuarios del sistema.

4.3.1 Diagrama de clases

El diagrama de clases con la arquitectura que tiene el sistema se representa en la figura 4.10.

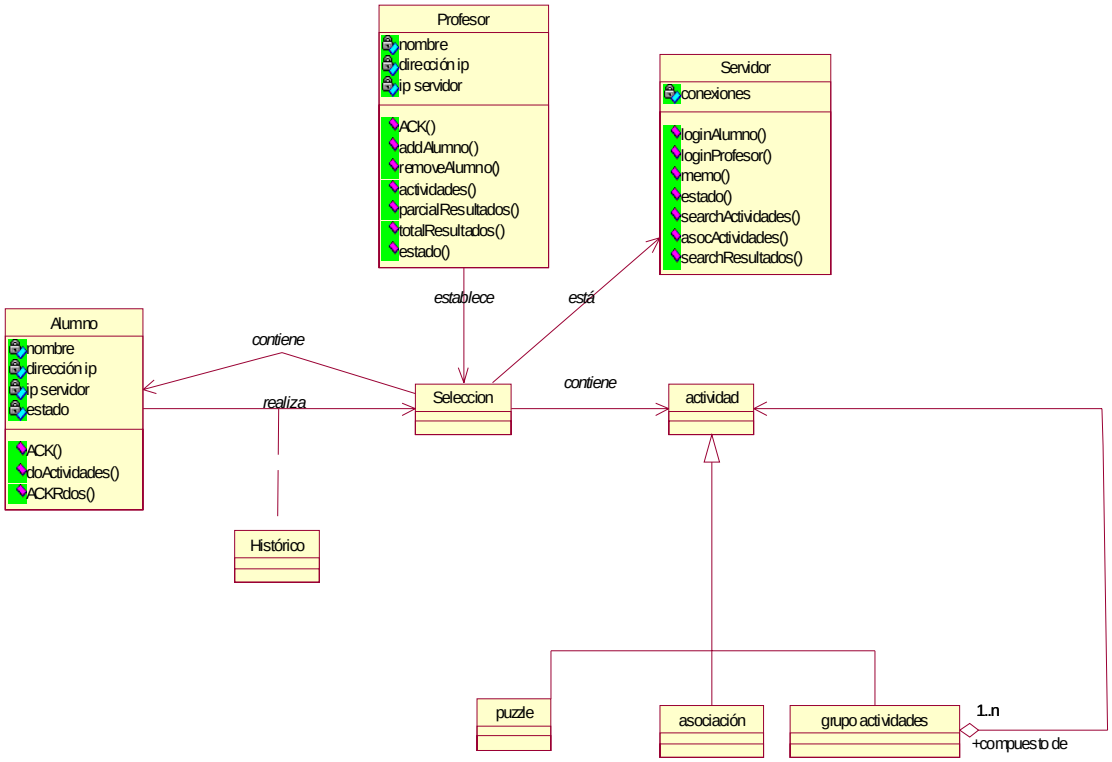

Figura 4.10 Diagrama de clases.

La aplicación se ha implementado en Flash por lo que las clases identificadas en el diagrama de la figura 4.10. no son clases reales, al no ser flash un lenguaje orientado a objetos; sino que se ha realizado una transformación libre de las partes que conforman las diferentes aplicaciones para construir un diagrama de clases que identifique el sistema.

A continuación se definirán las clases que componen dicho diagrama (tablas de la 4.11 a la 4.16), explicando a su vez los métodos que contienen, aclarando antes que los métodos de las clases indicadas se corresponden con las diversas peticiones de mensajes que se llevan a cabo durante el funcionamiento del sistema.

Propiedad	Descripción
Nombre:	Profesor
Descripción:	Esta clase representa todas las funciones que realiza el profesor.
Atributos:	*nombre*: nombre del profesor *dirección IP*: dirección IP de la máquina donde se está ejecutando la aplicación del profesor. *IP servidor*: dirección IP de la máquina donde se está ejecutando el servidor.
Métodos:	*ACK()*: la aplicación recibe una confirmación de conexión por parte del servidor. *addAlumno()*: se añade un alumno en la lista de alumnos del profesor. *removeAlumno()*: se elimina un alumno de la lista de alumnos del profesor. *actividades()*: se actualiza la lista de actividades del profesor. *parcialResultados()*: se actualiza la lista de resultados parciales del profesor. *totalResultados()*: se muestran los resultados generales que tiene el histórico en la aplicación del profesor. *estado()*: se actualiza el estado de cada alumno de la lista de alumnos que tiene el profesor.

Tabla 4.11 Clase Profesor.

Propiedad	Descripción
Nombre:	Alumno
Descripción:	Esta clase representa todas las funciones que realiza el alumno.
Atributos:	*nombre*: nombre del alumno *dirección IP*: dirección IP de la máquina donde se está ejecutando la aplicación del alumno. *IP servidor*: dirección IP de la máquina donde se está ejecutando el servidor. *estado*: estado del alumno (esperando, ocupado)

Métodos:	*ACK()*: la aplicación recibe una confirmación de conexión por parte del servidor. *doActividades()*: el alumno realiza las actividades que le ha enviado el profesor. *ACKRdos()*: la aplicación recibe una confirmación del servidor de que los resultados de la actividad han sido recibidos correctamente.

Tabla 4.12 Clase Alumno.

Propiedad	**Descripción**
Nombre:	Servidor
Descripción:	Esta clase representa todas las funciones que realiza el servidor.
Atributos:	*conexiones()*: el servidor maneja la información de todas las conexiones del profesor y los alumnos con él.
Métodos:	*loginAlumno()*: el servidor recibe el login de un alumno en particular, envía una respuesta al alumno y envía al profesor que dicho alumno está conectado. *loginProfesor():* el servidor recibe el login del profesor y le envía una respuesta. *memo()*: el servidor recibe los resultados de cada una de las actividades realizada por cada uno de los alumnos, almacena los datos y envía los resultados parciales al profesor. *estado()*: el servidor recibe el estado del alumno (esperando, ocupado) y se lo envía al profesor. *searchActividades()*: el servidor busca las actividades que hay en el servidor y le envía al profesor una lista con las que hay. *asocActividades()*: el servidor recibe la asociación actividad-alumno que ha realizado el profesor y le envía a cada alumno la lista de actividades que tiene que realizar. *searchResultados()*:el servidor busca los resultados generales que se encuentran almacenados en el histórico y le envía la lista de resultados generales al profesor.

Tabla 4.13 Clase Servidor.

Propiedad	Descripción
Nombre:	Selección
Descripción:	Esta clase representa la selección que es llevada a cabo por el profesor, donde se seleccionan un conjunto de alumnos y actividades para que dichos alumnos puedan realizar las actividades elegidas. Esta clase carece te atributos y de métodos.

Tabla 4.14 Clase Selección.

Propiedad	Descripción
Nombre:	Histórico
Descripción:	Esta clase representa el histórico donde están almacenados todos los datos del sistema. Esta clase carece te atributos y de métodos.

Tabla 4.15 Clase Histórico.

Propiedad	Descripción
Nombre:	actividad
Descripción:	Esta clase representa cada una de las actividades que se envían al alumno, es una clase abstracta y forma parte de un “patrón composite” que se definirá a continuación. Esta clase carece te atributos y de métodos.

Tabla 4.16 Clase actividad.

El **patrón Composite** [Gamma et al.,1994] , utilizado para gestionar las distintas actividades, sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

En el caso que nos ocupa se tiene la necesidad de crear una serie de clases para guardar información acerca de una serie de actividades que serán puzzles y

asociaciones. Además se necesita poder tratar también grupos de actividades, ya que la aplicación permite la selección de varias actividades a la vez para ofrecérselas al alumnado. Esto se soluciona con la figura 4.11., que se muestra a continuación:

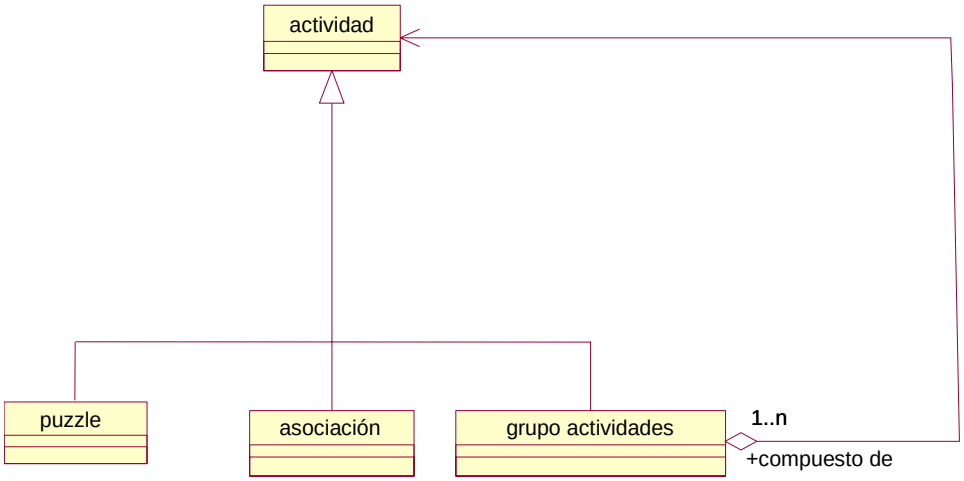

Figura 4.11 Diagrama composite.

4.3.2 Diagrama de Secuencia

Con el diagrama de secuencia se modela la interacción entre los objetos del sistema a través del tiempo, en él se refleja el paso de mensajes entre las diferentes aplicaciones de las que consta el sistema que nos ocupa, en la figura 4.12. mostrada a continuación, se tiene dicho diagrama.

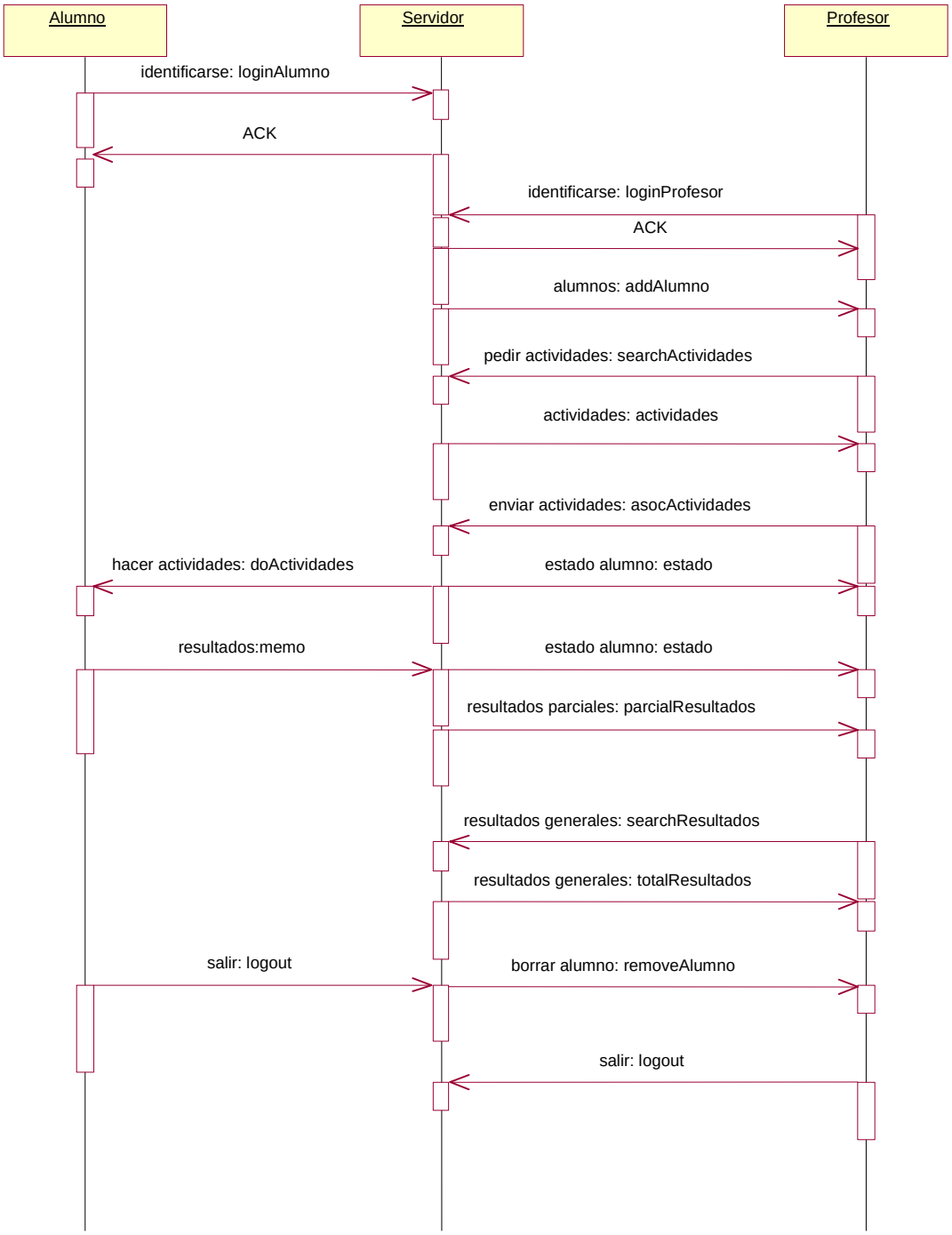

Figura 4.12 Diagrama de Secuencia.

En el diagrama de la figura 4.12, se puede observar el funcionamiento del sistema, ya que en él queda reflejado todo el paso de mensajes que se llevan a cabo para el correcto funcionamiento del sistema.

Primero se tienen que identificar tanto el alumno como el profesor para así poder entrar en cada una de sus aplicaciones correspondientes. A partir de entonces el

alumno se quedará esperando hasta que le llegue una actividad o paquete de actividades proveniente del profesor, entonces podrá realizar dichas actividades, cuando termine cada una de ellas la aplicación le enviará los resultados al profesor, y en el momento que quiera el alumno puede abandonar la aplicación.

Por otro lado el profesor una vez que se ha identificado pasa a disponer de la lista de actividades y de alumnos que hay en el sistema, a partir de entonces puede hacer la selección que desee de las unas y los otros y enviarlas a los respectivos alumnos para que realicen dichas actividades; en cualquier momento puede consultar los resultados generales, así como abandonar la aplicación.

4.3.3 Diagramas de presentación abstractos (AIO)

Es un diagrama de especificación de componentes a nivel abstracto donde se especifican aquellos mecanismos que deben proporcionarse al usuario, y que facilitan la realización de las tareas asociadas directamente con la calidad de la implementación ofrecida al usuario a través de la interfaz finalmente presentada. Modela por tanto los Objetos de Interacción Abstractos (AIO) necesarios para llevar a cabo las acciones requeridas por la tarea de identificación. Este diagrama se genera automáticamente con IdealXML desde el diagrama de tareas correspondiente, los iconos que utiliza IdealXML para la representación del diagrama de presentación son los mostrados en la tabla 4.17. a continuación:

Iconos	Descripción
	Icono asociado a un objeto *Container*. Objeto de interacción abstracto que permite aglutinar componentes en su interior.
	Icono asociado a un *Componente*. Objeto de interacción abstracto que mediante la incorporación de distintas facetas definidas en usiXML permite dar soporte a la interacción mantenida entre usuario y sistema.
	Iconos asociados a las diferentes facetas definidas en usiXML. A través de ellas es posible dotar a los componentes de interacción de facilidades para dar soporte a diferentes acciones de

	interacción, respectivamente, *entrada de datos*, *salida de datos*, *operaciones de control* y de *navegación*.

Tabla 4.17 Iconos del diagrama de presentación en IdealXML.

