Una Arquitectura SOA para sistemas de e-Learning a través de la integración de Web Services

Jorge Fontenla González
Universidad de Vigo
E.T.S.E. Telecomunicación
+34 986 814 073
jfontenla@det.uvigo.es

Manuel Caeiro Rodríguez
Universidad de Vigo
E.T.S.E. Telecomunicación
+34 986 813 468
Manuel.Caeiro@det.uvigo.es

Martín Llamas Nistal
Universidad de Vigo
E.T.S.E. Telecomunicación
+34 986 812 171
Martin.Llamas@det.uvigo.es

***Abstract*—En este artículo presentamos una nueva arquitectura de los denominados *Learning Management Systems* (LMSs) basada en la integración de herramientas de terceros accesibles como *Web Services*. Esta idea ya ha sido considerada como una vía prometedora de construir y extender sistemas de e-learning. No obstante, las propuestas existentes desembocan en una integración "suave" de LMSs y herramientas de terceros, que no es apropiada para contextos educativos. Básicamente, los LMSs pueden incluir herramientas de terceros pero no tienen ningún control sobre ellas. Por ello, los LMSs no pueden alterar el comportamiento de estas herramientas ni supervisar lo que los usuarios (e.g. los estudiantes) hacen con ellas. Para superar esta limitación hemos planteado una nueva arquitectura para LMSs que permite una integración "fuerte" de herramientas de terceros, mediante la que el LMS puede tener mayor control sobre la herramienta, permitiéndole gestionar diferentes aspectos de la misma (e.g. eventos, sesiones, permisos).**

I. Introducción

El vertiginoso avance científico y tecnológico de nuestra sociedad, ha llevado a organizaciones como universidades y empresas a proporcionar formación contínua a sus estudiantes y empleados a través de sistemas de e-learning conocidos como LMSs (*Learning Management Systems*). Estos sistemas se encargan de la administración, la provisión y el control de recursos y funcionalidades educativos.

Los LMSs actuales pueden ser considerados como aplicaciones Web complejas. Algunos ejemplos de LMSs son Moodle [1] o Blackboard [2]. Estos sitemas proporcionan un entorno centralizado para organizar información, soportar la comunicación entre profesores y estudiantes, facilitar el intercambio de documentos, responder cuestionarios online, etc. No obstante estos LMSs son demasiado genéricos. El problema "una talla no sirve a todos" es muy notorio en este ámbito, dado que las herramientas incluídas en los LMSs son demasiado generales y por tanto no se adaptan a necesidades específicas (e.g. los LMSs actuales no incluyen simuladores de terremotos que puedan ser empleados en un curso de geología). Además, los LMSs actuales son construidos como sistemas monolíticos, en los que resulta difícil separar en módulos cada pieza de funcionalidad. De esta forma, extender o modificar un LMS no es una tarea sencilla. Por ejemplo, resultaría complejo introducir el simulador de terremotos mencionado anteriormente en uno de los LMSs actuales. Estas limitaciones, que pueden ser descritas como deficiencias de adaptabilidad y de extensibilidad, están limitando el desarrollo y la usabilidad de los LMSs.

Estas deficiencias han sido identificadas como las principales carencias en los LMSs actuales, y algunas soluciones han sido propuestas pero con un éxito limitado. Por ejemplo, Moodle y Blackboard tienen la capacidad de extender sus funcionalidades mediante las llamadas "extensiones". No obstante, estos sistemas son diseñados primeramente como sistemas monolíticos, con la integración de herramientas externas considerada como un suplemento. Por ello, es posible incluir una nueva herramienta en Moodle o Blackboard, pero esto no es una verdadera integración. Además, el LMS carece de formas de monitorizar y controlar la forma en que los usuarios trabajan con las herramientas. Este es el caso, por ejemplo, cuando se quiere hacer un seguimiento de las actividades de los estudiantes.

Los problemas de adaptabilidad y extensibilidad encontrados en los LMSs existentes nos han llevado a concebir una arquitectura para mejorar la integración entre LMSs y herramientas de terceros. Esta propuesta está basada en las arquitecturas *Service Oriented Architecture* (SOA) [14], que permite al LMS integrar, configurar y usar herramientas de terceros expuestas como *Web Services* [14].

