Simposio de Informática y Telecomunicaciones SIT'02

Hacia un Sistema de Componentes Software para el Dominio del Aprendizaje Colaborativo Apoyado por Ordenador (CSCL)

Yannis A. Dimitriadis[1], Juan Ignacio Asensio[1], Javier Toquero[1], Laura Estébanez[1], Tomás Alberto Martín[1], Alejandra Martínez[2] *

[1] Dpto. de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, Universidad de Valladolid, Camino del Cementerio s/n, 47011 Valladolid
{yannis,juaase}@tel.uva.es, {toquero,rogero,tmarcam}@pireo.tel.uva.es
[2] Dpto. de Informática, Universidad de Valladolid, Camino del Cementerio s/n, 47011 Valladolid
amartine@infor.uva.es

Resumen La ISBC (Ingeniería de Software Basada en Componentes) está emergiendo como un nuevo paradigma realista para todo el ciclo de desarrollo de aplicaciones distribuidas. Sin embargo, es necesario analizar su aplicabilidad y los problemas derivados de su uso siguiendo criterios de eficiencia científica y las pautas dadas por la iniciativa MDA de OMG. En este artículo se presenta el trabajo destinado a la obtención de un sistema de componentes software en el dominio del CSCL (*Computer Supported Collaborative Learning* o Aprendizaje Colaborativo Apoyado por Ordenador). Este dominio es particularmente interesante dado su impacto social y económico, la necesidad de flexibilidad y adaptabilidad, su carácter multidisciplinar y los problemas derivados de la necesidad de disponer de aplicaciones adecuadas a los requisitos de pedagogos, proveedores de servicios y contenidos o usuarios finales. El trabajo experimental se ilustra a través del ciclo de desarrollo de un conjunto de puzzles colaborativos para educación infantil. En este sentido, se detallan las principales conclusiones derivadas de dicho desarrollo, incidiendo especialmente en aspectos de gestión de componentes, uso de patrones de diseño y marcos de aplicación, interoperabilidad e independencia de plataformas distribuidas, etc.

1 Introducción

La educación es un dominio clásico de aplicación de las Tecnologías de la Información y de las Telecomunicaciones (TICs). Se puede afirmar que las sucesivas

* Los autores desean dar las gracias a todos los miembros y colaboradores del grupo transdisciplinar EMIC (Educación, Medios, Informática y Cultura) de la Universidad de Valladolid por sus contribuciones. El trabajo presentado en este artículo está parcialmente financiado por el Ministerio de Ciencia y Tecnología (proyecto TIC2000-1054), la Junta de Castilla y León (proyecto VA 117/01) y la empresa Ediciones Don Bosco (Edebe).

innovaciones tecnológicas han tenido reflejo en el ámbito de la informática educativa [26].

El dominio educativo posee ciertas características que le son propias y que de forma permanente aparecen en el desarrollo de sistemas de software educativo. Una de las más recurrentes ha sido la necesidad de proporcionar sistemas flexibles, capaces de adaptarse a las necesidades concretas de cada situación educativa. Los profesores no aceptan fácilmente aplicaciones cerradas, sino que necesitan configurarlas atendiendo a su propio contexto educativo. A esta demanda de flexibilidad y adaptabilidad en el dominio educativo se ha ido respondiendo a través de diversas propuestas arquitectónicas [17]. Sin embargo, las propuestas ofrecidas quedan lejos de proporcionar sistemas abiertos que un profesor pueda adaptar a sus necesidades. O bien los sistemas son difíciles de adaptar, o bien las alternativas de configuración son demasiado reducidas.

El paradigma de componentes software ofrece, al menos potencialmente, la promesa de sistemas abiertos, modulares y configurables, que cumplen los requisitos de flexibilidad y capacidad de configuración por parte de profesores. Un reflejo de este interés es la existencia de varios proyectos de investigación importantes trabajando en este área [16,25,20].

Sin embargo, la mayoría de los sistemas existentes basados en componentes para educación se centran en sistemas aislados y no tienen en cuenta los aspectos colaborativos, elemento esencial en el paradigma educativo de constructivismo social [8]. Este nuevo aspecto implica la necesidad de estudiar la relación de los sistemas distribuidos, más allá del uso básico de las tecnologías Web, con la ISBC para el campo de la educación. Estos elementos (el nuevo paradigma educativo de constructivismo social, el papel de las interacciones sociales, así como el trabajo colaborativo y su apoyo por las TICs) dieron lugar al campo de CSCL [10]. Este dominio es particularmente interesante por:

- Su carácter fuertemente multidisciplinar (educación, psicología, sociología y soporte tecnológico).
- La especial complejidad de desarrollo de sistemas CSCL, puesto que se tienen que resolver problemas de interfaces hombre - hombre a través de la red, de sistemas distribuidos, o de la existencia de múltiples actores en el proceso de análisis y diseño, etc.
- La necesidad de flexibilidad y adaptación a las distintas situaciones y actividades educativas.