4.3.3.1 Realizar Actividad

Este diagrama (figura 4.13) está generado automáticamente a partir del diagrama de tareas de la figura 4.8.

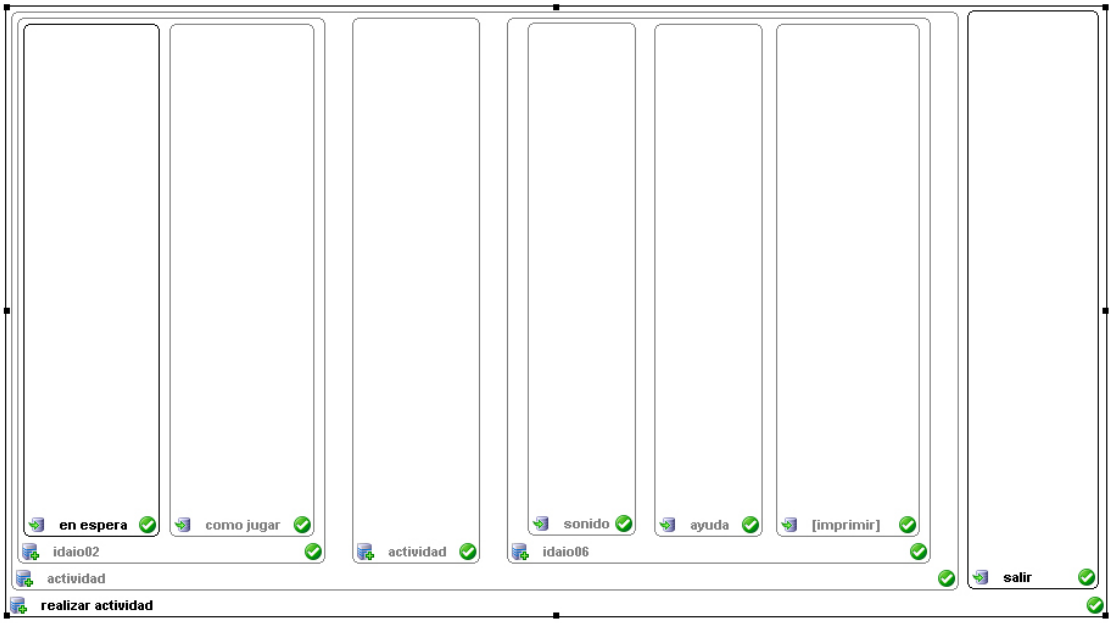

Figura 4.13 Diagrama de presentación: Realizar actividad.

El diagrama está formado por un objeto contenedor llamado “realizar actividad”, dentro del cual se tienen otro contenedor “actividad” y el componente “salir”.

El contenedor “actividad” a su vez está compuesto por el contenedor “idaio02”, por el contenedor “actividad” y por el contendor “idaio06”.

El contenedor “idaio02” está compuesto por el componente “en espera” y el componente “como jugar”, por último el contenedor “idaio06” está compuesto por los componentes “sonido”, “ayuda” e “[imprimir]”.

A continuación, en la figura 4.14. se hará una comparativa entre el diagrama de presentación que se obtiene después de realizar el diagrama de tareas y el prototipo inicial que se hizo de la interfaz.

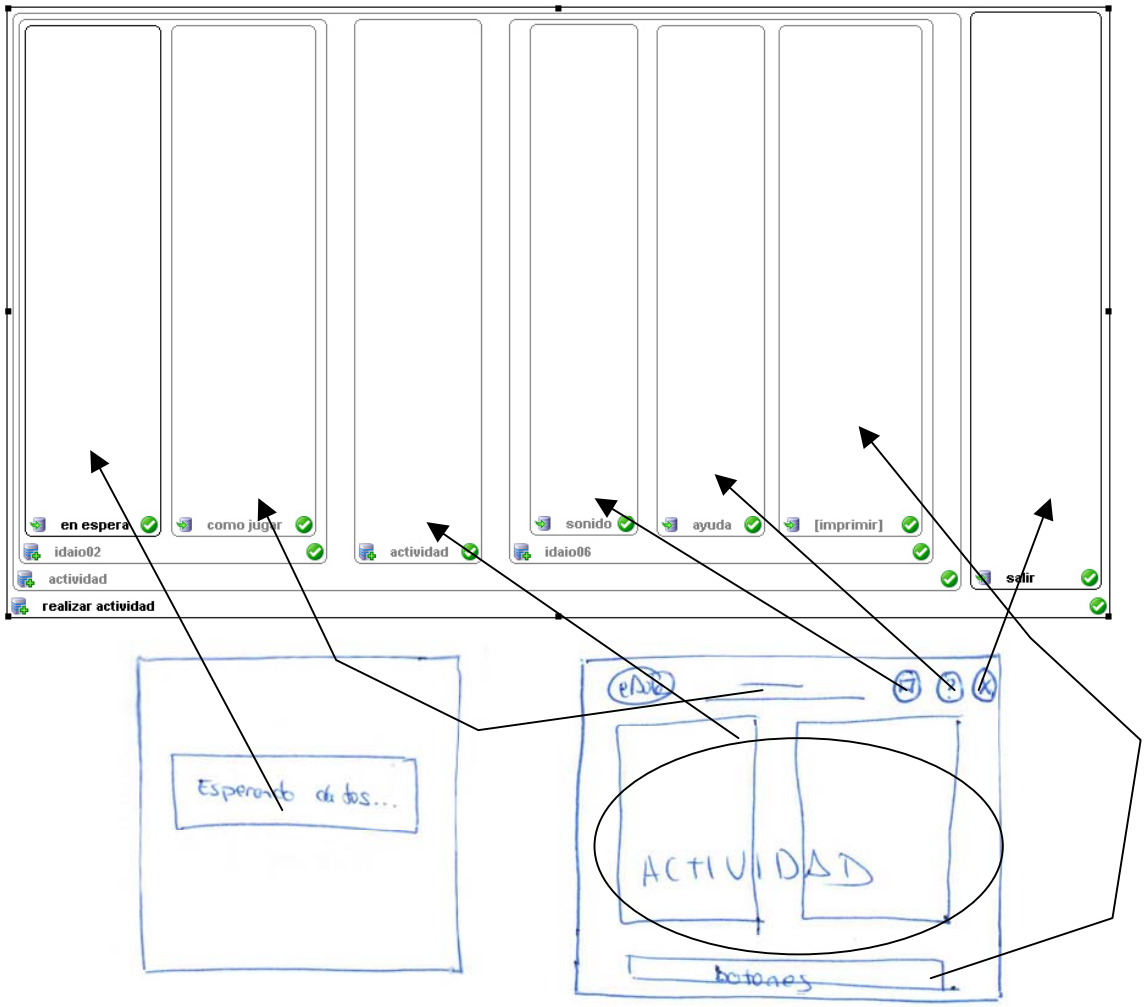

Figura 4.14 Comparativa de AIO (Realizar Actividad) con prototipo.

Se puede observar que el diagrama de presentación se acerca bastante a lo propuesto en el prototipo inicial. El principal objetivo con esta forma de trabajar es el de validar los requisitos lo antes posible dentro del proceso de desarrollo. Es para ello para lo que aunamos esfuerzos y técnicas procedentes de las dos disciplinas consideradas.

4.3.3.2 Definir actividad

Este diagrama (figura 4.15) está generado automáticamente a partir del diagrama de tareas de la figura 4.9.

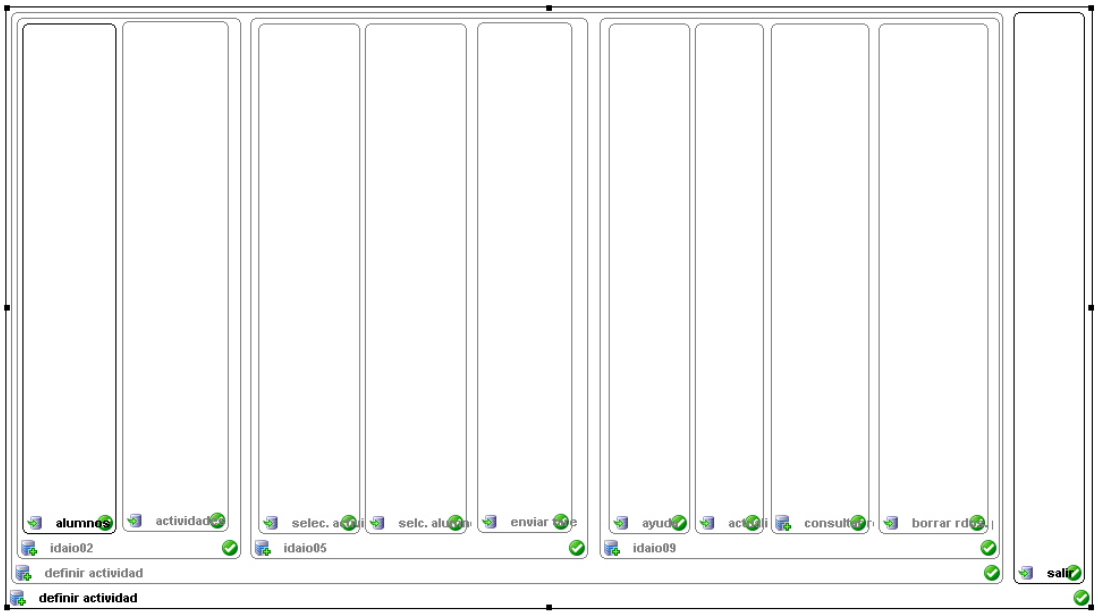

Figura 4.15 Diagrama de presentación: Definir actividad.

El diagrama está formado por un objeto contenedor llamado “definir actividad”, dentro del cual se tienen otro contenedor “definir actividad” y el componente “salir”.

El contenedor “definir actividad” a su vez está compuesto por el contenedor “idaio02”, por el contenedor “idaio05” y por el contendor “idaio09”.

El contenedor “idaio02” está compuesto por el componente “alumnos” y el componente “actividades”.

El contenedor “idaio05” está compuesto por el componente “selec. actividades”, por el componente “selec. alumnos” y por el componente “enviar tarea”.

Por último el contenedor “idaio09” está compuesto por los componentes “ayuda”, “actualizar actividades” y “borrar rdos. generales”, además de por el contenedor “consultar rdos. generales”.

A continuación, en la figura 4.16. se hará una comparativa entre el diagrama de presentación que se obtiene después de realizar el diagrama de tareas y el prototipo inicial que se hizo de la interfaz.

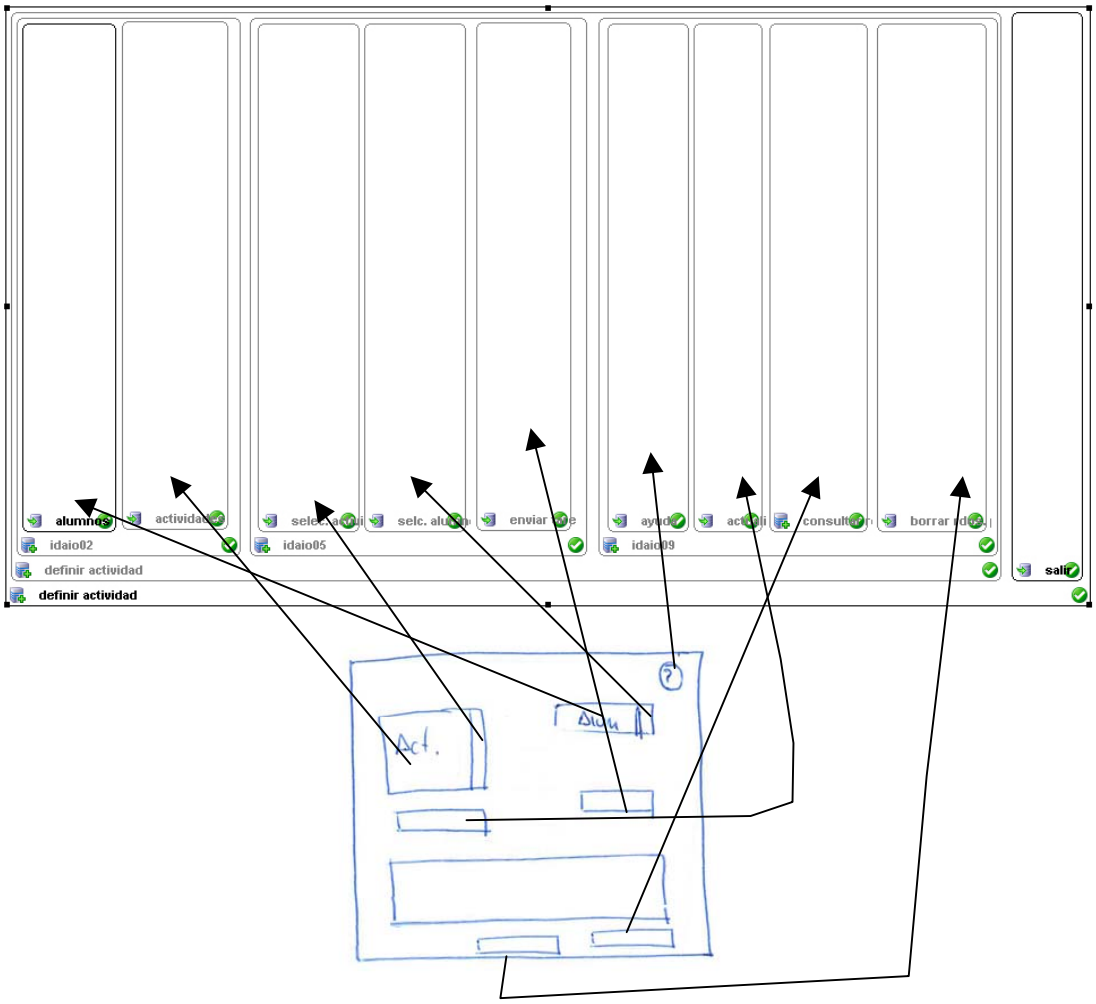

Figura 4.16 Comparativa de AIO (Definir Actividad) con prototipo.

Se puede observar que el diagrama de presentación se acerca bastante a lo propuesto en el prototipo inicial.

4.4 Conclusiones

En este capítulo se ha aplicado la metodología que se propuso al principio del mismo y que está reflejada en la figura 4.1. en un caso real, se han llevado a cabo las dos primeras fases de la misma, es decir, la “fase de análisis de requisitos” y la “fase de diseño abstracto”, teniendo en cuenta en ambos casos el punto de vista de la Ingeniería del Software y el punto de vista de la Interacción Persona-Ordenador, al ser estas las dos disciplinas que se han mantenido a lo largo de todo el desarrollo del proyecto.

Según dicha metodología al haberse hecho un desarrollo basado en modelos y la relación entre los mismos, el paso de la “fase de diseño abstracto” a la “fase de

implementación” debería ser automático, pero ya se indicó en puntos anteriores de esta memoria que en el caso que nos ocupa no va a ser así. La tercera y última fase de la metodología, la “fase de implementación”, se desarrollará en el siguiente capitulo de esta memoria, es decir, en el capitulo 5.Volver a resaltar la posibilidad e validar requisitos funcionales y no funcionales desde el principio. Además, también podríamos conseguir involucrar a los usuarios; profesores y alumnos con los prototipos elaborados, evaluándolos previamente a su implementación definitiva, que será descrita en el próximo capítulo.

CAPÍTULO 5 IMPLEMENTACIÓN Y DESCRIPCIÓN DE eAula

Tras haber desarrollado en anteriores capítulos como se ha desarrollado la herramienta que nos ocupa, en este capitulo se va ha definir como se distribuye, como se ha implementado y el funcionamiento de la misma.