Este artículo está organizado como sigue. La Sección II analiza diferentes formas en las que un LMS puede ser extendido. A continuación la Sección III hace una revisión de tecnologías actuales relacionadas con la extensión de aplicaciones Web, tanto de propósito general como específicas del dominio del e-learning. La Sección IV describe el modelo de negocio encerrado en la arquitectura propuesta, que se detalla en la Sección V. Seguidamente, la Sección VI describe un ejemplo de uso de la arquitectura en un escenario real. Finalmente, la Sección VII concluye el artículo con algunas conclusiones.

II. Extendiendo un LMS

El problema básico de esta investigación ha sido extender la funcionalidad de un LMS a un coste mínimo. Además, la nueva funcionalidad debe integrarse apropiadamente con la ya existente en el LMS. Este problema puede ser considerado en el contexto más amplio de extender aplicaciones Web. En este punto se han identificado cuatro alternativas [6]:

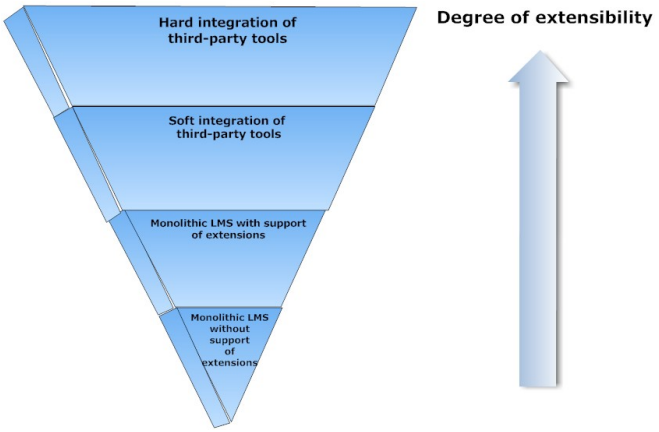

Fig. 1. Grados de extensibilidad en LMSs.

1) LMS monolítico sin soporte de extensiones. La funcionalidad del LMS sólo puede ser extendida modificando directamente su código fuente, lo que en el mejor de los casos resulta una tarea difícil.
2) LMS monolítico con soporte de extensiones. El LMS soporta la integración de nueva funcionalidad a través de plug-ins. La nueva funcionalidad está limitada por las posibilidades reconocidas originalmente en el LMS.
3) Integración suave de herramientas de terceros. La funcionalidad del LMS puede ser extendida mediante un hiperenlace a una herramienta (externa) de terceros. Una vez el usuario hace clic sobre él se muestra la interfaz gráfica de usuario de la herramienta. A partir de este momento los usuarios manejan un sistema que el LMS no puede controlar. Por tanto se añade nueva funcionalidad, pero con poco nivel de integración.
4) Integración fuerte de herramientas de terceros. Incluye la integración suave, pero permitiendo un mayor control sobre la herramienta integrada.

La Figura 1 muestra estas cuatro alternativas representadas en diferentes niveles de una pirámide invertida. El ancho de cada nivel de la pirámide es proporcional a la cantidad de nueva funcionalidad que puede conseguirse a determinado coste. Las soluciones basadas en herramientas de terceros tienen una mayor extension potencial que las otras. Dado que serían desarrolladas por terceros, únicamente se considera el coste de la integración pero no el de desarrollo. Para los desarrolladores del LMS, el coste de desarrollar tales herramientas es nulo.

La integración de herramientas de terceros es la base de las arquitecturas SOA. SOA es un concepto de arquitectura software que permite construir sistemas altamente escalables, basado en la invocación de funciones sin estado llamadas servicios. La forma más habitual de conseguir esto es mediante el uso de *Web Services*. Debido a este reducido nivel de acoplamiento es más plausible añadir nuevos servicios al sistema.

La integración fuerte de herramientas de terceros expuestas como *Web Services* encaja satisfactoriamente en nuestros propósitos, pues buscamos una solución que proporcione adaptabilidad y extensibilidad. La integración fuerte exhibe una ventaja importante sobre la integración suave, pues no solo implica la provisión de nueva funcionalidad sino también el control y supervisión de su comportamiento. La arquitectura propuesta en este artículo identifica diferentes aspectos que deben ser contemplados en este concepto de integración fuerte.