Desde los inicios de investigación y desarrollo en el campo de CSCL y en CSCW (*Computer Supported Cooperative Work* o Trabajo Cooperativo Apoyado por Ordenador) a principios de los años 90, se han publicado notables esfuerzos para la mejora de la eficiencia y eficacia en el proceso de desarrollo de sistemas CSCL. Entre ellos podemos destacar el uso de *toolkits* [18], patrones de diseño [6], lenguajes de propósito específico [1] y extensiones de modelos de componentes para soportar cierto grado de adaptabilidad (*tailoring*) [22]. Sin embargo, estos intentos son parciales y adolecen de problemas tales como dependencia de las tecnologías subyacentes, falta de un marco para el diseño participativo, poca atención a las necesidades de los educadores, etc.

En este artículo se presenta el trabajo desarrollado dentro del grupo transdisciplinar EMIC destinado a la obtención de un sistema basado en componentes para el dominio de CSCL. En este sentido se pretende extraer lecciones sobre los múltiples aspectos del empleo de ISBD en sistemas distribuidos. Se ha optado por su estudio en un dominio concreto, por razones de eficiencia y siguiendo la tendencia reflejada en el MDA (*Model Driven Architecture* o Arquitectura Guiada por Modelo) [5] de OMG (*Object Management Group* o Grupo de Gestión de Objetos). Dicha tendencia aboga por *separar la especificación, basada en "modelos", de la funcionalidad de un sistema de la especificación, también mediante "modelos", de la implementación de dicha funcionalidad en una plataforma de tecnología específica*. Aprovechando los méritos de esta aproximación, que permite independizar los aspectos propios de un dominio de las cuestiones tecnológicas de soporte, podemos crear sistemas completos y significativos, ya que se refieren a un dominio concreto. En nuestro caso, se aprovecha la experiencia anterior del grupo que condujo a la propuesta del marco conceptual Delfos (*a Description of tele-Educational Layer-Framework Oriented to learning Situations*) [14] para el diseño y evaluación de sistemas CSCL reales.

Inicialmente, en la sección 2, se presentan los antecedentes en relación con el uso de las TICs en la educación y los problemas derivados del software educativo cuando se trata el dominio CSCL. Así, podemos plantear los requisitos y la importancia del dominio con respecto a la ISBC y contrastar soluciones existentes. Posteriormente, en la sección 3, formulamos el marco, los objetivos y la metodología empleada en el proyecto de investigación COSACO (Componentes Software para Aplicaciones de Aprendizaje Colaborativo). Con el objetivo de ilustrar nuestro trabajo experimental, presentamos en la sección 4 el desarrollo de un conjunto de puzzles colaborativos y discutimos los problemas asociados a la gestión de componentes, el uso de patrones de diseño y marcos de aplicación, la interoperabilidad e independencia de plataformas distribuidas, etc. Finalmente, en la sección 5, se exponen las principales conclusiones y las líneas actuales y futuras de trabajo.

2 Posibilidades de la ISBC aplicada a CSCL

Los esfuerzos por aplicar las innovaciones tecnológicas a la mejora de la educación se han sucedido a lo largo de la historia. En el caso de las TICs, han dado lugar a diversos paradigmas de informática educativa: enseñanza asistida por ordenador, sistemas tutores inteligentes, simulaciones o micromundos, y más recientemente, con la generalización de las redes de ordenadores, a la enseñanza a distancia y al paradigma CSCL [26]. Esta diversidad no se basa en la tecnología utilizada, sino en la teoría sobre el aprendizaje en la que se basa cada una de ellas. Por ejemplo, el CSCL es un paradigma basado en teorías que resaltan la influencia de las interacciones sociales como mediadoras del aprendizaje [10].

El CSCL es un área íntimamente relacionada con el CSCW [2], ya que ambas se orientan al soporte de la colaboración como base para la consecución de un objetivo (eficiencia en el trabajo en CSCW, mejora del aprendizaje en

CSCL). Ambas tecnologías comparten problemas comunes, entre los que destaca el carácter multidisciplinar del trabajo y la complejidad del desarrollo de sistemas distribuidos. El CSCL añade a estas dificultades las procedentes del dominio educativo, que comentamos brevemente a continuación.

La aplicación de cualquier herramienta, y por tanto, de una aplicación informática al aula requiere que los profesores sean capaces de adaptarla al contexto educativo y social en el que se encuentran, así como a su estilo pedagógico. Esta necesidad de personalización ha conducido tradicionalmente al desarrollo de un gran número de aplicaciones específicas. Éstas suelen ser aplicaciones monolíticas, dependientes de tecnologías particulares, e incompatibles entre sí, por lo que los profesores encuentran grandes dificultades para integrar varias de ellas en sus clases. Estos proyectos poseen un alto índice de fracaso, debido a su falta de capacidad para adaptarse a nuevas situaciones educativas y para asumir las innovaciones tecnológicas que continuamente se están produciendo [19].

Por otro lado, el desarrollo de software educativo de calidad implica la presencia de un gran número de actores: expertos en pedagogía y en la materia de estudio, profesores, programadores, diseñadores gráficos, y alumnos. El coste de este tipo de proyectos interdisciplinares es muy alto, y sólo es sostenible si los resultados son aplicables en una gran cantidad de situaciones y si son capaces de asumir los cambios tecnológicos [16].