Primero se concluirá la aplicación de la metodología que se inició en el capitulo anterior (figura 4.1.), terminándolo en este con la “fase de implementación”, indicado cómo se ha implementado y qué lenguajes se han utilizado.

Después se indicará el funcionamiento general de cada una de las partes que forman el sistema eAula para que el lector se pueda hacer una idea general del funcionamiento de la misma en conjunto.

Por último se darán unos manuales de las aplicaciones del profesor y el alumno, para que así quede totalmente claro no sólo su funcionamiento sino la utilización de cada una de ellas.

5.1 Estructura de eAula

eAula se puede dividir en tres bloques principalmente, por un lado se tiene la aplicación del profesor que tiene tres componentes que son por un lado el seguimiento del aula, el editor de actividades y el interpretador de resultados, en este proyecto solo se ha abarcado el seguimiento del aula; por otro lado se tiene la aplicación del alumno cuya finalidad es la de realizar actividades y por último se tiene la aplicación del servidor que tiene dos funciones principales la primera y más importante es la

distribución de datos y la segunda el almacenamiento de los mismos. Esta estructura queda reflejada a continuación en la figura 5.1.

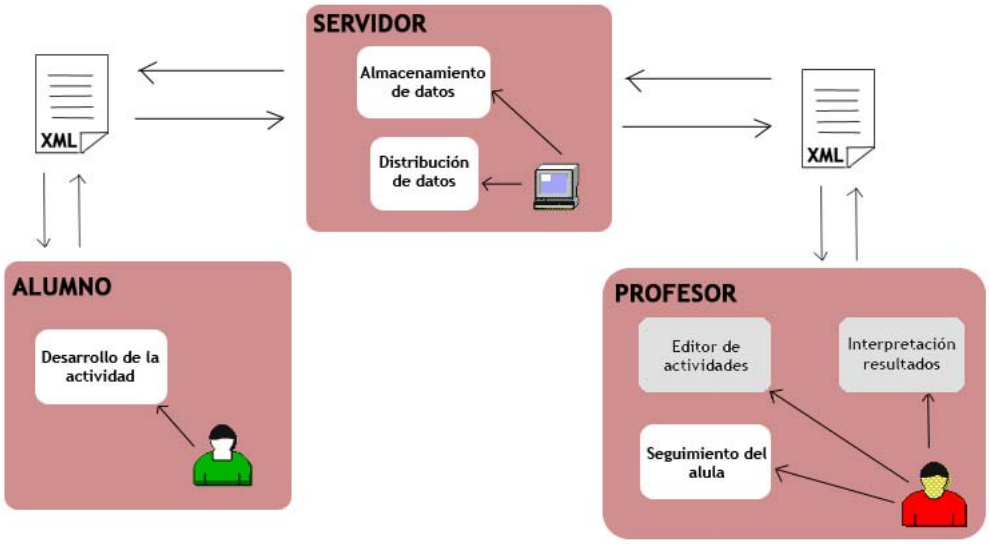

Figura 5.1 Bloques en los que se divide el proyecto.

Una vez que se han presentado a grandes rasgos cada una de las aplicaciones que conforman eAula se indicará en los siguientes apartados de este punto en que consiste cada una de las partes de la aplicación que se muestran en la figura 5.1.

5.1.1 Aplicación del alumno.

Esta parte del proyecto abarca la interacción final con el niño, es decir, mediante esta aplicación el alumno pueden realizar las actividades que el profesor seleccione de entre una lista de actividades posibles. Es por tanto la parte del aula en la que el niño se implica realmente, ya que el resto es transparente para él.

5.1.2 Aplicación del profesor.

Dentro del aula virtual la parte del profesor se puede dividir a su vez en otros tres módulos, que son:

- Editor de actividades.
- Seguimiento del aula.
- Interpretación de resultados.

En este proyecto se abarca la parte de seguimiento del aula tal y como se indicó en el apartado anterior, que es aquella mediante la cual el profesor selecciona las

actividades y las envía a los alumnos que decida. Así mismo puede consultar los resultados obtenidos por los mismos.

5.1.3 Servidor (conexión entre alumno y profesor).

El papel que juega el servidor en el aula virtual es la de comunicar la aplicación del alumno con la del profesor, esta comunicación se lleva a cabo mediante el intercambio de ficheros XML; por medio de esos ficheros se le enviá la información tanto al profesor como al alumno de manera transparente para ellos.

Más adelante, en este mismo punto, se describirán detenidamente cada una de estas partes, así como la funcionalidad de las mismas.

5.2 Fase de implementación

En esta fase de la metodología por tanto, se realizará el paso de los objetos de interacción abstractos diseñados en el capitulo anterior a la implementación de los mismos como objetos de interacción concretos (CIOs, Concrete Interation Objects) que serán los que al final formen parte de la interfaz final ejecutable del sistema.

Según se ha indicado anteriormente el objetivo del desarrollo de aplicaciones basadas en modelos es que partiendo de la especificación de la interfaz en modelos y seleccionando la plataforma destino, se realice una transformación automática. El problema que se presenta es que los lenguajes y herramientas que hay actualmente no consiguen la traducción cien por cien a la plataforma destino, por lo que en el caso de estudio que nos ocupa lo que se ha hecho es realizar el modelado y después basándose en el mismo se ha programado una interfaz usando unas determinadas herramientas, dichas herramientas han sido Flash 8 para las aplicaciones del profesor y el alumno y Turbo Delphi para el servidor.

A continuación se describirán los componentes de ambas plataformas que se han utilizado para la implementación de las respectivas aplicaciones.

5.3 Flash: implementación de las herramientas de profesor y alumno

Para el desarrollo de las aplicaciones del profesor y sobre todo la del alumno se ha optado por la implementación en flash, esta decisión se tomó por las facilidades que aporta esta plataforma a la hora de realizar aplicaciones más o menos animadas y más agradables para el usuario, sobre todo si se tiene en cuenta que dicho usuario está entre los 3 y los 5 años, como es el caso del alumno. ¿Qué facilidades son estas?

- Se pueden hacer animaciones fácilmente que hagan que la aplicación sea más amigable para un niño de las edades que nos ocupan.
- Se pueden incluir archivos de sonido en diferentes formatos, lo que hace que se pueda disponer de locuciones para que aquellos niños que todavía no saben leer con claridad reciban la explicación de cómo funciona la actividad que se disponen a realizar.
- Las aplicaciones se pueden ver sin necesidad de que el alumno vea el entorno externo a la misma, sobre todo en el caso del alumno, la aplicación le ocupa toda la pantalla del ordenador, por lo que su atención queda focalizada a la misma, sin que tenga botones ajenos a la misma que puedan confundirle.
- Puede haber cambios entre las pantallas de la aplicación de forma automática así como cambios en los elementos internos de la misma que harán que el usuario no se despiste.
- Se puede centrar la atención del usuario con el movimiento de alguno de los elementos de la pantalla, como en nuestro caso sucede en la aplicación del alumno en la cual el movimiento de algunos componentes de la pantalla le indican al niño que debe tarea debe llevar a cabo en ese momento.

Como se puede observa la mayoría de las características que se han puesto como ventaja a la hora de realizar la implementación en Flash están relacionadas directamente con la usabilidad, este es uno de los puntos más importantes por el que se escogió dicha tecnología y es que se puede hacer que la aplicación sea más usable de una manera más sencilla que en otras plataformas.

También se han planteado algunos problemas por haber escogido dicha tecnología, ya que por ejemplo al no ser una plataforma muy preparada para la Orientación a Objetos ha llevado un poco más de esfuerzo el seguir la metodología que se ha seguido a lo largo de todo el proyecto, este problema ya se indicó cuando se mostró el diagrama de clases en el capítulo anterior.

Otra ventaja que ha aportado Flash para ser elegido finalmente como plataforma en el desarrollo de esta herramienta ha sido el poder realizar conexiones remotas para hacer posible el funcionamiento del aula virtual, esto se ha podido realizar gracias a los socket de Flash y al manejo que tiene el mismo de los xml, ya que toda la parte de envío de mensajes entre las diferentes aplicaciones se lleva a cabo mediante el envío de ficheros xml y utilizando el objeto XMLSocket de Flash.

El único inconveniente que se tenía respecto a utilizar Flash es que éste dispone de la tecnología que implementa un socket desde el ordenador del cliente abriendo una conexión con el servidor, pero para poder utilizar el objeto XMLSocket se debe tener un servidor implementado en otra plataforma, ya que dicha tecnología no permite la realización de un servidor en Flash, es por eso que dicho servidor se implementó utilizando Turbo Delphi como se explicará en el punto siguiente.

A continuación se pasa a describir los componentes de Flash que se han utilizado en la implementación de las aplicaciones del alumno y del profesor y la correspondencia que podrían tener los mismos con los diferentes componentes del modelo abstracto que se planteó en el capitulo anterior.

5.3.1 Flash: Componentes utilizados en la implementación de la herramienta.

TextInput: TextInput es un componente de una sola línea que ajusta el objeto TextField nativo de ActionScript. Puede utilizar estilos para personalizar el componente TextInput; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo “disabledColor”. El componente TextInput se puede formatear también con HTML o como un campo de contraseña que disfraza el texto.

Este componente se correspondería, en el modelo de presentación abstracto, con un componente de tipo input. 

TextArea: TextArea es un componente para varias líneas, el componente TextArea ajusta el objeto TextField nativo de ActionScript. Puede utilizar estilos para

personalizar el componente TextArea; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo “disabledColor”. El componente TextArea también se puede formatear con HTML o como un campo de contraseña que disfraza el texto.

Este componente se correspondería en el modelo de presentación abstracto, con un componente, de tipo input. 

List: El componente List es un cuadro de lista de desplazamiento de selección única o múltiple. Una lista también puede mostrar gráficos, incluidos otros componentes.

Este componente se correspondería en el modelo de presentación abstracto, con un componente  , puede ser de tipo output, para mostrar información  , y de tipo control  ya que permite la realización de asociaciones.

Button: El componente Button es un botón rectangular de la IU que puede cambiarse de tamaño. También se le puede añadir un icono personalizado y cambiar su comportamiento de botón de comando a conmutador.

Este componente se correspondería en el modelo de presentación abstracto, con un componente, de tipo control. 

DataGrid: El componente DataGrid permite crear visualizaciones y aplicaciones de datos con muchas posibilidades. Se puede utilizar para crear instancias de un juego de registros mediante Macromedia Flash Remoting y mostrarlo en columnas. También se pueden utilizar datos de un juego de datos o de una matriz para definir un componente DataGrid.

Este componente se correspondería en el modelo de presentación abstracto, con un componente  , puede ser de tipo output, para mostrar información  y de tipo control  ya que permite realizar acciones asociadas al cambio de estado.

Alert: El componente Alert permite mostrar una ventana con un mensaje para el usuario y botones de respuesta. Esta ventana tiene una barra de título que se puede rellenar con texto, un mensaje que se puede personalizar y botones con etiquetas que se pueden cambiar. Una ventana Alert puede tener cualquier combinación de botones Sí, No, Aceptar y Cancelar.

Este componente se correspondería en el modelo de presentación abstracto, con un componente, de tipo output ya que muestra información. 

Loader: El componente Loader es un contenedor que puede mostrar archivos SWF o JPEG (pero no archivos JPEG *progresivos*). Se puede ajustar el tamaño del contenido del componente o cambiar su tamaño para que quepa el contenido. De forma predeterminada, el contenido se ajusta al componente Loader. También puede cargar el contenido en tiempo de ejecución y controlar el progreso de carga.

Este componente se correspondería en el modelo de presentación abstracto, con un componente, de tipo output ya que muestra información. 

UI ScrollBar: El componente UIScrollBar permite añadir una barra de desplazamiento a un campo de texto Puede añadir una barra de desplazamiento a un campo de texto durante la edición o en tiempo de ejecución con ActionScript.

Este componente se correspondería en el modelo de presentación abstracto, con un componente, de tipo control. 

Además de estos componentes, Flash tiene otros muchos más que no se van a explicar aquí y que pueden consultarse en el propio programa o en la página oficial [http://www.adobe.com/support/documentation/es/flash/ - userguides]. El resumen de las correspondencias que se han hecho entre los componentes de Flash utilizados y los componentes a los que corresponderían en el modelo de presentación abstracto se puede ver a continuación en la comparativa de la tabla 5.1.

Componente Flash	Componente modelo abstracto
TextInput	
TextArea	
List	
Button	
DataGrid	
Alert	

Loader	
UI ScrollBar	

Tabla 5.1 Componentes de Flash y los AIOs.

5.4 Turbo Delphi: implementación de la herramienta servidor

Para la implementación del servidor se ha utilizado Turbo Delphi que es un entorno de desarrollo creado por Borland®, que utiliza el lenguaje Delphi y que es gratuito [http://www.codegear.com/downloads/free/turbo].

La razón por la que se decidió utilizar esta plataforma después de sopesar varias opciones fue su sencillez, modularidad y que además dispone de muchos componentes que lo hacen idóneo para realizar aplicaciones del tipo que se necesitaba, es decir aplicaciones de servidor, pudiendo realizar dicha aplicación sin tener que llegar a hacer una programación a bajo nivel, además permite que se pueda hacer de manera visual, sin que suponga gran dificultad.

A continuación, al igual que se hizo en el punto anterior con Flash, se pasa a describir los componentes de Turbo Delphi que se han utilizado en la implementación del servidor y la correspondencia que podrían tener los mismos con los diferentes componentes del modelo abstracto que se planteó en el capitulo anterior.

5.4.1 Turbo Delphi: Componentes utilizados en la implementación del servidor.

TForm: Se trata de un componente "contenedor", lo que le permite albergar en su interior otros componentes. Aunque él en sí dispone ya de una cierta funcionalidad, realmente su utilidad se obtiene cuando en él se insertan otros componentes que sirvan para mostrar o solicitar información al usuario del programa.

Este componente se correspondería en el modelo de presentación abstracto, con un contenedor.

TListBox: Este componente sirve para albergar una amplia cantidad de opciones distintas entre las que seleccionar. En el caso de que el número de elementos exceda las dimensiones de la lista, en ésta aparecerán las barras de desplazamiento correspondientes. Los elementos contenidos en cada momento en la lista pueden ser gestionados mediante la propiedad *Items*.

Este componente se correspondería en el modelo de presentación abstracto, con un componente  , puede ser de tipo output, para mostrar información  , y de tipo control  ya que permite la realización de asociaciones.

 **TMemo:** Este componente se utiliza cuando se tiene que mostrar un texto que puede ser más o menos extenso, puede tratarse de forma casi idéntica a un control *Edit*, con la diferencia de que es posible trabajar con una mayor extensión de texto, que puede estar distribuido en múltiples líneas.

Este componente se correspondería en el modelo de presentación abstracto, con un componente, de tipo input. 

 **TTcpServer:** Se utiliza para crear aplicaciones de servidor de tipo TCP, se puede utilizar junto con el *TTcpClient* para desarrollar un cliente-servidor.

Este componente se correspondería en el modelo de presentación abstracto, con un contenedor. 

Además de estos componentes Turbo Delphi dispone de otros muchos más para poder ser utilizados. Del mismo modo que se hizo en el punto anterior, a continuación se muestra una comparativa entre ambos tipos de componentes en la tabla 5.2.

Componente Turbo Delphi	Componente modelo abstracto
TForm	
TListBox	
TMemo	
TTcpServer	

Tabla 5.2 Componentes de Turbo Delphi y los AIOs.

5.5 Intercambio de datos entre las aplicaciones

Tras explicar como se ha implementado cada una de las aplicaciones que componen eAula solo queda indicar como se realiza el intercambio de datos entre ellas. La comunicación se realiza mediante mensajes en XML. Se eligió este formato para los

mensajes porque es muy fácil de manejar tanto en Turbo Delphi en el servidor como para Flash en el alumno y el profesor.