III. Propuestas existentes

Hoy en día existen diversas propuestas relacionadas con la integración de herramientas de terceros, en algunos casos expuestas como *Web Services*. No obstante esta integración se realiza usualmente a un nivel de integración suave, pues el nivel de control del sistema sobre la nueva funcionalidad es muy bajo.

Esta sección analiza las propuestas más interesantes. Comenzamos con aquellas soluciones para integrar herramientas de terceros en sitios Web de propósito general, para continuar con aquellas específicas del ámbito del e-learning.

A. Soluciones de propósito general

El impacto de la denominada Web 2.0 ha convertido Internet en una inmensa plataforma que proporciona almacenamiento de datos así como numerosas aplicaciones para manipularlos. Esta tendencia ha llevado a la aparición de las *Rich Internet Applications* (RIAs). Estas RIAs son posibles gracias al uso de algunas tecnologías de propósito general, algunas de las cuales se comentan a continuación: widgets y mashups.

Los widgets son una aproximación de propósito general que está creciendo en relevancia. Los widgets son pequeñas aplicaciones que típicamente proporcionan acceso a funcionalidades empleadas frecuentemente. Cada motor de widgets propietario tiene su propia API, lo que ha llevado a la especificación de widgets del W3C [3]. No obstante, aunque son un tipo de aplicación prometedor pretenden ser aplicaciones muy sencillas que no pueden cubrir muchas necesidades educativas. Además, hasta donde sabemos los motores actuales adolecen de control sobre los widgets, esto es, ocultan los detalles de lo que está ocurriendo "en el interior de la herramienta". Por ejemplo, no es posible cuántos usuarios están utilizando la herramienta simultáneamente.

Mencionamos también otro tipo de aplicaciones de propósito general: los mashups. Un mashup [4] son aplicaciones Web consistentes en una composición ad-hoc de servicios (datos o funcionalidad) de diferentes fuentes para crear servicios completamente nuevos. Los mashups están adquiriendo popularidad hoy en día, debido a aplicaciones Web ampliamente utilizadas como Google Maps. No obstante, no sólo presentan las mismas carencias que los widgets, sino que además la variedad potencial de mashups está limitada por la variedad de las fuentes.

B. Soluciones específicas del e-learning

La integración de herramientas de terceros en LMSs ha llamado la atención de entidades estandarizadoras del ámbito del e-learning, que han iniciado iniciativas para definir cómo

se debe producir dicha integración. Algunos resultados de este esfuerzo se describen a continuación.

La especificación más importante sobre el tema hoy en día es *IMS Tools Interoperability* (IMS-TI) [9]. IMS-TI hace uso de una combinación de *Web Services* y soluciones proxy para integrar herramientas externas en un LMS. La especificación se encuentra actualmente en su versión 1.0 y con la 2.0 bajo desarrollo.

En nuestra opinión IMS-TI presenta dos inconvenientes principales. El primero es que, a pesar de que permite la ejecución transparente de herramientas, no proporciona mecanismos para controlar y gestionar su uso, lo que redunda en que sólo permite alcanzar integración suave. El segundo es la ausencia de implementaciones de referencia que puedan ser empleadas para nuevos desarrollos. Tan solo existen unas pocas implementaciones, como el prototipo presentado en la "alt-i-lab 2005 Conference" [5], lo cual no es acorde con las pretensiones de IMS-TI.

Otra arquitectura propuesta para integrar herramientas en algunos LMSs es *CopperCore Service Integration* (CCSI) [8]. CCSI es una capa intermedia en estos LMSs entre la capa de presentación y las herramientas, cuya misión es adaptar las llamadas entre ambas. Esta transformación es posible porque CCSI exhibe una interfaz con un conjunto de métodos predefinidos para cada tipo de herramienta que puede ser accedido. Aunque la especificación de CCSI no menciona *Web Services*, pueden ser integradas si la adaptación de la llamada llevada a cabo por CCSI involucra el establecimiento de una conexión de Internet.