Ante esta realidad, la búsqueda de arquitecturas abiertas y adaptables para el desarrollo de software educativo ha sido una constante a lo largo de los años. Las soluciones ofrecidas incluyen entornos educativos de programación, sistemas de autor, y el uso de estándares abiertos para la creación de materiales [17]. A pesar de los avances, ninguna de estas líneas de trabajo ha respondido totalmente a las necesidades de los nuevos entornos educativos colaborativos comentados anteriormente.

La tecnología de componentes software ofrece la promesa de la composición de herramientas procedentes de diversos proveedores, y por tanto aparece como una solución potencial a los problemas aquí señalados. De acuerdo con [24], un componente es una *"unidad de composición de aplicaciones software que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio"*. Esta idea se presenta potencialmente interesante para la solución de los problemas de reutilización y composición de aplicaciones tan presentes en el dominio educativo ya comentadas anteriormente. De hecho, ha dado lugar a varios proyectos, en los que se ha aplicado con éxito la idea de los componentes al desarrollo de aplicaciones educativas [16]. Como conclusión de estas primeras experiencias, los autores afirman que la adopción del enfoque de componentes dentro del mundo educativo pasa por acercar la visión de los componentes a los usuarios (profesores, diseñadores del curriculum), así como por aumentar la oferta de componentes software disponibles, de cara a conseguir su aceptación en el dominio escolar. La cuestión del soporte a la colaboración no fue considerada en estos proyectos.

La Ingeniería del Software Basada en Componentes (ISBC) estudia el diseño y desarrollo de aplicaciones distribuidas basadas en componentes software reutilizables. Esta disciplina ha dado lugar desarrollo de plataformas de componentes software distribuidos, lo que ha abierto el camino para el estudio de las posibilidades del paradigma de componentes en entornos abiertos distribuidos. El proyecto COSACO plantea esta cuestión centrada en el dominio del CSCL.

3 El Sistema Propuesto

COSACO es un proyecto en el que educadores y tecnólogos trabajan conjuntamente para investigar de qué manera (y en qué grado) la tecnología de desarrollo software basado en componentes puede ayudar a solucionar problemas de falta de flexibilidad y de integración de las aplicaciones CSCL comentados en la sección 2.

COSACO tiene la ventaja de contar con el trabajo previo de su grupo de investigadores en el campo CSCL, tanto desde la perspectiva pedagógica como tecnológica, y que dió como fruto el marco telemático-educativo Delfos [14] en el que se analizan y formalizan las interacciones mutuas entre ambos dominios y donde se proponen mecanismos concretos para posibilitarlas facilitando, de esa manera, el desarrollo de sistemas TIC que reflejen y realimenten los principios y conceptos educativos en ellos subyacentes.

El objetivo final de COSACO es disponer de una biblioteca de componentes software y las herramientas asociadas que permitan, de manera flexible y poco costosa, el desarrollo de aplicaciones de componentes software distribuidos CSCL particularizadas a entornos concretos con requisitos educativos determinados.

En este sentido, el modelo del entorno o sistema propuesto se podría resumir en:

- La generación de componentes educativos abstractos por parte de los pedagogos a partir del análisis de técnicas o situaciones CSCL.
- La provisión de soporte a los proveedores de contenidos y servicios para que sean capaces de particularizar los componentes anteriores a los requisitos educativos de un entorno concreto.
- La obtención de componentes software concretos, a partir de los componentes abstractos ya mencionados, y la generación de aplicaciones CSCL basadas en ellos mediante la herramienta adecuada (herramienta que, en el contexto de COSACO, recibe el nombre de *herramienta de gestión de arquitectura-componentes*).

Como se puede apreciar, el proyecto intenta tener en cuenta las necesidades de los actores principales involucrados en la generación y aplicación de software educativo: pedagogos, proveedores de servicios y contenidos y desarrolladores software.

Mediante esa aproximación, alguien conocedor del dominio CL pero no del campo del desarrollo software basado en componentes podría obtener aplicaciones CSCL particularizadas a sus requisitos.

La consecución de los objetivos de COSACO pasa por el desarrollo de dos líneas de trabajo:

- Identificación, caracterización y desarrollo de componentes software reutilizables y particularizables que puedan ser empleados en diferentes tipos de aplicaciones CSCL.
- Análisis y desarrollo de las herramientas necesarias para, empleando los componentes de la biblioteca ya comentada, crear aplicaciones CSCL que cumplan con requisitos educativos concretos.

Dos tipos de tareas se plantean para avanzar en las dos líneas de trabajo anteriores:

- Estudio de aplicaciones CSCL ya desarrolladas por el grupo de investigación para identificar funcionalidades comunes y, por lo tanto, para avanzar en la identificación de los componentes básicos CSCL.
- Desarrollar, desde cero, aplicaciones CSCL basadas en componentes distribuidos para:
  - contribuir a la tarea de identificación de componentes software básicos en el dominio CSCL.
  - disponer de prototipos sobre los que probar las tecnologías involucradas y las herramientas a desarrollar en el proyecto: fundamentalmente la *herramienta de gestión de arquitectura-componentes*.