Las actividades se encuentran almacenadas en el servidor siguiendo una estructura determinada, cada actividad se compone de un xml donde se indica que tipo de actividad es y los datos que contiene, y de una carpeta donde se encuentran los datos de la actividad en concreto, la carpeta debe tener el mismo nombre que el xml y dentro de ella tendrá dos carpetas, una para las imágenes que se llamará img y otra para el audio que se llamará aud. A continuación se pueden observar unos ejemplos de la estructura que debe tener una actividad (figura 5.2.), así como de un xml de puzzle (figura 5.3.) y de un xml de asociación (figura 5.4.).

Figura 5.2 Estructura de una actividad.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<actividad tipo="puzzle">
        <titulo>El rey león</titulo>
        <imagen nombre="rey_leon.png" piezas="12" imprimible="rey_leon.pdf"/>
</actividad>
```

Figura 5.3 XML puzzle.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<actividad tipo="asociacion">
<titulo>Parejas de animales</titulo>
<conjunto1 tipo="">
        <elemento>Toro</elemento>
        <elemento>Caballo</elemento>
        <elemento>Gato</elemento>
        <elemento>Gallo</elemento>
</conjunto1>
<conjunto2 tipo="">
        <elemento>Vaca</elemento>
        <elemento>Yegua</elemento>
        <elemento>Gata</elemento>
        <elemento>Gallina</elemento>
</conjunto2>
</actividad>
```

Figura 5.4 XML asociación.

5.6 Funcionamiento de eAula

En este punto se va a describir cual es el funcionamiento de eAula de una manera general, ya que en puntos posteriores dentro de este mismo capitulo se darán unos manuales de las aplicaciones más completos, tal y como se indicó al principio de este capítulo.

Se ejecuta la aplicación del servidor, dentro del cual debe haber al menos una actividad para que se pueda trabajar con ella.

El profesor se conecta al sistema introduciendo su nombre, una vez hecho esto la aplicación envía al servidor los datos del profesor (nombre, IP), de esta forma el profesor queda dado de alta dentro del sistema.

Al igual que en el caso del profesor, el alumno se conecta al sistema introduciendo su nombre (debido a la corta edad de los alumnos, será el profesor el encargado de introducir el login del usuario en la aplicación), una vez hecho esto la aplicación envía al servidor los datos del alumno que se acaba de conectar al sistema (nombre, IP), de esta forma el alumno queda dado de alta dentro del sistema. Inmediatamente después el servidor envía un mensaje al profesor diciéndole que usuario se acaba de conectar, de este modo la aplicación del profesor identifica al alumno y lo añade a su “lista de alumnos conectados”. Si cuando se conecta el profesor ya había conectados alumnos antes de que él lo hiciera, nada más conectarse se le rellenará la “lista de alumnos conectados” con los que ya estaban conectados al sistema, y posteriormente se le irán añadiendo a esa lista los nuevos alumnos que se vayan conectando.

Cuando el profesor entra en el sistema, además de rellenarse la “lista de alumnos conectados”, se le rellena de manera automática la “lista de actividades disponibles”, la cual podrá actualizar cuando quiera, para lo que tendrá que mandar una petición al servidor para que éste le diga las actividades que hay disponibles en cada momento (las actividades se encontraran almacenadas en el servidor, la aplicación encargada de crear actividades las deberá haber dejado previamente en el servidor siguiendo una estructura determinada). El servidor contesta al profesor enviándole el conjunto de actividades de las que se dispone.

Una vez el profesor sabe los alumnos que tiene conectados y las actividades que tiene disponibles ya puede asignarle a un alumno determinado una determinada

actividad, para ello en su lista de alumnos selecciona el alumno o alumnos de la lista disponible a los que va dirigida la orden y selecciona de la lista de actividades aquella o aquellas que quiere que dichos alumnos realicen. Una vez seleccionadas le envía al servidor la relación alumno(s)-actividad(es).

La aplicación del alumno se encuentra a la espera de recibir las actividades a cargar, una vez que las recibe las carga de manera automática, por lo que el alumno ya puede realizar las actividades que a partir de ese momento tiene disponibles en su aplicación. Al mismo tiempo que el servidor envía al alumno la tarea a realizar le envía también al profesor el estado del alumno, para que así pueda controlar en todo momento que alumno está realizando alguna actividad y cual está esperando recibir una tarea.

Conforme va terminando las actividades la aplicación del alumno envía al servidor un mensaje indicándole la actividad que ha realizado y los aciertos que ha tenido en dicha actividad, así como de que alumno se trata.

Una vez que el servidor recibe esa información, la envía al profesor que podrá seguir a tiempo real la evolución de los alumnos, así como volver a enviar a un alumno determinado una actividad determinada si cuando recibe los resultados parciales se da cuenta de que debería repetirla. Al igual que envía al profesor los resultados parciales, los almacena en el histórico formando parte así de los resultados generales, lo que podrán ser consultados en cualquier momento por el profesor si así lo desea, además de consultar los resultados generales también podrá borrarlos.

5.7 Aportaciones de eAula: JClic vs. eAula

Después de explicar el funcionamiento de eAula, en este punto se compara su funcionamiento con una de las herramientas de las que se hablaba en el capitulo 2.

JClic es una herramienta con la que se pueden realizar tanto actividades de asociación como puzzles, sopas de letras, crucigramas o actividades de texto; de estos tipos de actividades en eAula se tienen las asociaciones y los puzzles, al considerar que son las que mejor se adaptan al rango de edades que nos ocupa.

La principal diferencia que encontramos entre eAula y JClic es que mientras que JClic es una herramienta de autoaprendizaje, eAula pretende ser una herramienta de aprendizaje asistido. Es decir, con JClic yo descargo en mi ordenador un determinado número de actividades y las voy realizando en el orden que considero oportuno,

mientras que se van generando unos informes de resultados con los que puedo medir mi curva de aprendizaje, esta es la primera pega que se encontró a JClic al probarlo, si se tiene en cuenta que en el proyecto que nos ocupa consiste en realizar una herramienta de aprendizaje para unos usuarios de edades comprendidas entre los 3 y los 5 años. Con eAula lo que tenemos es que el niño esta sentado delante de su ordenador con la aplicación arrancada y en el momento que el profesor considera oportuno le envía una tarea a realizar, esa tarea puede constar de una o varias actividades, el niño una vez que le llegan las actividades puede empezar a realizarlas y conforme las vaya finalizando los resultados se envían de manera automática al profesor, por lo que el niño no tiene que preocuparse por los resultados y el profesor no tiene que preocuparse de ir niño por niño viendo que tal les ha salido una determinada actividad. Aunque en las edades que nos ocupa es bastante común que el profesor vaya a ocuparse personalmente de cada niño, el que pueda enviarles de una manera automática las tareas que debe realizar facilita la enseñanza durante el transcurso de una clase; por ejemplo si en un aula tenemos que hay algún alumno que va más lento que el resto o que le cuesta más hacer las cosas que a sus compañeros con el uso del aula virtual hacemos que cada uno de los niños avance al ritmo que pueda sin tener que esperarse a que sus compañeros acaben, en el caso de ser más rápido haciendo sus tareas, o sin tener la presión de que le están esperando en el caso de ser él quien retrasa al resto. Con el aula virtual cada niño va a su ritmo realizando las tareas que le ha enviado el profesor a su terminal de una manera personalizada y enviando de manera transparente para él los resultados de las actividades que va realizando al profesor, agilizando por tanto el transcurso de la clase. Así mismo, en JClic nos encontramos con que un alumno puede saltarse las actividades sin ningún problema dando al botón de “siguiente actividad”, mientras que en eAula al tratarse de un aprendizaje guiado no se puede saltar ninguna actividad, realizando de este modo todas aquellas actividades que le han sido encomendadas por el profesor.

Otro aspecto que se intentó mejorar del JClic es que al ser una aplicación off-line, los resultados que el niño obtiene se almacenan en una máquina determinada, por lo que cada niño debería sentarse en el mismo ordenador cada vez que ejecutase el programa y no se debería sentar más de un niño por puesto informático para que los resultados obtenidos en la evaluación que la aplicación hace del aprendizaje del niño fuese real. Estas condiciones cuando nos referimos a un colegio no son nada realistas ya

que es bastante lógico que exista el aula de informática donde se realizarán unas determinadas actividades, puesto que toda la educación del niño, y más teniendo en cuenta el rango de edades con el que estamos tratando, no se puede llevar a cabo delante del ordenador; por lo que esa aula de informática sería compartida por varios alumnos, de incluso varios cursos y por varios profesores; por lo que al final no tendríamos un seguimiento real de los avances del niño. Sin embargo, con eAula esto no sucedería, ya que está preparada para esta situación y lo que hace es ir almacenando los resultados de cada uno de los alumnos de una manera univoca de tal forma que cada vez que un niño se siente en un puesto informático, independientemente de si fue en ese puesto en el que se sentó la última vez, y realice unas determinadas actividades, sus resultados llegaran al profesor indicándole que alumno es y que actividad ha realizado, así como los aciertos que ha tenido en dicha actividad, por lo que el profesor podrá llevar un control real de los resultados que ha obtenido cada uno de sus alumnos.

Otra cosa que se ha intentado mejorar en eAula es la interfaz de la aplicación, ya que en JClic nos encontramos con muchos botones que pueden liar al niño, además debe realizar varios pasos para cargar el paquete de actividades, teniendo que moverse por el disco duro del ordenador o el soporte donde se hallen las actividades, mientras que en eAula las actividades simplemente se ejecutan en la aplicación del alumno sin que él tenga que hacer nada, y en cada momento se tienen únicamente los botones necesarios para realizar las tareas pertinentes. Del mismo modo en JClic además de tener muchos botones nos encontramos con que un botón importante como es el de apagar y encender el sonido no está debidamente remarcado (figura 2.6, en azul) mientras que en eAula está en un lugar donde es más fácil de localizar.

En este proyecto se han intentado mejorar los aspectos de JClic que no parecían adecuados para la temprana edad en la que se ha centrado, como los que se han citado anteriormente.

5.8 Manual de usuario (Servidor)

En este punto se va ha describir el funcionamiento de la aplicación servidor, una vez que se ejecuta la aplicación nos encontramos con la pantalla de la figura 5.6., donde inicialmente aparecerán todas las pantallas vacía, la única que puede contener datos es la de arriba a la derecha, ya que en esa pantalla se muestran los datos del

histórico, por lo que si hay datos guardados se mostrarán nada más iniciarse la aplicación.

Además de tener ejecutada la aplicación, en la máquina donde se encuentre el ejecutable del servidor debe haber una carpeta que se llame www y dentro a su vez otra carpeta que se llamará actividades, donde se encontrarán almacenadas las diferentes actividades, que habrán sido puestas ahí previamente por la aplicación de edición de actividades y que deberán seguir la estructura que se indicó en la figura 5.2., dicha estructura queda reflejada en la figura 5.5. que se muestra a continuación.

Por último se debe tener un archivo xml donde se encuentre la aplicación del profesor y donde se encuentre cada una de las aplicaciones de los alumnos, ese archivo se llamará "direccionserver.xml" y contendrá la dirección del servidor, para que de este modo si el servidor cambia de máquina, solo se deberá actualizar ese fichero y seguirá funcionando. En dicho fichero pondrá lo siguiente: "*<?xml version="1.0" encoding="iso-8859-1"?><direccionserver>dirección IP de la máquina donde se encuentre ejecutándose el servidor</direccionserver*".

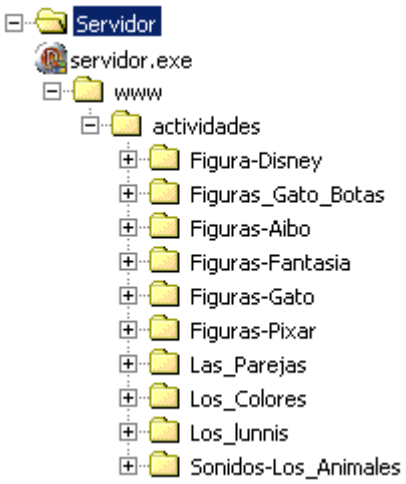

Figura 5.5 Estructura de carpetas del servidor.

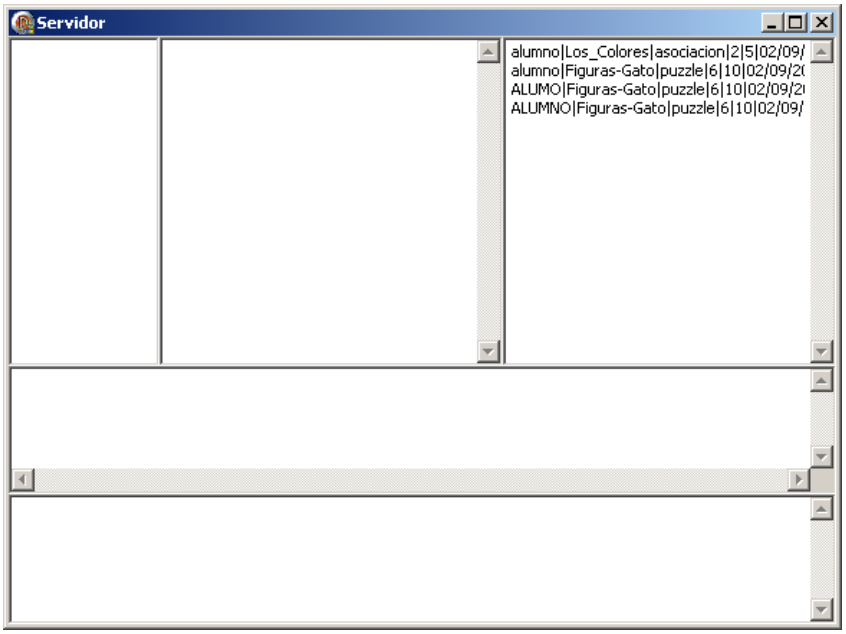

Figura 5.6 Captura de la pantalla inicial del servidor.

Una vez se vayan conectando el profesor y los alumnos al sistema, en el servidor irán apareciendo sus direcciones IP, así como las peticiones que se le hacen de archivos por el puerto 80 y las peticiones xml que se le van haciendo; después de un rato en funcionamiento la pantalla del servidor deberá tener aproximadamente el aspecto que se muestra en la figura 5.7.

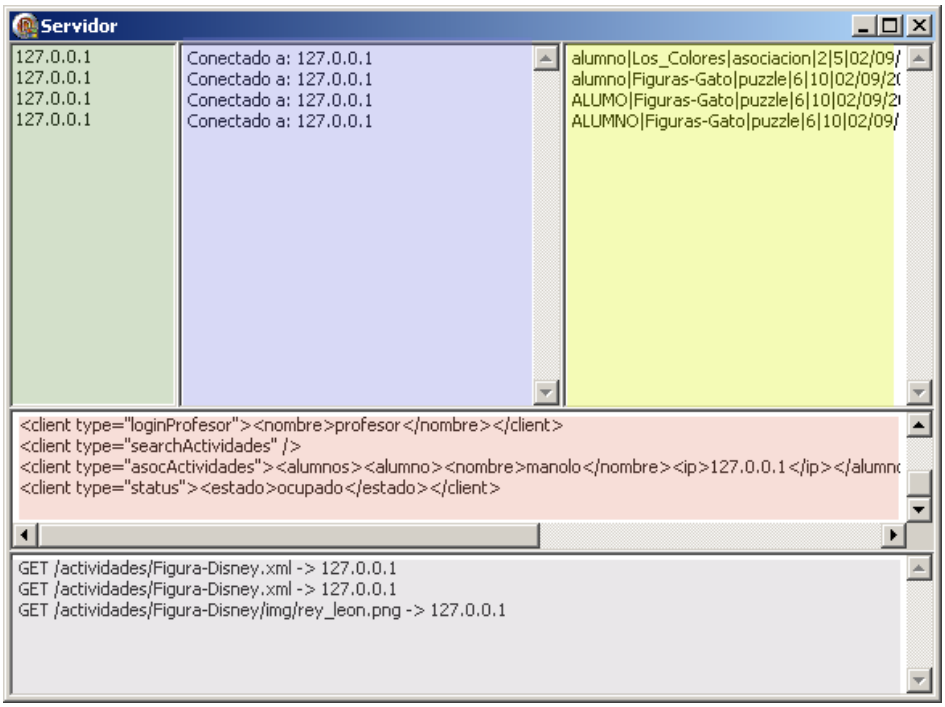

Figura 5.7 Captura de la pantalla del servidor en funcionamiento.