CCSI presenta importantes limitaciones que hacen que su aceptación no sea tan amplia como cabría desear:

- En primer lugar, como IMS-TI, ni proporciona mecanismos para controlar y gestionar el uso de las herramientas, ni permite supervisar la actividad de los usuarios.
- En segundo lugar, únicamente puede ser integrada una herramienta de cada tipo (e.g. no es posible integrar dos editores de texto diferentes simultáneamente), lo que reduce drásticamente las posibilidades del sistema de satisfacer las necesidades, preferencias y limitaciones personales de los usuarios.
- Finalmente, el propio diseño arquitectural de CCSI implica trabajo extra para los desarrolladores de aplicaciones, pues no solo deben suministrar las herramientas en sí sino también adaptadores para su integración con CCSI.

Podemos decir que IMS-TI y CCSI proporcionan únicamente integración suave de herramientas. Además, al contrario que las tecnologías de propósito general descritas en la Sección III-A están en estado de desarrollo temprano, y por tanto no hay sistemas reales que las empleen.

IV. Modelo de negocio

La principal idea que subyace este artículo es extender las funcionalidades de un LMS conectándolo con nuevas herramientas a medida que éstas estén disponibles. Estas herramientas pueden ser desarrolladas como herramientas de terceros y suministradas por servidores externos como *Web Services*, en concordancia con la aproximación SOA.

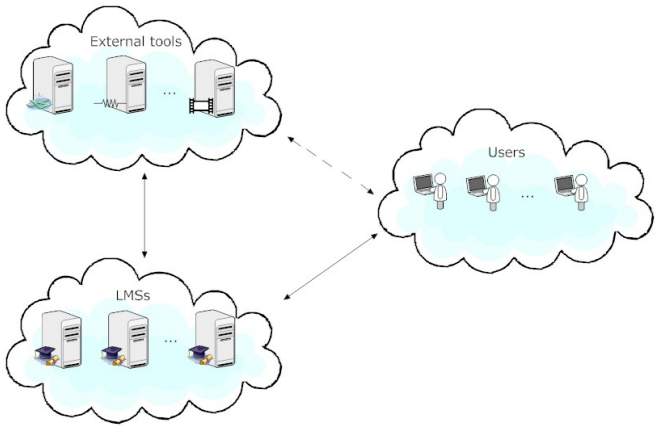

Fig. 2. Modelo de negocio.

La Figura 2 ilustra el nuevo modelo de negocio, en el que podemos ver tres tipos de proveedores involucrados representados como nubes: los LMSs, las herramientas educativas y los usuarios finales. Por un lado, los LMSs proporcionan el núcleo de funcionalidad (e.g. secuenciación de tareas, almacenamiento de datos personales) de las plataformas educativas. Junto con esta funcionalidad, los desarrolladores de LMSs adoptan algunas especificaciones para soportar la interacción con herramientas de terceros. Esta interacción se representa como una flecha que une la nube de los LMSs con la de las herramientas, pues hace posible la integración de cualquier par LMS–herramienta. Por otro lado, los desarrolladores de herramientas también deben adoptar estas especificaciones de interacción a la hora de crear sus productos. Es importante darse cuenta de que en este escenario las herramientas educativas son productos software autónomos que no precisan de un LMS para funcionar. En su lugar, los LMSs las usan para complementar sus funcionalidades. Finalmente tenemos los usuarios finales, que acceden a los LMSs y a las herramientas. Cuando un usuario accede a la herramienta ésta ha de comunicarse con el LMS para informarle de lo que el usuario hace y para permitir su control por parte del LMS.

Este modelo de negocio supone un giro copernicano respecto al modelo tradicional del ámbito del e-learning. Hasta ahora, el LMS ocupaba el centro del modelo de negocio pues incluía toda la funcionalidad. Con la aproximación de los *Web Services* se da la vuelta a este esquema, quitando al LMS del centro y concediendole la misma importancia que a las herramientas. Ahora el LMS puede utilizar muchas herramientas, y cada herramienta puede ser empleada por muchos LMSs. Esta solución implica que el desarrollo de LMSs y herramientas pueden seguir cursos separados.

Este modelo de negocio está relacionado con el incipiente concepto de *Cloud Computing* [13]. Esta tecnología está demostrando que muchas aplicaciones que previamente debían estar instaladas localmente pueden ser accesibles a través de Internet proporcionando una experiencia de usuario satisfac-

toria. La misma aproximación puede ser adoptada a la hora de cubrir la necesidad de herramientas en LMSs, donde los simuladores, editores de texto o herramientas de evaluación no tienen por qué ejecutarse localmente.