Puesto que son varias las tecnologías de componentes distribuidos existentes en la actualidad, en COSACO se pretende diseñar aplicaciones cuya funcionalidad y arquitectura sean independientes de dichas tecnologías. En ese sentido, en el proyecto se pretende utilizar los principios y conceptos (ya mencionados en la sección 1) de la iniciativa MDA de OMG [5]: intentar obtener modelos de arquitectura de aplicación independientes de tecnologías particulares de componentes distribuidos.

La siguiente sección describe el trabajo realizado (y los resultados del mismo) dentro de COSACO destinado a la obtención de componentes software para construir un tipo concreto de aplicaciones CSCL: *puzzles colaborativos*. La construcción del soporte tecnológico (CS) de diferentes aplicaciones "puzzle" particularizadas a diferentes situaciones de aprendizaje colaborativo (CL) se lleva a cabo gracias a una herramienta de gestión de componentes que también se describe en la sección 4.

4 Caso de Estudio: un Puzzle Colaborativo

Tal y como se ha comentado en la sección anterior, una de las líneas de actuación planteadas dentro del proyecto COSACO consiste en desarrollar prototipos de aplicaciones, basadas en componentes, de aprendizaje colaborativo con los que poder avanzar hacia el objetivo de identificar componentes del dominio CSCL

evaluando, al mismo tiempo, la flexibilidad y capacidad de reutilización de los mismos.

Dentro del ámbito del proyecto COSACO se trató de determinar, por tanto, un tipo de aplicación CSCL cuyo nivel de complejidad fuera lo suficientemente bajo como para permitir el desarrollo rápido de prototipos y, al mismo tiempo, lo suficientemente elevado como para compartir la problemática de las aplicaciones CSCL más elaboradas de forma que los componentes identificados y desarrollados pudieran, potencialmente, ser reutilizados en la mayoría de ellas. La decisión, en este sentido, fue la de desarrollar una aplicación de resolución colaborativa de *puzzles* destinada a alumnos de educación infantil. Este tipo de aplicaciones destaca por su conocido beneficio educativo y socializador [7] además de por su capacidad de reflejar el proceso de construcción del conocimiento.

En la sección 4.1 se detallan las características de la aplicación *puzzle colaborativo*, qué variantes se desarrollaron, cómo se llevó a cabo dicho desarrollo, qué tecnologías se emplearon y cuáles fueron los resultados obtenidos en términos de componentes CSCL identificados y grado de reutilización de los mismos en otros tipos de aplicaciones del dominio CSCL.

Ahora bien, los objetivos del proyecto COSACO no se limitan al desarrollo de prototipos de aplicaciones CSCL sino, también, a evaluar de qué manera la ingeniería software basada en componentes puede contribuir a solucionar los problemas identificados en la sección 2. Uno de dichos problemas residía en la necesidad de que los educadores pudieran adaptar el software CSCL al entorno particular de uso y a los objetivos de aprendizaje fijados en cada caso. Con ese objetivo, y aprovechando los resultados del desarrollo de la aplicación *puzzle colaborativo*, dentro del proyecto COSACO se está desarrollando una aplicación de gestión de componentes software destinada a que un usuario desconocedor de las particularidades de la ISBC sea capaz, a partir de una biblioteca de componentes software existente, de generar una aplicación de tipo *puzzle colaborativo* con características de funcionamiento adecuadas a sus necesidades. La sección 4.2 describe esta aplicación de gestión de componentes, cuáles son sus objetivos, de qué manera se está desarrollando y cómo debería evolucionar en el futuro para servir de apoyo a la aplicación de la ISBC dentro del dominio CSCL.

4.1 Un Puzzle Colaborativo Basado en Componentes

Objetivos, Ámbito y Características del "Puzzle Colaborativo". Como de su propio nombre se puede inferir, la aplicación *puzzle colaborativo* desarrollada dentro del proyecto COSACO tiene como finalidad que un grupo de alumnos de educación infantil (niños de edades comprendidas entre los 3 y los 6 años) participe en la resolución de un problema consistente en la colocación de un conjunto de piezas en unas posiciones concretas dentro de un tablero.

Para perfilar los requisitos funcionales de dicha aplicación tales como criterios de colocación de piezas (en función de la forma de los huecos del tablero, en función del contenido visual de las piezas, etc.), forma de las piezas, número de piezas, número de jugadores, reparto de las piezas entre los jugadores, método

para mover las piezas, visibilidad entre jugadores, concurrencia (control de acceso de los jugadores a las piezas), etc., se realizó una encuesta entre un conjunto de expertos en pedagogía infantil cuyo resultado determinó que la mejor aproximación al problema consistiría en desarrollar dos variantes del *puzzle colaborativo*. En ambos casos la aplicación iría destinada a un número de jugadores comprendido entre 1 y 4 y se emplearían piezas rectangulares. Sin embargo, en una de las variantes el reparto de las piezas sería aleatorio (todos los jugadores dispondrían del mismo número de piezas), cada jugador podría ver y acceder únicamente a sus piezas, no se permitiría colocar piezas en posiciones incorrectas y el movimiendo de las mismas se realizaría mediante acciones de "arrastre" (*drag & drop*). Por contra, en la otra variante todos los jugadores dispondrían de todas las piezas, jugarían según un turno fijo, sí se les permitiría colocar piezas en posiciones incorrectas del tablero y deberían mover las mismas mediante dobles pulsaciones de los botones del ratón.