Se ha aprovechado la figura 5.7 para además de para mostrar el aspecto del servidor en funcionamiento para indicar que se muestra en cada una de sus partes.

En la parte pintada en verde aparecerán las direcciones de todos los usuarios que estén conectados, en la figura 5.7. aparece siempre la misma dirección porque se han ejecutado todos los clientes en la misma máquina, del mismo modo en la pantalla pintado en azul irán apareciendo las conexiones y desconexiones que se van realizando (la IP de las mismas), esta pantalla aunque parezca redundante teniendo la anterior no lo es, ya que en la pantalla pintada en verde cuando un usuario se desconecta simplemente desaparece de la lista, mientras que en la pantalla en azul aparecerá como que ese usuario se ha desconectado del sistema.

En la pantalla pintada en amarillo en la figura 5.7, aparecerá el histórico tal y como se ha indicado más arriba, esta pantalla aparecerá en vacío únicamente cuando el histórico no tenga datos y se irá actualizando en tiempo real cuando nuevos datos sean introducidos.

En la pantalla en rojo se muestran los mensajes en xml que le llegan al servidor, y en la pantalla gris aparecen las peticiones de archivos que se hacen al servidor.

5.9 Manual de usuario (Profesor)

En este punto se va a explicar paso a paso el funcionamiento de la aplicación del profesor desde el punto de vista del usuario, es decir, de que forma se debe utilizar.

Al ejecutar la aplicación lo primero que se encuentra el usuario es con la pantalla de identificación mostrada en la figura 5.8.

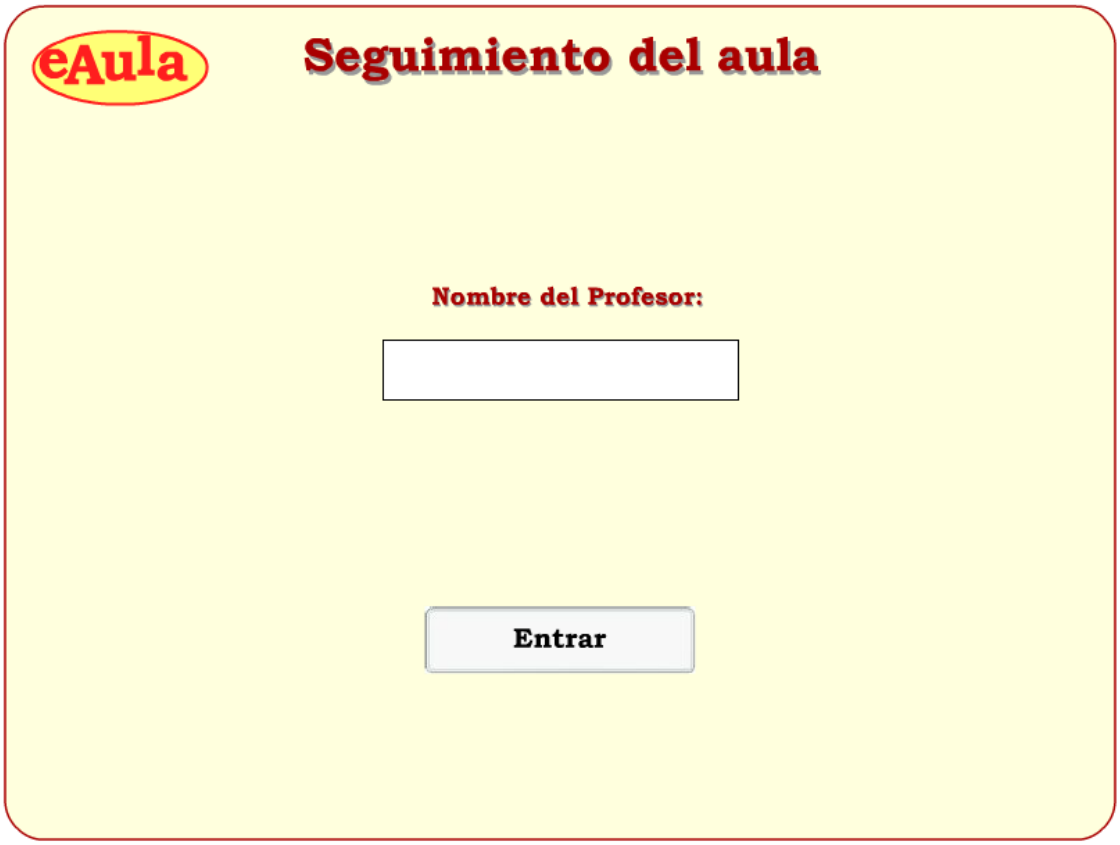

Figura 5.8 Captura de la pantalla de identificación del profesor.

Una vez que el profesor introduzca su nombre y pulse a entrar pasará a la siguiente pantalla de la aplicación, si se pulsa sobre entrar sin haber introducido los datos aparecerá un mensaje de error avisando que se debe introducir el nombre antes de pulsar entrar, este mensaje se muestra a continuación en la figura 5.9.

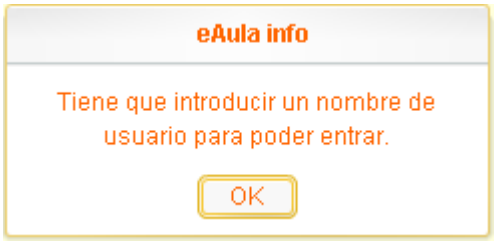

Figura 5.9 Captura mensaje de error (sin introducir nombre).

Una vez que se entra en la aplicación aparece la pantalla que se muestra en la figura 5.10.

Figura 5.10 Captura de la aplicación del profesor.

En la pantalla de la figura 5.10. se le marcan al profesor los pasos que debe seguir para enviar una o varias actividades a los alumnos, primero debe elegir la o las actividades que desea enviar (1), después debe elegir el o los alumnos a quien se las va a enviar (2), si intenta seleccionar los alumnos antes que las actividades la aplicación no le dejará que lo haga, teniendo en cuenta la usabilidad de la que se lleva hablando a lo largo de toda la memoria, hasta que no elige al menos una actividad los alumnos permanecen inactivos, como se puede observar en la figura 5.10. del mismo modo que no se le permite pulsar el botón enviar.

Después de elegir la o las actividades y el o los alumnos debe pulsar sobre el botón de enviar y en ese momento las actividades llegaran a los correspondientes alumno y en la aplicación del profesor aparecerán como ocupados los alumnos a los que le haya enviado actividades, tal y como puede verse a continuación en la figura 5.11.

Figura 5.11 Cambio de estado del alumno.

En la figura 5.11., se puede observar como ya no están bloqueados los alumnos, puesto que el profesor ha seleccionado una actividad y como el estado del alumno al que se la ha mandado a pasado de esperando a ocupad (marcado en rojo), si ahora el profesor intentase volver a enviar una actividad al mismo alumno le aparecería un mensaje de error indicándole que dicho alumno está ocupado, el mensaje de error se muestra en la figura 5.12.

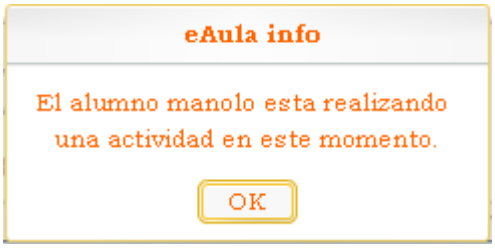

Figura 5.12 Mensaje de error de la aplicación del profesor.

Como el profesor no tiene por qué tener un conocimiento previo del funcionamiento de la aplicación, si el que los pasos estén numerados no es suficiente para que entienda el funcionamiento, puede pulsar sobre el botón de ayuda y le aparecerá la pantalla que se muestra a continuación en la figura 5.13. y que describe los pasos a realizar para el envío de actividades.

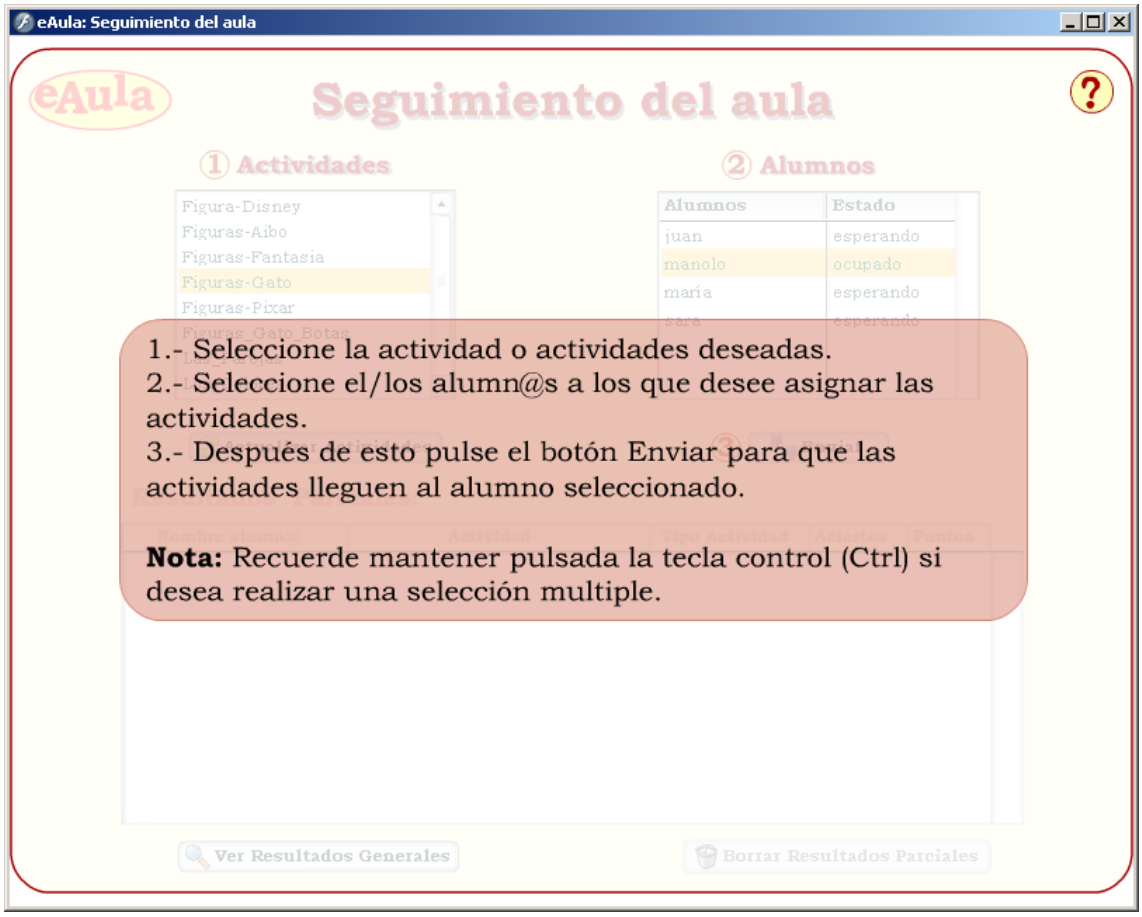

Figura 5.13 Pantalla de ayuda de la aplicación del profesor.

Por último solo queda explicar el manejo de los resultados por parte del profesor, ya que además de consultarlos, puede borrarlos. Se empezará por los resultados parciales que se le van actualizando en tiempo real cada vez que un alumno termina una

determinada actividad, estos resultados se le irán colocando en la parte de debajo de la pantalla de la aplicación, tal y como muestra la figura 5.14, estos resultados pueden aparecer en tres colores diferentes; si el alumno a completado la actividad muy bien y tiene un 9 o un 10 en la nota de la misma, sus resultados aparecerán en verde, si por el contrario ha obtenido una puntuación de 3 o menos sus resultados aparecerán en rojo para que el profesor pueda darse cuenta inmediatamente y actuar en consecuencia volviendo a enviar a dicho alumno la misma actividad para que mejore sus resultados; si los resultados obtenidos por el alumno no están en ninguno de estos dos casos aparecerán en negro. Además el profesor puede en cualquier momento vaciar la lista de resultados parciales pulsando sobre “Borrar los resultados parciales”.

Figura 5.14 Captura de la aplicación del profesor (resultados parciales).

Por último, queda explicar lo referente a los resultados generales; en cualquier momento el profesor puede consultar los resultados generales del aula, es decir el histórico de todos los resultados de los alumnos que se va almacenando en tiempo real

en el servidor, para conseguir esto solo tiene que pulsar sobre “Ver Resultados Generales” y aparecerá la pantalla que se muestra en la figura 5.15. Dichos resultados puede ordenarse por cada una de sus columnas (en la figura 5.15. se encuentran ordenados por nombre de la actividad) y los puede borrar en el momento en que lo desee pulsando sobre “Borrar Resultados Generales”, pero debe tener en cuenta que una vez borrados no existe una copia de los mismos.

Figura 5.15 Captura de la aplicación del profesor (resultados generales).

5.10 Manual de usuario (Alumno)

En este punto se va a explicar paso a paso el funcionamiento de la aplicación desarrollada desde el punto de vista del alumno desde, es decir, al igual que se hizo con el profesor en el punto anterior aquí se va a explicar cómo debe utilizar la aplicación el alumno.

Como indicaciones generales para toda la aplicación se quiere comentar desde el principio que todos los botones de la misma, al mantener el ratón sobre ellos, informan a través de voz de una frase que describe para lo que sirven, del mismo modo cada vez que se entra en una actividad también se ejecuta una locución que dice al alumno lo que debe hacer para resolver la actividad.

Al ejecutar la aplicación lo primero que se encuentra el usuario es con la pantalla de identificación mostrada en la figura 5.16.

Figura 5.16 Captura de la pantalla de identificación del alumno.

Una vez que el alumno introduce su nombre (se debe tener en cuenta que algunos niños pueden ser muy pequeños, en estos casos será el profesor quien introduzca en la aplicación el nombre del niño) y pulse al botón etiquetado con “entrar” pasará a la siguiente pantalla de la aplicación, si se pulsa sobre el botón etiquetado con “entrar” sin haber introducido los datos aparecerá un mensaje de error avisando que se debe introducir el nombre antes de pulsar entrar, este mensaje de error sería el mismo que le aparece al profesor en circunstancias similares y que se muestra en la figura 5.9. Una vez que se entra en la aplicación aparece la pantalla que se muestra a continuación en la figura 5.17.

Figura 5.17 Captura de la pantalla “en espera” del alumno.

La pantalla mostrada en la figura 5.17 será la que vea el alumno siempre que se encuentre en estado de espera, es decir, cuando esté esperando que se le encomiende alguna tarea, esta pantalla está formada por un par de animaciones cuya función es además de informar al alumno (en las letras que van pasando pone “esperando datos del profesor”) mantenerlo atento a la pantalla, esta segunda parte se consigue con la

animación del muñeco que duerme. Además al entrar en esta pantalla, por si el niño es demasiado pequeño para leer el mensaje, dicho mensaje se le se le facilita a través de un sonido.