V. Integración fuerte de LMSs y herramientas de terceros

Las soluciones mencionadas en la Sección III muestran el interés existente en la integración de herramientas en LMSs. Un punto común a todas ellas es que no permiten más que integración suave. Este es un inconveniente principal desde el punto de vista de la adaptabilidad. No obstante estas soluciones suponen pasos en la dirección correcta.

La arquitectura propuesta en esta sección pretende ir un paso más allá. La integración suave sigue estando soportada, y representa el nivel más bajo de integración, pero son posibles niveles de integración adicionales. Estos niveles involucran aspectos como la autentificación y autorización, la notificación de eventos, la persistencia, la gestión de instancias y la invocación de métodos específicos. En esta arquitectura la integración fuerte de herramientas de terceros con un LMS es llevada a cabo en cada uno de estos aspectos, aunque es posible considerar niveles intermedios de integración tomando únicamente un subconjunto de esos aspectos.

Las siguientes secciones describen los principales componentes de la arquitectura. La descripción se organiza como sigue:

- Extensiones del modelo de datos para LMSs. Estas extensiones están dedicadas a mantener especificaciones sobre el control y supervisión de las herramientas.
- Extensiones de los componentes del LMS. Estos componentes son responsables del procesamiento de las correspondientes especificaciones del modelo de datos extendido.
- Extensiones de las interfaces de las herramientas. Estas herramientas también necesitan soportar interfaces dedicadas a permitir los diferentes niveles de integración con el LMS. El nivel al que una cierta herramienta puede ser integrada depende del soporte de estas interfaces.
- Un protocolo de gestión de la interacción que describa el esquema de integración básico entre LMSs y herramientas.
- Un protocolo de autentificación que soporte el acceso de los usuarios del LMS a la herramienta, sin necesidad de *sign-ons* adicionales.
- Un protocolo de gestión de instancias que describa cómo el LMS puede controlar las instancias de una herramienta.
- Un protocolo de gestión de sesiones que facilite la persistencia de los datos manejados en la herramienta.

A. Extensión del modelo de datos

La forma de realizar la integración de herramientas debe ser descrita y especificada apropiadamente. Por ejemplo, si el LMS está interesado en recibir información sobre las acciones de los estudiantes en un simulador es necesario indicar de qué información se trata. Por tanto, el modelo de datos existente en el LMS debe ser extendido de cara a soportar la especificación de los diferentes aspectos involucrados en la integración fuerte. Si alguno de estos aspectos no puede ser expresado en el modelo de datos del LMS, éste no podrá soportar el correspondiente nivel de integración.

Las extensiones al modelo de datos han sido producidas en el contexto de los LMSs basadas en Educational Modeling Languages (EMLs) [10]. Los EMLs son lenguajes y modelos de datos propuestos para realizar descripciones de unidades didácticas, que luego puedan ser ejecutadas por LMSs compatibles. Específicamente hemos adoptado el lenguaje Perspective-oriented EML (PoEML) [10]. La principal característica de PoEML es la separación del modelado en varias partes llamadas perspectivas que pueden ser abordadas separadamente. Aquí únicamente describiremos brevemente aquellas perspectivas que soporten la integración de herramientas de terceros:

- La perspectiva de Herramientas modela las caracteristicas de las herramientas necesarias en las unidades didácticas. Una de las características más originales de PoEML es que permite una caracterizar las herramientas de forma indirecta según sus aspectos funcionales (la funcionalidad esperada) y no funcionales (los permisos que permite conceder, los eventos que notifica, y los métodos públicos de que dispone). Más tarde el LMS es responsable de buscar e integrar una herramienta con tales características.
- La perspectiva de Autorización trata de la asignación de permisos a los participantes de la unidad didáctica. Esta perspectiva permite indicar qué permisos (e.g. lectura, escritura) deben ser asignados a cada participante a la hora de manejar una cierta herramienta.
- La perspectiva de Percepción se encarga del procesado de eventos. Modela la captura de eventos relevantes generados de la interacción de los participantes con las herramientas, de su procesado (e.g. filtrado, agregación) y su notificación a participantes interesados en ellos.
- La perspectiva de Interacción concierne la invocación automática y controlada de métodos públicos de las herramientas. Esta perspectiva involucra el conjunto de métodos que deben ser invocados en una cierta herramienta en un determinado momento. Los métodos pueden ser invocados cuando se reciben ciertos eventos, lo cual puede ser útil en ciertos escenarios.