Las dos variantes de la aplicación *puzzle colaborativo* incorporarían la funcionalidad necesaria como para soportar cinco posibles *papeles* (*roles*) diferentes en que sus usuarios podrían actuar: *profesor* (controla la formación de grupos y el inicio y finalización de las partidas), *alumno* (usuario que juega al *puzzle* propiamente dicho), *diseñador* (usuario encargado de la adaptación de la aplicación en función de los objetivos educativos), *evaluador* (del proceso de aprendizaje) y *administrador* (encargado del mantenimento, gestión y configuración de la aplicación).

Metodología, Decisiones de Diseño y Tecnologías Empleadas. Como paso previo al desarrollo de las variantes del *puzzle colaborativo* anteriormente caracterizadas, se tomaron una serie de decisiones tecnológicas y metodológicas que es interesante conocer puesto que su influencia en los resultados obtenidos fue determinante. Estas decisiones se resumen a continuación:

1. *Marco Arquitectónico y Conceptual*: Para dar soporte al carácter distribudo y multiusuario del *puzzle* colaborativo se optó por emplear una arquitectura software estructurada en tres capas: capa de cliente, capa intermedia (con los componentes que dan soporte a la lógica de la aplicación) y capa de sistemas de información (con acceso a gestores de bases de datos) [9]. Por otro lado, se reutilizaron las propuestas y recomendaciones para el desarrollo de aplicaciones CSCL recogidas en el marco conceptual Delfos [15] ya mencionado en la sección 1.
2. *Metodología de desarrollo software*: Todos los pasos previos a la implementación y pruebas (análisis de requisitos, diseño arquitectónico, diseño detallado, etc.) fueron guiados por un proceso de desarrollo software obtenido de la particularización de RUP (*Rational Unified Process* o Proceso Unificado de Rational) [11] al entorno particular del proyecto COSACO. Por lo tanto, UML (*Unified Modeling Language* o Lenguaje Unificado de Modelado) [4] constituyó el lenguaje base para la mayor parte de los *artefactos* software generados durante el desarrollo obteniendo, de esa manera, modelos de la aplicación independientes de tecnologías de implementación concretas.

3. *Patrones de diseño*: En el diseño de la aplicación se han reutilizado una serie de patrones de diseño software:
   - Patrones TOP (*Ten Object Patterns* o Diez Patrones de Objetos) [6] diseñados especialmente para ser empleados en aplicaciones colaborativas.
   - El patrón MVC (*Model View Controller* o Modelo-Vista-Controlador) [23] que permite independizar los datos manejados por la aplicación de la forma en que son presentados al usuario.
4. *Plataforma de componentes distribuidos*: A la hora de escoger las tecnologías de implementación concretas sobre las que plasmar las labores de modelado descritas anteriormente, para esta primera aplicación CSCL desarrollada dentro del proyecto COSACO se decidió utilizar una plataforma de componentes distribudos con arquitectura J2EE (*Java 2 Enterprise Edition*) [13] y modelo de componentes EJB (*Enterprise Java Beans*) [12] por su mayor madurez con respecto a otras alternativas ya mencionadas en la sección 2, independencia respecto a plataforma y disponibilidad de implementaciones. De entre las implementaciones disponibles se decidió utilizar la denominada "Implementación de Referencia" de Sun Microsystems sobre el sistema operativo *Microsoft Windows 2000*.

Resultados y Discusión. Como resultado del proceso de desarrollo anteriormente comentado, se obtuvieron sendos prototipos de *puzzle colaborativo* por cada una de las variantes ya comentadas. En ambas variantes únicamente se tuvo en cuenta el soporte para los *roles* de alumno, profesor y administrador.

En relación con los objetivos del proyecto COSACO cabe destacar que los dos prototipos poseen una arquitectura formada por cinco componentes software diferentes (al haber implementado la aplicación sobre una plataforma J2EE se puede hablar, por lo tanto, de cinco *Enterprise Java Beans*): *Gestor de Acceso*, *Sala de Espera*, *Gestor de Sesiones*, *Gestor de Juego* y *Gestor de Turnos*. La figura 1 esquematiza dicha estructura.

Sin entrar a describir la funcionalidad de cada componente, lo más relevante desde la perspectiva de este artículo reside en el interrogante de si esta división en componentes es la más adecuada cuando se persigue, como es el caso del proyecto COSACO, la flexibilidad y la reutilización de dichos componentes en las aplicaciones de un dominio concreto como es CSCL.