En el momento en que la aplicación del alumno recibe alguna actividad del profesor, la aplicación pasa automáticamente a mostrar esa actividad, si solo es una, o la primera del grupo de actividades cuando hay más de una. En este punto, el alumno se puede enfrentar a dos pantallas diferentes, la correspondiente a un puzzle y la correspondiente a una asociación.

A continuación se van a explicar cual es el funcionamiento de los dos tipos de actividad, empezando por el puzzle, al entrar en un puzzle el alumno se encontrará con la pantalla que aparece en la figura 5.18.

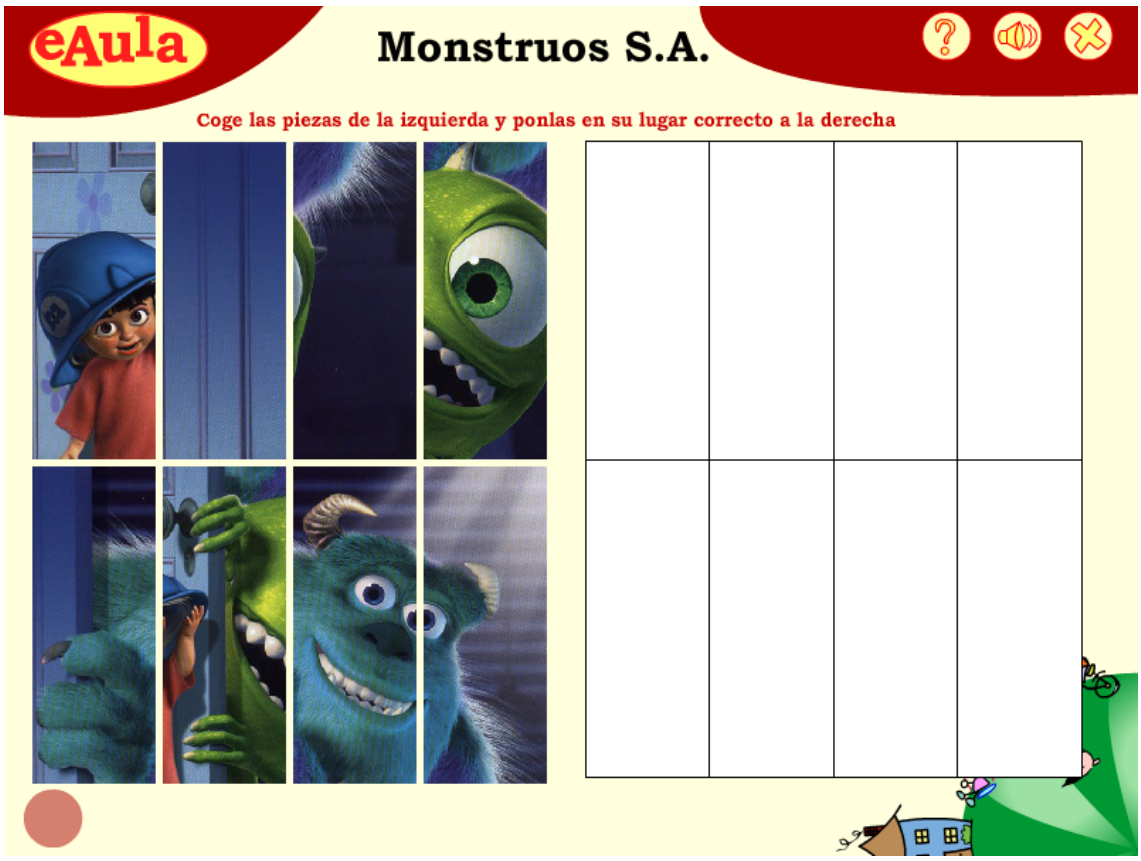

Figura 5.18 Captura de la pantalla de un puzzle (estado inicial).

En la pantalla mostrada en la figura 5.18 el alumno tiene la pantalla dividida principalmente en tres secciones, la primera sección de la pantalla del puzzle sería la de la izquierda donde se encuentran todas las piezas del puzzle que se debe resolver, el

número de piezas depende de la dificultad que se le quiera poner al puzzle y varia pudiendo ser 6,8,9,12 ó 16. Estas piezas se deben coger una a una y arrastralas sobre las cajas en blanco de la derecha (que serían la segunda sección del puzzle), dichas piezas se quedarán donde se coloquen siempre y cuando se hayan colocado en su posición correcta, ya que si no volverán automáticamente al lugar donde estaban antes de cogerlas, por último la tercera sección de la pantalla del puzzle sería donde se encuentra la botonera, en la parte superior, con esos botones el alumno puede anular las locuciones (al entrar en cada actividad se informa mediante sonido en qué consiste la misma), salir de la aplicación o mostrar la ayuda. Al pulsar sobre la ayuda se le mostrará en la parte de la derecha el puzzle resuelto, tal y como se puede observar en la figura 5.19.

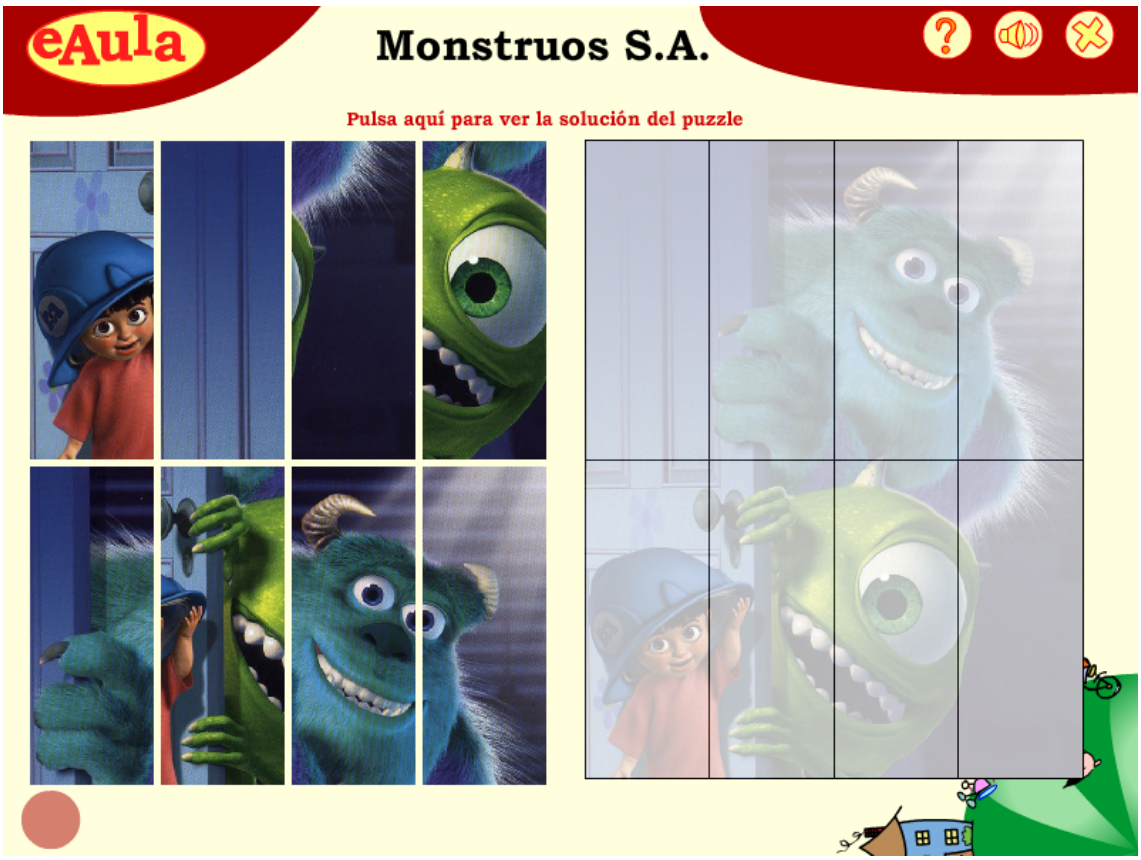

Figura 5.19 Captura de la pantalla de un puzzle (mostrando ayuda).

La ayuda que se muestra en la figura 5.19 solo es visible mientras el alumno está pulsado el botón de la ayuda, una vez que lo suelta se vuelven a ver las casillas en blanco, ya que si no sería muy sencillo que lo resolviese.

Una vez que termina el puzzle de manera correcta el circulo rojo situado en la parte inferior izquierda se convierte mediante una animación en una barra que le muestra dos botones, uno con el que puede imprimir la imagen del puzzle que acaba de resolver para colorearlo y otro con el que puede pasar a la siguiente actividad. Esta barra se utiliza siguiendo los criterios ergonómicos de los que se habló en el capitulo 3 para focalizar la atención del niño e indicarle qué debe hacer a continuación, y a su vez muestra en cada momento los botones que necesita. En la figura 5.20 se muestra el puzzle ya terminado con la barra que se ha indicado antes y en la figura 5.21 se muestra el dibujo que obtendría el niño pulsado sobre el botón de imprimir.

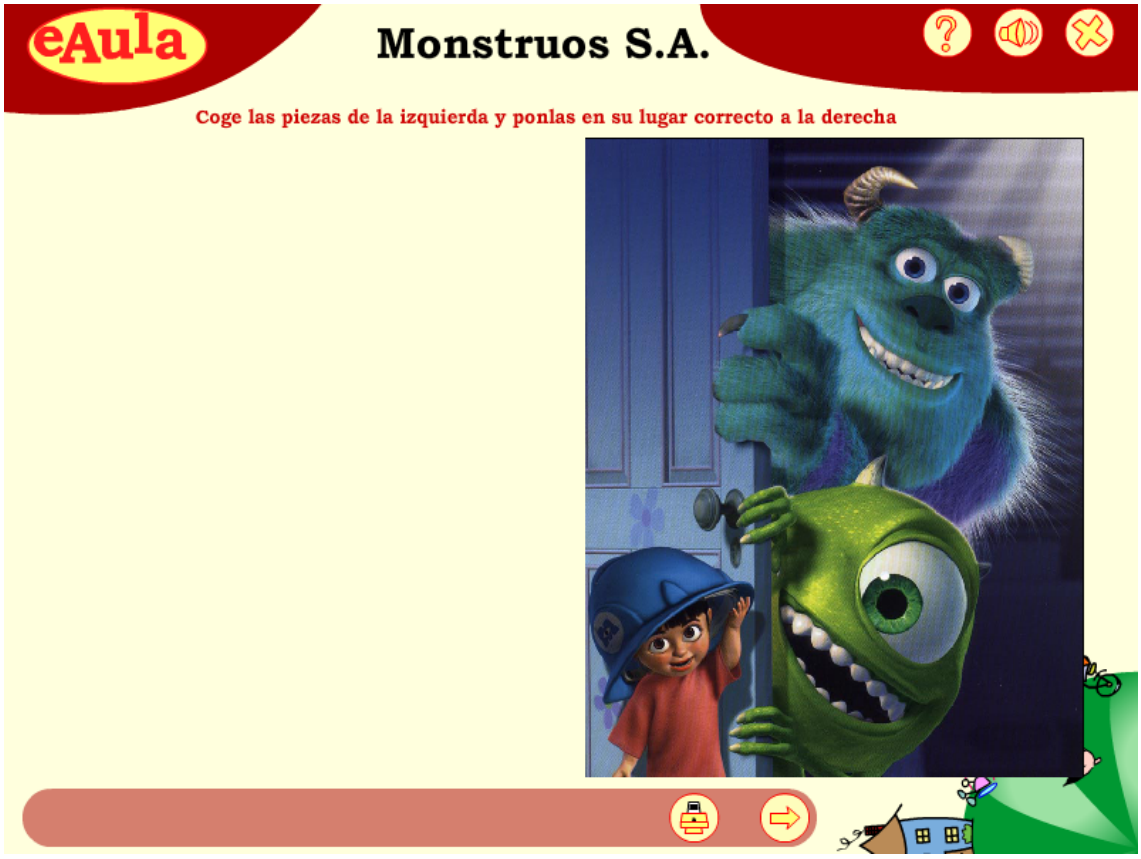

Figura 5.20 Captura de la pantalla de un puzzle (puzzle resuelto).

Figura 5.21 Lámina para colorear.

Si la actividad en lugar de ser un puzzle es una asociación, el alumno al entrar se encontrará con la pantalla que se muestra a continuación en la figura 5.22.

Figura 5.22 Captura de la pantalla de una asociación (estado inicial).

En la pantalla que se muestra en la figura 5.22. se muestra el estado inicial de una asociación y al igual que sucedía con el puzzle se puede dividir en 3 secciones. La primera sección sería la de los elementos de la izquierda donde se tienen en este caso unas imágenes de unos altavoces y unos numeritos en un círculo, dicho circulo cambia de color cuando se coloca el ratón encima de él (como se puede ver en la figura 5.22 en el número 2). Esta actividad consiste en arrastrar los círculos de la izquierda sobre su correspondiente pareja de los de la derecha. Los elementos de la derecha, en este caso imágenes, formarían la segunda sección de la actividad. Tanto los elementos de la izquierda como los de la derecha puede ser imágenes, texto o sonido (en cuyo caso el alumno deberá pulsar sobre el altavoz para que se emita el sonido que corresponda).

Una vez que el alumno ha arrastrado todos los círculos de la izquierda sobre los de la derecha, el circulo situado en la parte inferior izquierda, al igual que pasaba en el puzzle, mediante una animación se coloca en el centro de la pantalla y muestra el botón de chequeo de la actividad, el cual al ser pulsado por el alumno corregirá la actividad

que se acaba de realizar. A continuación en la figura 5.23. se muestra la pantalla cuando aparece dicho botón.

Figura 5.23 Captura de la pantalla de una asociación (botón corrección).

Una vez que el alumno pulsa sobre el botón de corregir (figura 5.23) en los círculos de la derecha aparecerán en verde los aciertos y en rojo los fallos y además la barra se desplazará de nuevo par mostrar el botón de ir a la siguiente actividad. Dicho cambio se puede ver a continuación en la figura 5.24.

Figura 5.24 Captura de la pantalla de una asociación (terminada).

Por último solo queda indicar que tanto en el puzzle como en la asociación si el alumno pulsa sobre el botón de cerrar, el alumno pasa a la pantalla de salida donde se encuentra con dos botones animados donde debe confirmar que desea salir, pulsando sobre “si” o por el contrario si no desea abandonar la aplicación deberá pulsar sobre “no”. Si se pulsa el botón de cerrar a mitad de realizar una actividad y después se pulsa “no” el alumno volverá exactamente a donde dejó la actividad, esta pantalla de salida se muestra en la figura 5.25.