La fuerte estructuración de PoEML hace de él un lenguaje adecuado para conformar el modelo de datos de un LMS. El LMS obtiene de cada una de las perspectivas toda la información concerniente al desarrollo de la unidad didáctica. No obstante, este artículo se centra en las perspectivas de Herramientas, Autorización, Percepción e Interacción, que mantienen la información para soportar la integración fuerte con herramientas. En cualquier caso, sería posible considerar un modelo de datos diferente, pero debería involucrar los mismos aspectos con el fin de permitir nuestro concepto de integración fuerte.

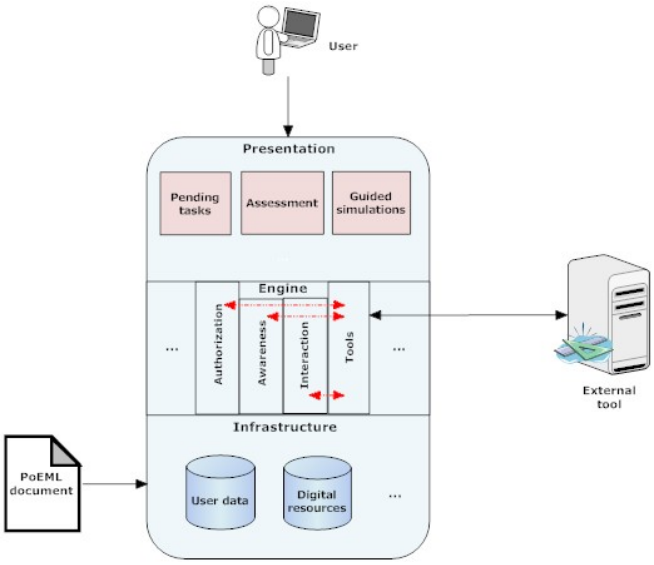

Fig. 3. Estructura de un LMS basado en PoEML.

B. Extensión de los componentes del LMS

El soporte de integración fuerte requiere nuevas funcionalidades en el LMS. Estas funcionalidades están dedicadas a controlar y supervisar las herramientas que deben ser integradas. La descomposición en perspectivas llevada a cabo en PoEML nos permite abordar el diseño de un LMS de forma modular, descartando el diseño monolítico de los LMSs actuales. En la Figura 3 se puede apreciar cómo el LMS puede ser construido en tres capas:

- La capa central es el Motor. Esta parte proporciona el núcleo de funcionalidad del sistema. El motor está compuesto de módulos independientes acordes a las perspectivas de PoEML. El módulo relacionado con la perspectiva de Herramientas es el responsable de gestionar la interacción y configuración de herramientas.
- Encima del motor se sitúa la capa de Presentación, en la que se pueden encontrar aquellas aplicaciones que conforman la interfaz de usuario del LMS.
- Bajo el motor podemos encontrar la capa de Infraestructura. Esta capa proporciona una serie de funcionalidades de almacenamiento y servicios de propósito general. Cabe mencionar que este motor es el que recibe el fichero en formato PoEML con la descripción del curso, que es procesada y ejecutada por el motor.

Dado el débil acoplamiento que existe entre las partes que conforman el motor, heredado de la que existe entre las perspectivas de PoEML, es posible desarrollarlas independientemente. De esta forma ninguno de los módulos correspondientes a otras perspectivas influye en la forma en se implementan que los módulos de las perspectivas de Herramientas, Autorización, Percepción e Interacción, los cuales son el foco de este artículo. Por tanto, en lo siguiente podemos ignorar la arquitectura del resto del LMS y centrarnos en:

- El componente de Herramientas, que se encarga de la interacción con las herramientas. Primeramente accede a las bases de datos de la infraestructura para obtener la URL de una determinada herramienta. A continuación lleva a cabo tareas de bajo nivel relativas a la autentificación (Sección V-E), gestión de instancias (Sección V-F) y persistencia (Sección V-G). Finalmente, acepta solicitudes de los componentes de Autorización y Interacción para asignar permisos e invocar métodos públicos respectivamente en la herramienta, y notifica al componente de Percepción de los eventos generados por la herramienta (véanse las flechas entre estos componentes en la Figura 3).
- El componente de Autorización, que gestiona la asignación de permisos a participantes en las herramientas.
- El componente de Percepción, que permite suscribirse a eventos de la herramienta, procesarlos cuando se reciban, y notificarlos a los usuarios interesados.
- El componente de Interacción, que permite invocar métodos en las herramientas.