La flexibilidad y reutilización [21] se alcanzan cuando se consigue un compromiso adecuado entre *granularidad* y *especificidad* dentro de un dominio. La *granularidad* se refiere a la cantidad de partes que constituyen un sistema. En el caso del *puzzle colaborativo* desarrollado, compuesto únicamente por cinco componentes, se puede concluir que el grado de granularidad es bajo. Esto afecta a la reutilización y a la flexibilidad de los componentes ya que cuanto menos granular sea la aplicación menos probable será que dichos componentes puedan ser empleados en otras aplicaciones. Ahora bien, se corre el riesgo de que, al aumentar la granularidad, se disminuya la *especificidad* dentro de un dominio. Dicho de otro modo, al aumentar la granularidad se hace más difícil identificar componentes que, por sí solos, constituyan una parte funcional importante de un

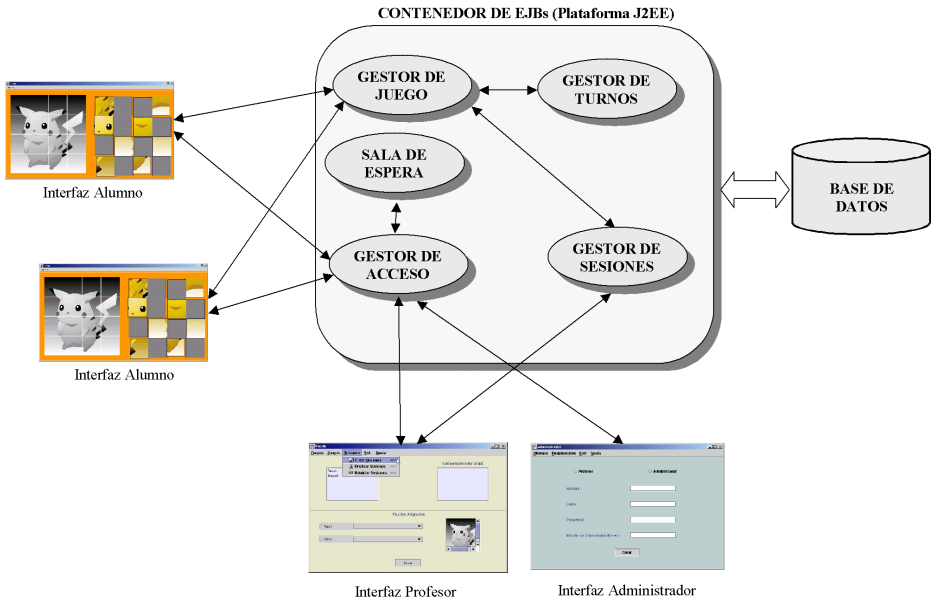

Figura 1. Estructura de componentes de la aplicación *puzzle colaborativo*

conjunto apreciable de aplicaciones del dominio (CSCL en el caso del proyecto COSACO). Así, por ejemplo, aunque el grado de *granularidad* del *puzzle colaborativo* descrito es bajo, se puede apreciar que componentes tales como *Gestor de Acceso*, *Gestor de Sesiones*, *Gestor de Turnos* y *Sala de Espera* soportan funcionalidades que son comunes a un gran número de aplicaciones dentro del dominio CSCL y, eventualmente, podrían reutilizarse. Por contra, el componente *Gestor de Juego* es únicamente reutilizable en otras aplicaciones de tipo *juego educativo* o *puzzle*, ya que recoge la lógica de la actividad educativa.

Otro aspecto a tener en cuenta cuando se persigue la flexibilidad y reutilización de componentes es la *generalización funcional*: si el diseño de un componente se hace teniendo en cuenta las interacciones con otros componenes de un tipo concreto de aplicación, será difícil reutilizar ese componente en otros tipos de aplicaciones en que dichas interacciones no son posibles. Este hecho está íntimamente relacionado con la *granularidad* puesto que un aumento de la misma trae consigo un aumento de las interdependencias entre componentes y, por lo tanto, una dependencia creciente de los componentes con respecto a la arquitectura de la aplicación en que se van a utilizar. En este caso se puede recurrir, como ha sucedido en el *puzzle colaborativo*, a patrones de diseño que permiten reutilizar, no únicamente componentes, sino también partes de la arquitectura de una aplicación en arquitecturas de aplicaciones de otros tipos.

4.2 Herramienta de Gestión de Componentes

Objetivos y Ámbito del Sistema Desarrollado. La *herramienta de gestión de componentes* que se está desarrollando dentro del ámbito del proyecto COSACO constituye un primer paso hacia el objetivo último de disponer de una herramienta que permita generar aplicaciones CSCL, basadas en componentes, particularizables a situaciones concretas sin necesidad de conocer los detalles propios de la ISBC.