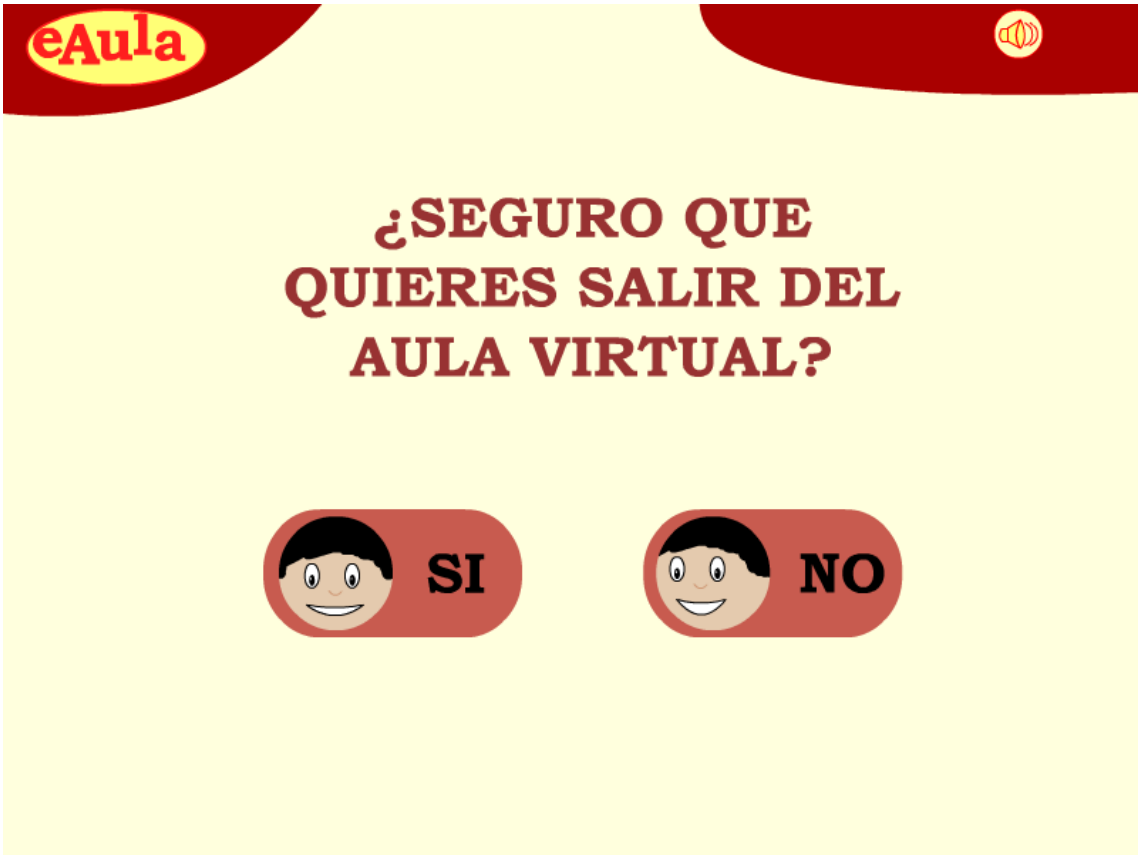

Figura 5.25 Captura de la pantalla de salida de la aplicación del alumno.

5.11 Conclusiones

En este capitulo se ha mostrado cómo, partiendo de las especificaciones que se hacían en el capitulo 4, se ha llegado a la implementación de la aplicación, aunque se debe tener en cuenta que tal y como se dijo en el primer punto de este capitulo, el paso entre la especificación hecha en el capitulo 4 y la implementación que se ha realizado en este capitulo 5 debería hacerse de forma automática o semiautomática, si tenemos en cuenta la metodología, mostrada en la figura 4.1., y que se ha seguido a lo largo de todo el desarrollo del proyecto. Lo ideal sería que después de haber hecho un modelado de la aplicación de forma abstracta y después de haber concretado las características de la misma mediante el uso de la Ingeniería del Software se pudiese llegar a una implementación sino total por lo menos parcial de lo desarrollado en las fases anteriores a la fase de implementación.

Además de mostrar cómo se ha llegado a la implementación de la aplicación en este capítulo, se han expuesto diversos manuales para explicar el funcionamiento de cada una de las partes que componen eAula haciendo hincapié en aquellos eventos que

se producen en la misma y que están relacionados directamente con las técnicas IPO que se propusieron en el capitulo 3 de esta memoria, ya que se han tenido bastante en cuenta las mismas, sobre todo en la fase de implementación que es a la que se refiere en este capitulo.

Las técnicas IPO que se propusieron en el capitulo 3 de esta memoria han sido de gran utilidad a la hora de llevar a cabo la implementación de la aplicación, sobre todo teniendo en cuenta que va dirigida a niños por lo que se debe prestar una gran atención a la interfaz y alo que con ella el usuario puede hacer y cómo puede hacerlo, y esto no es un tema que se trate a nivel de usuario desde la Ingeniería del Software o por lo menos no tan en profundidad como la aplicación requería.

CAPÍTULO 6 CONCLUSIONES Y TRABAJO FUTURO

En este último capítulo de la memoria se van a indicar las conclusiones generales que han quedado tras la realización del proyecto, teniendo en cuenta todos los pasos por los que ha seguido desde su inicio hasta su finalización.

Se debe indicar, en primer lugar, que los objetivos que se plantearon al inicio se han cumplido de manera satisfactoria ya que se ha logrado llevar a cabo la idea que se propuso inicialmente y se han cubierto todas las metas que se plantearon en los capítulos 1 y 2 de esta memoria, es decir se ha conseguido la realización de un aula virtual compuesta de tres aplicaciones que interactúan entre sí de manera transparente para el usuario y que tienen en cuenta las necesidades de los mismos.

También se debe señalar que dicho objetivo se ha conseguido aplicando la metodología que se propuso inicialmente, aunque no se hayan seguido todas las fases de manera estricta, tal y como se ha indicado en capítulos anteriores. Aunque esto se debe, fundamentalmente a que en la actualidad no existen los medios para pasar automáticamente desde las fases iniciales a la fase final de dicha metodología.

Por último destacar, que para que se haya podido llevar a cabo el proyecto se han utilizado técnicas y métodos propuestos que se propusieron inicialmente, se han cubierto las necesidades funcionales de la aplicación utilizando técnicas de Ingeniería del Software y las necesidades de los usuarios de la aplicación se han considerado, a su vez, utilizando técnica de Interacción Persona-Ordenador.

6.1 Conclusiones

Tras llevar a cabo la realización de este proyecto y después de la información que se ha consultado para su realización y del trabajo que se ha realizado se han obtenido varias conclusiones que se van a exponer a continuación:

Tal y como se indica al principio de esta memoria hoy en día la informática es un mundo con el que prácticamente todos tienen contacto alguna vez en su vida, cada vez este contacto se tiene a más tierna infancia y cada vez más la informática es necesaria para los quehaceres de la vida cotidiana, ya que hoy en día casi todo se mueve utilizando un ordenador para ello, por lo que cuanto antes se empiece a familiarizar una persona con el mundo de los ordenadores, más fácil le será después convivir con ellos, ya que es bastante probable que en su futuro tenga que utilizarlos.

Con el aula virtual lo que se consigue es que los niños empiecen desde los tres años a utilizar un ordenador con la misma normalidad con la que empiezan a utilizar un lápiz o un bolígrafo ya que esto les será de utilidad en un futuro.

A esto se le ha añadido que la mejor manera de que un niño en el rango de edad al que está dirigido este proyecto aprenda es mediante el juego, esta idea está respaldada por la L.O.E. que así lo indica en su currículo infantil, el cual se puede consultar en esta URL MEC [http://www.mec.es/educa/jsp/plantilla.jsp?id=9&area=sistema-educativo .].

Por lo tanto, en este proyecto se han unido las necesidades de que el niño se familiarice con el ordenador con la necesidad del aprendizaje mediante el juego creando para ello un aula virtual donde el niño aprende mediante juegos en un ordenador.

Para llevar a cabo la realización de este aula se han tenido en cuenta las necesidades de los niños con un rango de edad entre los 3 y los 5 años, haciendo que la interfaz de la aplicación sea usable para ellos, logrando de este modo que de una manera sencilla y divertida para ellos aprendan sin mucho esfuerzo.

También cabe destacar que ha sido de gran utilidad para la realización de la aplicación el haber tenido en cuenta desde el principio del desarrollo software la usabilidad, tal y como se indicaba en el capitulo 3 que se debía hacer, además también desde el principio se han utilizado técnicas de Ingeniería del Software que han facilitado llevar a cabo la realización del mismo desde el punto de vista funcional y aunque el desarrollo Flash no parece segur esta estela.

6.2 Trabajo futuro

Como se ha indicado en repetidas ocasiones a lo largo de esta memoria el uso de la informática es un tema que cada vez está más en auge, por lo tanto como trabajos futuros se propone la ampliación de esta herramienta en las siguientes direcciones:

- Un trabajo futuro inicial sería la realización del ***eAula author*** , es decir de una herramienta que se encargue de hacer y montar las actividades que colgará en el servidor para que eAula pueda trabajar con ellas, ya que en la realización de este proyecto al no existir dicha herramienta se han colgado directamente en el servidor.
- También sería útil la realización de nuevos juegos, ya que actualmente eAula cuenta con asociaciones y puzzles, pero sería interesante por ejemplo que contase también con actividades de tipo “memory” o de tipo “crucigrama”.... .
- Se podría llevar a cabo la adaptación de eAula para diferentes rangos de edad, para lo que habría que adaptar tanto los juegos, como la manera de puntuar y la interfaz que se muestra.
- Otra tarea importante a realizar sería el dar a eAula la posibilidad de cambio automático en la interfaz, cambio de temas y de colores, sobre todo enfocándolo no tanto a la estética como a la necesidad de los niños con algún tipo de discapacidad, que necesiten unos colores determinados para poder realizar los ejercicios.
- También sería interesante la realización de una versión de eAula vía web para que los niños que no pueda ir al colegio puedan seguir la clase desde casa, para lo cual se integraría una especie de chat o video conferencia para que dichos niños pudiesen comunicarse con el profesor, así como añadir un foro bien para los niños o para que los padres puedan hacer consultas al profesor vía web.
- Por último, para poder mejorar la parte usable de la interfaz sería interesante que se pudiesen llevar a cabo algunas de las técnicas de las que se habló en el capítulo tres de la memoria, implantando por ejemplo en un aula real eAula y realizando *test de usabilidad* para ver si realmente es fácil de usar y para mejorar las cosas que no lo sean lo suficiente.
- Otro posible trabajo futuro o complemento al trabajo realizado podría ser/debería ser montar algún experimento y ver qué aceptación, facilidades de uso o problemas se pueden encontrar al trabajar con un colectivo de niños/as de las edades consideradas.

BIBLIOGRAFIA

[Bastien et al., 1993] BASTIEN, Christian J.M; SCAPIN, Dominique L. (1993) *Critères Ergonomiques pour l'Évaluation d'Interfaces Utilisateurs (2.1).*

[Bauer, 1997] BAUER, F. (1997) *Software engineering: a practitioner's approach.* McGraw-Hill.

[Boehm et al., 1978] BOEHM, B.W; BROWN, J.R.; LIPOW, M.; MacLEOD, G.J; MERRITT, M.J (1978) *Characteristics of Software Quality*, North-Holland, N.Y.

[Booth, 1989] BOOTH, P (1898) *An Introduction to Human-Computer Interaction.* Move,UK. Lawrence Earlbaum Associates.

[Carroll, 2001] CARROLL, J.M. (2001) *Human-Computer Interaction in the New Millenium.* Addison-Wesley Publishing.

[Ferré] FERRÉ, Xavier *Principios Básicos de Usabilidad para Ingenieros Software.*

[Grady, 1987]. GRADY, Robert B. (1987) *Practical Software Metrics for Project Managementand Process Improvement.* Englewood Cliffs, N.J.: Prentice-Hall,1992.

[Hix et al., 1993] HIX, D.; HARTSON, H.R. (1993) *Developing User Interfaces: Ensuring Usability Through Product and Process.* John Wiley and Sons.

[ISO,1991] ISO/IEC Standard (1991) *ISO-9126 Software Product Evaluation.* Quality Characteristics and Guidelines for Their Use.

[Karat et al., 2003] KARAT, J. ;KARAT, C.N. (2003) *The Evolution of User-Centered Focus in the Human-Computer Interaction Field.* IBM Systems

Journal

[López et al.] LÓPEZ, Victor; MONTERO, Francisco; LOZANO, Maria; FERNANDEZ, Antonio; GONZÁLEZ, Pascual; PARRA, Marta; LÓPEZ, Rigoberto; MONTAÑES, Juan *Virtual-Prismaker: Ayudando en el proceso de Aprendizaje*. Escuela Politécnica Superior de Albacete (Departamento de Informática), Escuela Universitaria de Profesorado de E.G.B. (Departamento de Psicología), Instituto de Desarrollo Regional (Sección Tecnología de la Información).Universidad de Castilla-La Mancha

[Marchionini et al., 2003] MARCHIONINI, G.; LEVI, M.D. (2003) *Digital goverment information services: the Bureau of Labor Statistics case.* In Interactions.

[McCall et al., 1977] McCALL, J.A.; RICHARDS, P.K.; WALTERS, G.F. (1977) *Factors in Software Quality* RADC TR-77-369, US Rome Air Development Center Reports NTIS AD/A-049 014,015,1977.

[Mendoza] MENDOZA, L. *Sistemas de Información III.* Universidad Simón Bolívar (Departamento de Procesos y Sistemas)

[Montero, 2005] MONTERO, Francisco (2005) *Integración de calidad y experiencia en el desarrollo de interfaces de usuario dirigida por modelos.* Tesis doctoral realizada en la Universidad Politécnica de Castilla-La Mancha.

[Nielsen, 1993] NIELSEN, J. (1993) *Usability Engineering. AP Professional.*

[Paternò, 2000] PATERNÒ, F.(2000) *Model–based design and evaluation of interactive application.* Springer–Verlag.

[Preece et al.,1994] PREECE, J.; ROGERS, Y.; SHARP, H; BENYON, D.; HOLLAND,S.; CAREY, T. (1994) *Human-Computer Interaction.* Adison Wesley.

[Pressman, 1992] PRESSMAN, F. (1992) *Software engineering: a practitiones's approach*. McGraw-Hill.

[Rodríguez, 2003] MONTAÑES RODRIGUEZ, J (2003) *Aprender y Jugar (Actividades educativas mediante el material lúdico-didáctico Prismaker System)*, Cuenca: Ediciones de la Universidad de Castilla-La Mancha.

[Schuler et al.,1993] SCHULER, D.; NAMIOKA, A. (1993) *Participatory Design: Principles and Practices.* Lawrence Erlbaum Associates.

[Sheiderman, 1998] SHNEIDERMAN, B. (1998) *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley.

[El Eman et al., 1997] EL EMAM, Khaled (editor); DROUIN, Jean-Normand (editor); MELO, Walcélio (editor); DORLING, Alec (foreword by) (1997) *SPICE: The Theory and Practice of Software Process Improvement and Capability Detemination.*. Wiley-IEEE Computer Society Press.

[Whiteside et al.,1988] WHITESIDE, J.; BENNETT, J.; HOLTZBLATT, K. (1988) *"Usability Engineering: Our Experience and Evolution". Handbook of Human-Computer Interaction.* Elsevier North-Holland.

ANEXO A EVOLUCIÓN DE eAula

En este anexo lo que se va ha hacer es mostrar la evolución que ha sufrido la aplicación desde los prototipos iniciales hasta la actualidad, para ello se irá mostrando la evolución de las pantallas en las diferentes versiones de eAula, excepto en aquellos casos donde las pantallas fueron eliminadas y se mostrará solo dicha pantalla.