C. Extensión de las interfaces de las herramientas

De forma similar a las extensiones consideradas en el LMS, las herramientas a ser integradas necesitan soportar interfaces específicas para permitir los correspondientes niveles de integración. Estas interfaces incluyen métodos que pueden ser invocados por el LMS para suscribirse a eventos, alterar el comportamiento de la herramienta, etc. Los siguientes puntos ofrecen una breve explicación de cada una de estas interfaces:

- Interfaz de autentificación. Proporciona el control de acceso básico a la herramienta. Incluye métodos que permiten a usuarios autorizados acceder a la herramienta, y a invitar a otros usuarios en su nombre.
- Interfaz de gestión de instancias. Permite controlar las diferentes instancias de la herramienta que pueden ser accedidas por un usuario. En esta interfaz podemos encontrar métodos para crear una nueva instancia para un usuario concreto en un determinado escenario educativo, para borrar instancias o para suspenderlas temporalmente.
- Interfaz de gestión de sesiones. Se encarga de la gestión de los datos generados por el usuario durante su interacción con la herramienta. Esto incluye retomar su trabajo con la herramienta en el punto en que lo dejaron, y transferir copias de respaldo de los datos desde la herramienta al LMS.
- Interfaz de gestión de permisos. Permite al LMS conceder y revocar permisos para usuarios específicos de la herramienta, sobre elementos de datos específicos y con una fecha de expiración específica.
- Interfaz de gestión de eventos. Proporciona métodos que permiten al LMS controlar los eventos disparados por la herramienta. Esto incluye recibir eventos por parte de la herramienta conteniendo información sobre las actividades de los usuarios y reenviar los eventos a participantes específicos.
- Interfaz de métodos especificos. Permiten acceder a la verdadera funcionalidad de la herramienta, y por tanto sus