El funcionamiento ideal de esta herramienta se resume en los siguientes pasos:

1. El usuario de la herramienta decide qué tipo de aplicación CSCL desea generar y con qué características funcionales. Hay que destacar que la información que se solicita al usuario se limita a cuestiones propias del dominio CSCL.
2. La herramienta acude a una biblioteca de componentes CSCL y escoge los más adecuados para satisfacer los requisitos determinados por el usuario.
3. La herramienta modifica las propiedades de los componentes elegidos (en los casos en que sea necesario) y los "ensambla" para formar una aplicación.
4. Por último, la herramienta despliega la aplicación CSCL generada en la plataforma de componentes distribuidos correspondiente.

En la primera versión de la herramienta, desarrollada dentro del proyecto COSACO, los objetivos han sido menos ambiciosos y, tal y como se describe a continuación, simplemente se ha intentado explorar las necesidades de una herramienta de las características descritas.

Metodología y Tecnología Empleadas. La *herramienta de gestión de componentes*, en su primera versión, se limita a permitir la generación de una de las dos variantes de la aplicación *puzzle colaborativo* descritas en la sección 4.1.

En función de la información introducida por el usuario (se desea un *puzzle* con turno o sin turno, con visibilidad de todas las piezas o no, etc.), la herramienta decide cuál de las variantes de *puzzle colaborativo* se ha de generar e intenta localizar, de entre un repositorio de componentes, aquellos que mejor se ajustan a la variante escogida. Además, la *herramienta de gestión de componentes* es capaz de configurar ciertas propiedades de algunos de los componentes (por ejemplo, para indicar si en el tablero se ha de dibujar o no, como fondo, la figura que se ha de completar con las pieza de las que disponen los alumnos).

Los componentes escogidos y particularizados se "ensamblan", de acuerdo con la arquitectura de componentes mostrada en la figura 1, y se despliegan en la plataforma de componentes que, en el caso de la aplicación *puzzle colaborativo*, ha de ser compatible con la arquitectura *J2EE*.

En todos los casos, las operaciones de modificación de propiedades y "ensamblaje" de componentes se basan en una adecuada modificación de los *descriptores de despliegue* [13] de los *Enterprise Java Beans* involucrados.

La localización de los componentes necesarios, en función de la información del dominio CSCL, se basa en la introducción de metadatos *ad-hoc* en dichos descriptores de despliegue.

Resultados y Discusión De esta primera versión de la *herramienta de gestión de componentes* limitada a la generación de *puzzle colaborativos* se han podido extraer una serie de conclusiones de interés que indican cómo seguir avanzando hacia una herramienta válida para la generación y particularización de aplicaciones CSCL de diversos tipos a partir de una biblioteca de componentes y en función de información propia del dominio CSCL (objetivos de aprendizaje, técnicas de colaboración, etc.):

- Para que la herramienta de gestión de componentes no se limite a tipos de aplicaciones concretas, es necesario que pueda manejar múltiples descripciones de arquitecturas de aplicación. Dichas descripciones arquitectónicas:
  - Han de ser descripciones independientes de la tecnología de componentes distribuidos. Los conceptos y principios de la ya mencionada MDA [5] han de ternerse presentes en este contexto.
  - Pueden ser descripciones de arquitecturas no completamente fijadas sino que, por el contrario, han de poder presentar ciertos "grados de libertad". El objetivo es que una misma descripción de arquitectura de aplicación pueda dar lugar a diferentes aplicaciones en función de los requisitos del dominio CSCL proporcionados por el usuario. La formalización de estas, podríamos llamar, descripciones de *marcos de aplicación* [3] es una de las tareas a desarrollar dentro del proyecto COSACO.
- Es necesario disponer de una correspondencia entre requisitos del dominio CSCL y características funcionales de los componentes software CSCL disponibles. Éste deberá ser uno de los resultados obtenidos del análisis conjunto de pedagogos y tecnólogos del dominio CSCL dentro del proyecto COSACO.
- Como consecuencia de los dos puntos anteriores, habrá que formalizar los metadatos que permitiran caracterizar las particularidades de los componentes CSCL desarrollados. Dichos metadatos permitirán a la *herramienta de gestión de componentes* localizar aquellos que mejor se adaptan a los requisitos determinados por el usuario de la herramienta para un tipo de aplicación CSCL concreto.

5 Conclusiones y trabajo futuro

En este artículo se ha presentado el marco de trabajo y las primeras experiencias conducentes a la obtención de un sistema de gestión de componentes software para el dominio CSCL. Este esfuerzo se basa en la existencia de un grupo transdisciplinar que cubre las múltiples necesidades del dominio elegido. Esta experiencia, reflejada en el marco conceptual Delfos, permite analizar los retos del uso de la ISBC en un dominio suficientemente complejo e importante en condiciones reales.

En este sentido, se está procediendo a la identificación y creación de una biblioteca de componentes a partir de aplicaciones existentes o de nueva creación, como es el caso de los *puzzles colaborativos* presentados en este artículo. Este proceso se ha mostrado especialmente complicado por el problema de la

granularidad y de las exigencias de reutilización para atender a las diversas y cambiantes necesidades educativas. Por contra, el uso de patrones de diseño se ha mostrado especialmente útil.