A.1 Evolución de la aplicación del alumno

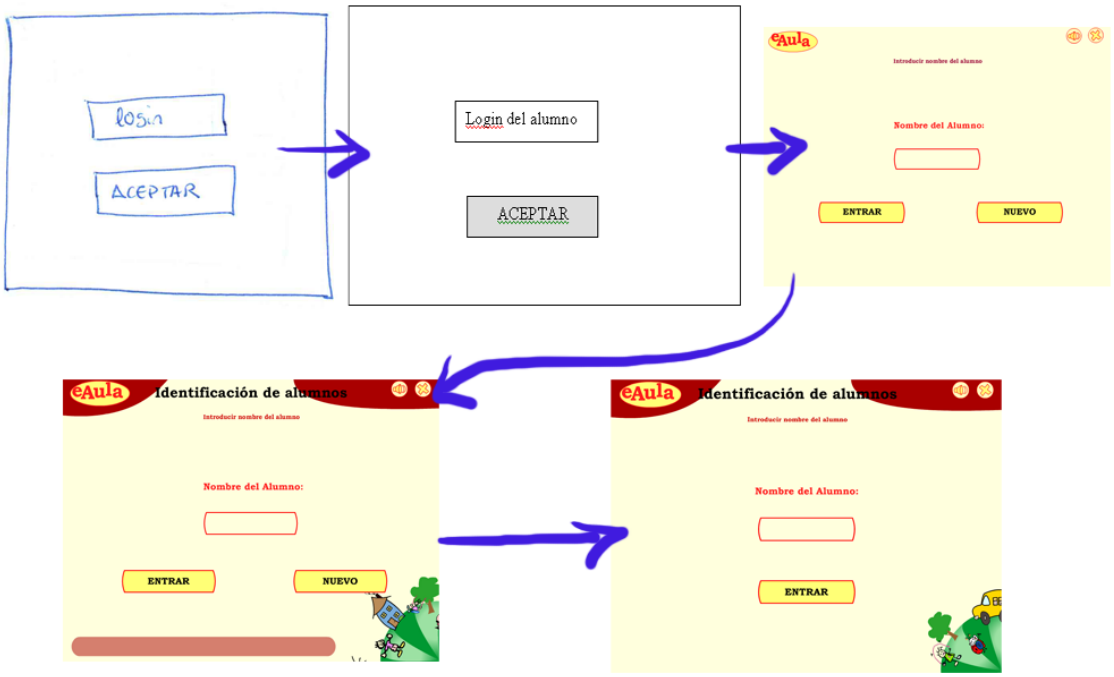

Figura ANEXO A.1 Evolución pantalla identificación alumno.

En la figura ANEXOA.1 se puede observar la evolución que ha sufrido la pantalla de identificación del alumno.

Figura ANEXO A.2 Evolución pantalla "esperando..." (alumno).

En la figura ANEXOA.2 se puede observar la evolución que ha sufrido la pantalla de "esperando datos del profesor", que durante un tiempo fue una pantalla de bienvenida. Su aspecto cambió además de por mantener una coherencia con el resto de la interfaz de la aplicación, porque al tener ahora una animación mientras se muestra dicha pantalla hace que el alumno mantenga la atención puesta en la pantalla y no se distraiga de lo que está haciendo mientras espera a que el profesor le envíe la tarea a realizar.

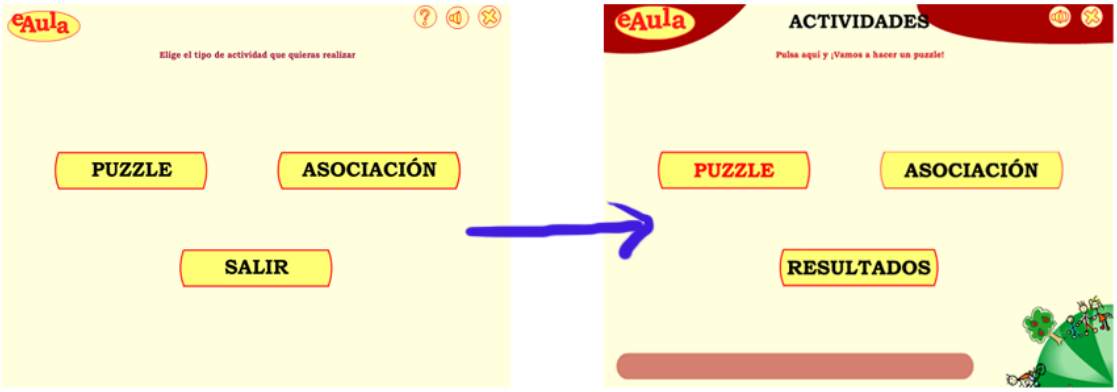

Figura ANEXO A. 3 Evolución pantalla "menú" (alumno).

En la figura ANEXOA.3 se puede observar la evolución de lo que inicialmente iba a ser un menú donde el alumno podría elegir que actividad realizar o salir de la aplicación (en el de la izquierda) y elegir que actividad realizar o consultar sus

resultados (en la de la derecha), pero esta funcionalidad se desestimó al no considerarse propia ni de un aprendizaje guiado ni de la edad que se trata en este proyecto.

Figura ANEXO A. 4 Evolución pantalla "resultados" (alumno).

En la figura ANEXOA.4 y haciendo referencia a lo que se hablaba antes sobre la figura ANEXOA.3, se muestra lo que en un principio eran los resultados que se podían consultar desde cada una de las máquinas y que reflejaba los resultados obtenidos por todos los alumnos que se habían sentado en la máquina, por lo que el profesor para hacer una consulta general sobre la evolución de la clase tenía que ir uno por uno por todos los ordenadores de la misma, por lo que está opción se descartó al no ser nada práctica y se cambió por unos resultados generales que se consultan desde la aplicación del profesor y cuya pantalla se mostrará a continuación en la figura ANEXOA.9.

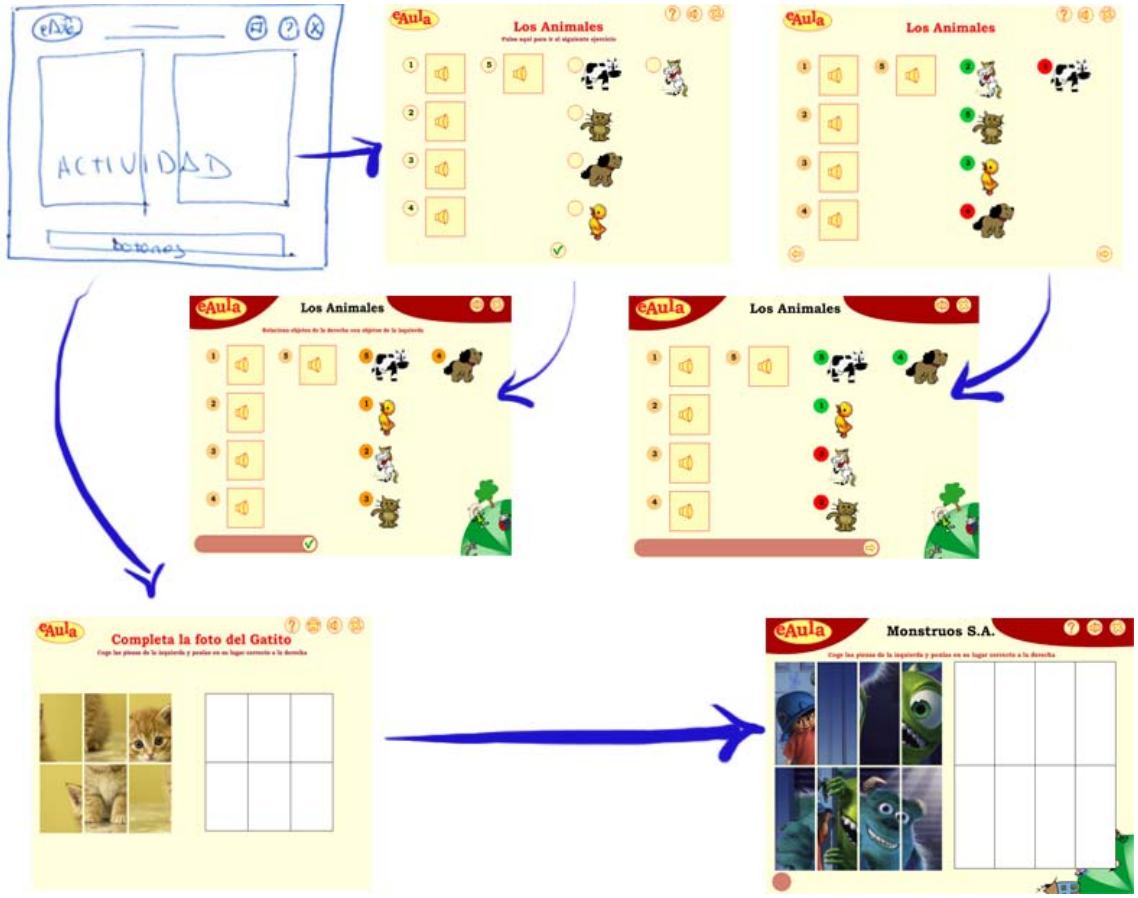

Figura ANEXO A. 5 Evolución pantalla actividades (alumno).

En la figura ANEXOA.5 se puede ver la evolución que has sufrido las pantallas de las actividades, tanto las asociaciones como los puzzles. Estos cambios no han sido solo a nivel visual sino también a nivel funcional, ya que antes el alumno podía saltarse la actividad sin necesidad de hacerla dando al botón de siguiente (como se puede ver en la figura ANEXOA.5 arriba a la izquierda) mientras que actualmente no puede saltarse ninguna actividad.

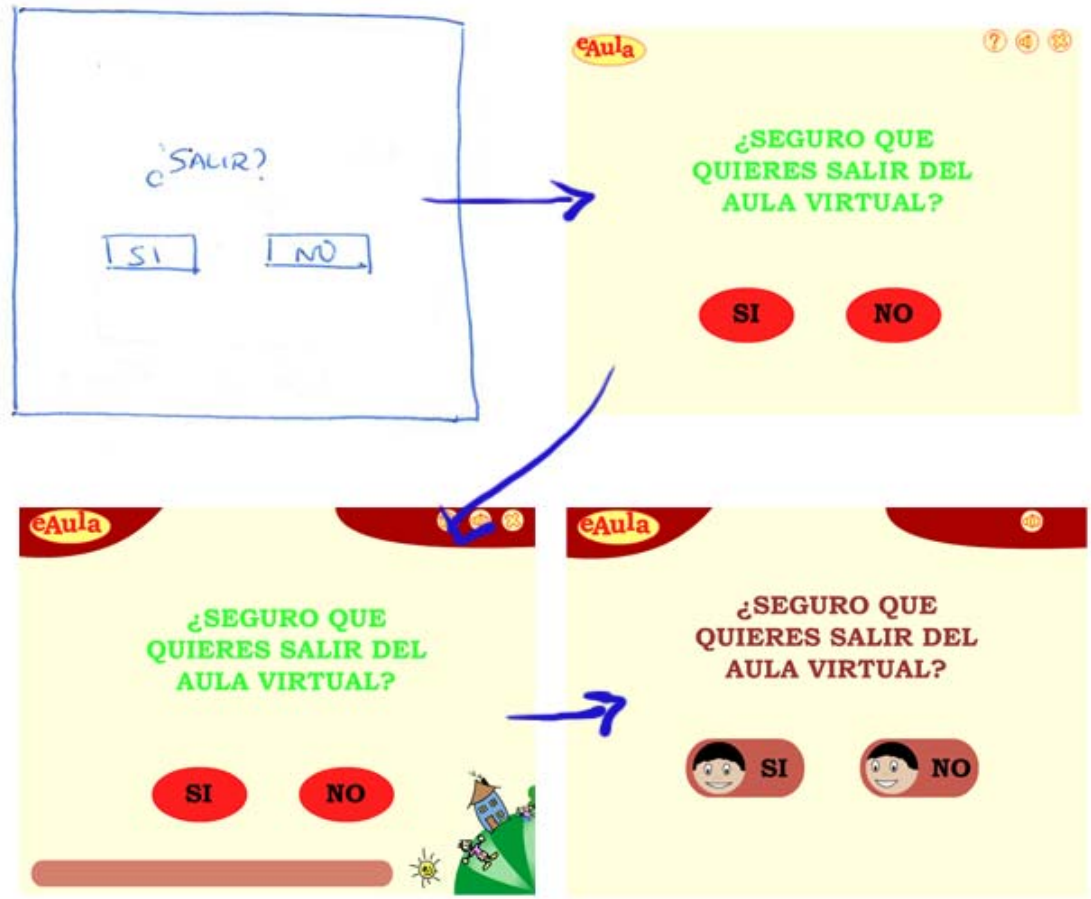

Figura ANEXO A. 6 Evolución pantalla salir (alumno).

En la figura ANEXOA.6 se muestra la evolución que ha sufrido la pantalla de salir del alumno. Con el aspecto que tiene actualmente, no solo guarda coherencia con el resto de las pantallas de la aplicación, sino que además mediante la animación que se realiza en cada uno de los botones (en el botón etiquetado con SI el muñeco mueve la cabeza de arriba abajo y en el botón etiquetado con NO la mueve de izquierda a derecha) se le está indicando al alumno la acción que se llevará a cabo si pulsa un botón u otro.

Con la existencia de esta pantalla además se evita que el alumno salga de la aplicación por equivocación, ya que si no era su intención pulsar el botón de cerrar desde esta pantalla podrá volver exactamente a donde se encontraba y si por el contrario desea salir, solo tendrá que volver a pulsar sobre el botón de etiquetado con SI.

A.2 Evolución de la aplicación del profesor

Figura ANEXO A. 7 Evolución pantalla identificarse (profesor).

En la figura ANEXOA.7 se puede observar la evolución que ha sufrido la pantalla de identificación del profesor, en este caso los cambios han sido más por motivo estético que funcional, ya que ahora su aspecto guarda una mayor concordancia con el resto de pantalla de la aplicación, pero su funcionalidad es la misma que tenía antes.

Figura ANEXO A. 8 Evolución pantalla de envío de actividades (profesor).

En la figura ANEXOA.8 se puede observar la evolución que ha sufrido la interfaz del profesor desde un prototipo inicial hasta la versión actual, ya no solo a nivel visual sino a nivel funcional, ya que en una primera versión solo podía enviar actividades a un alumno en concreto y ahora puede enviarlas a uno o varios, además antes solo podía ver los resultados parciales y ahora puede verlos y borrarlos cuando desee para refrescar la pantalla. Y otra cosa que se ha añadido a la funcionalidad es que antes para que viese las actividades disponibles tenía que cargarlas inicialmente y ahora una vez que arranca la aplicación se le rellena la lista de actividades automáticamente y en cualquier momento puede refrescarla para actualizar cambios.

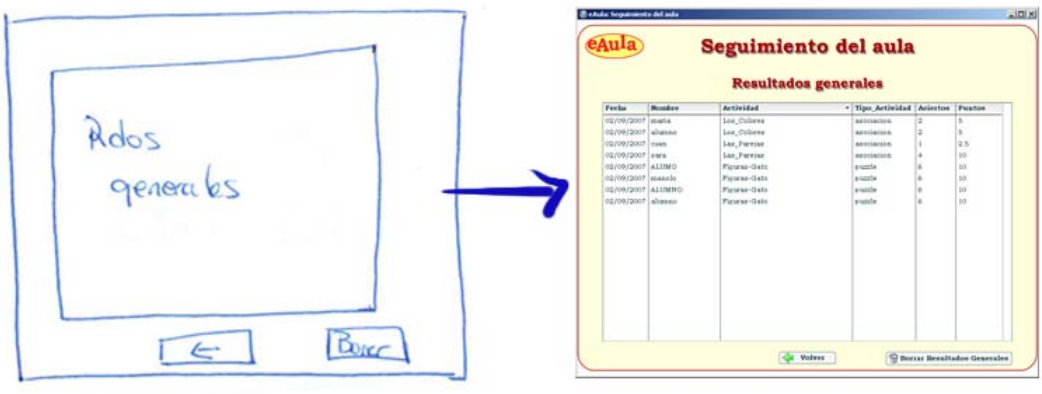

Figura ANEXO A. 9 Evolución pantalla de resultados generales (profesor).

En la figura ANEXOA.9 se puede observar la evolución que ha tenido la pantalla de resultados generales, a esta figura ya se hacía referencia en el punto de antes cuando se hablaba de la aplicación del alumno, ya que, como se comentó entonces antes los resultados se mostraban desde la aplicación del alumno, pero esto se cambió a como está actualmente porque atendía mejor a las necesidades de los usuarios de las aplicaciones.