En suma, se espera poder proponer, a medio plazo, una agrupación de componentes, patrones de diseño y metodologías suficientemente flexible para el dominio del aprendizaje colaborativo.

Este marco de desarrollo de aplicaciones CSCL se debe probar de forma exhaustiva tanto por nuestro grupo en otras aplicaciones del dominio como por otros grupos de la comunidad de proveedores de servicios y contenidos educativos.

Referencias

1. M. Cortes. A Coordination Language for Building Collaborative Applications. *Computer Supported Cooperative Work*, 9(1):5–31, 2000.
2. C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware. Some issues and experiences. *Communications of the ACM*, 34(1):9–28, jan 1991.
3. M.E. Fayad, D.C. Schmidt, and R.E. Johnson. *Implementing Application Frameworks*. John Wiley & Sons, 1999.
4. Object Management Group. *Model Driven Architecture (MDA)*. OMG document norms/2001-07-01, Julio 2001.
5. Object Management Group. *Unified Modeling Language, v1.4*. OMG document formal/2001-09-67, Septiembre 2001.
6. L.A. Guerrero and D.A. Fuller. A Pattern System for the Development of Collaborative Applications. *Information and Software Technology*, 43(7):457–467, 2001.
7. D.V. Johnson and R.T. Johnson. *Learning Together and Alone: Cooperative, Competitive and Individualistic Learning*. Allyn and Bacon, 1999.
8. D. H. Jonassen, K. L. Peck, and B.G. Wilson. *Learning with Technology: a constructivist perspective*. Prentice Hall, Inc., Upper Saddle River, NJ, 1999.
9. N. Kassem. *Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition*. Sun Microsystems, Octubre 2000.
10. T. Koschmann, editor. *CSCL:Theory and Practice of an Emerging Paradigm*. Lawrence Erlbaum, Mahwah, N.J, 1996.
11. P. Kruchten. *The Rational Unified Process*. Addison-Wesley, 1999.
12. Sun Microsystems. *Enterprise Java Beans Specification v1.1*. Accesible en http://java.sun.com/products/ejb/docs.html, Diciembre 1999.
13. Sun Microsystems. *Java 2 Enterprise Edition Platform Specificacion v1.3*. Accesible en http://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf, Julio 2001.
14. C.A. Osuna and Y.A. Dimitriadis. A Framework for the Development of Educational Collaborative Applications based on Social Constructivism. In *Proceedings of CYTED RITOS International Workshop on Groupware, CRIWG '99, Cancún, Quintana Roo, México*, pages 71 – 80. IEEE Computer Society Press, Septiembre 1999.
15. C.A. Osuna, Y.A. Dimitriadis, and A. Martínez. Using a Theoretical Framework for the Development of Educational Collaborative Applications Based on Social Constructivism. In *Proceedings of the ECSL01 European Conference on Computer-Supported Collaborative Learning*, pages 577 – 584, Maastricht, Holanda, Marzo 2001. Maastricht MacLuhan Institute. Accesible en http://www.mmi.unimaas.nl/euro-cscl/proceedings.

16. J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, J. Philips, N. Jackiw, and D. Suthers. Developing Educational Software Components. *IEEE Computer*, 32(9):50 – 58, Septiembre 1999.
17. J. Roschelle, J. Kaput, W. Stroup, and T.M. Kahn. Scalable Integration of Educational Software: Exploring The Promise of Component Architectures. *Journal of Interactive Media in Education*, (6):1 – 31, Octubre 1998. Accesible en http://www-jime.open.ac.uk/98/6.
18. M. Roseman and S. Greenberg. Building Real-Time Groupware with Groupkit, a Groupware Toolkit. *ACM Transactions on Computer-Human Interactions (TOCHI)*, 3(1):66–106, Marzo 1996.
19. J. M. Sancho. Más compacto, más rápido, más potente, más eficaz, más barato, ... In J. M. Sancho and L. M. Millán, editors, *Hoy ya es mañana. Tecnologías y Educación: un diálogo necesario*, pages 131–150. Publicaciones M.C.E.P., Sevilla, España, 1995.
20. Proyecto SEED del programa IST de la Unión Europea, 2002. Accesible en http://ilios.cti.gr/seed.
21. M. Sparling. Lessons Learned Through Six Years of Component-Based Development. *Communications of the ACM*, 43(10):47 – 53, Octubre 2000.
22. O. Stiemerling and B. Armin. The EVOLVE Project: Component-Based Tailorability for CSCW Applications. *AI & Society*, 14:120–141, 2000.
23. D. Suthers. Architectures for Computer Supported Collaborative Learning. In *Proceedings of the IEEE International Conference on Advanced Learning Tecnologies. Madison, Wisconsin, USA*, Agosto 2001.
24. C. Szypersky. *Component Software. Beyond Object-Oriented Programming.* Addisson Wesley, NY, USA, 1998.
25. Stanford University. Educational Software Components for Tomorrow, 2002. Accesible en http://www.escot.org/.
26. A. Vaquero. Las TIC para la Enseñanza, Formación y el Aprendizaje. *Novática*, 132:4 – 14, Marzo 1998.