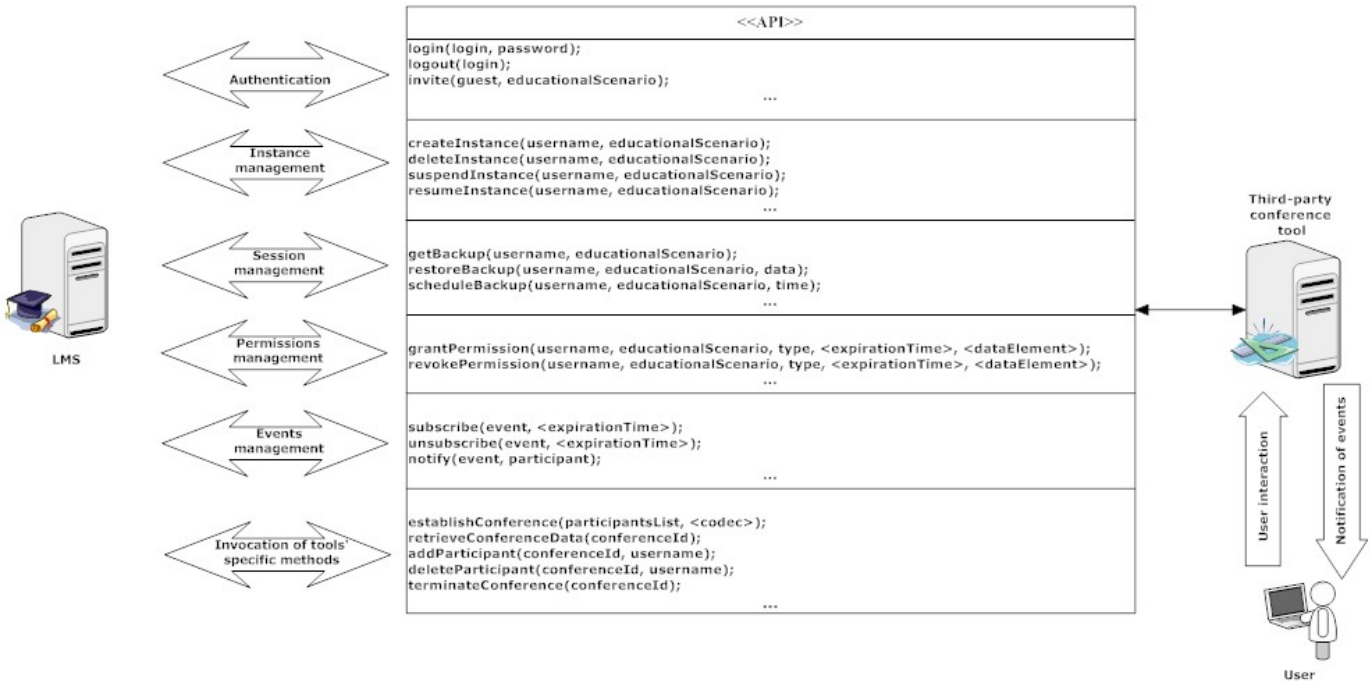

Fig. 4. Extensiones de las interfaces para herramientas de terceros, y su interacción con el LMS.

métodos varían de una herramienta a otra. Esta interfaz puede ser empleada para alterar el comportamiento que los usuarios perciben de la herramienta.

La Figura 4 muestra una representación esquemática de estas interfaces. Para la interfaz de métodos específicos se considera una herramienta genérica de conferencia (e.g. un chat). En la figura el usuario únicamente interactúa con la herramienta, pero en un escenario real también lo hace con el LMS. De hecho la interacción del usuario con la herramienta comienza cuando el LMS le proporciona un hiperenlace hacia ella.

Para evitar ambigüedades acerca del significado de los métodos específicos de las herramientas, éstos pueden ser descritos mediante una ontología. A efectos de pruebas hemos desarrollado una ontología sencilla enfocada en las herramientas más comunes empleadas en sistemas de e-learning [12].

D. Protocolo de gestión de la interacción

De forma previa al uso de una herramienta de terceros debe llevarse a cabo un proceso de integración de la misma. La Figura 5 muestra la orquestación que tiene lugar para integrar y usar una herramienta. El proceso comienza cuando el componente de Herramientas almacena su URL junto con sus características funcionales y no funcionales (1). Seguidamente un estudiante accede al LMS para iniciar o reanudar sus actividades (2). En caso de necesitarse, el LMS ofrece un hiperenlace a la herramienta emplazado en la página que se muestra al estudiante. Cuando éste pica en el enlace, se le muestra una ventana interna conteniendo la interfaz gráfica de usuario de la herramienta (3). Debido a las restricciones de seguridad de los iFrames se ha planteado el uso de la libreria

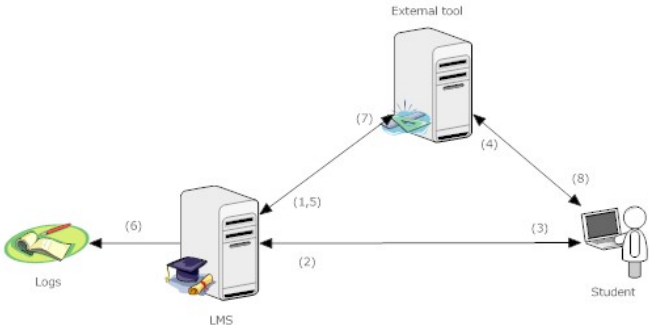

Fig. 5. Configuración y uso de herramientas de terceros.

de PHP cURL [11] para recuperar la página con la interfaz de la herramienta y emplazarla en el árbol DOM de la página del LMS.

Cuando el usuario accede a la herramienta (4) dispara eventos que contienen información relativa a las actividades que está llevando (e.g. “El usuario XXX ha entrado en la sala de chat”), los cuales son capturados por el LMS (5) y almacenados como logs (6). El módulo de Percepción puede filtrar estos eventos y notificarlos a participantes interesados (típicamente profesores). Tras la recepción de un evento el LMS puede invocar métodos públicos de la herramienta (7), para proporcionar al estudiante una experiencia más interactiva (8). Por ejemplo, cuando el LMS recibe el evento “El estudiante XXX ha accedido a la Lección 1” muchas veces seguidas puede interpretar que el estudiante tiene problemas de comprensión en la misma base de la materia, y arrancar una simulación guiada invocando el método público corres-

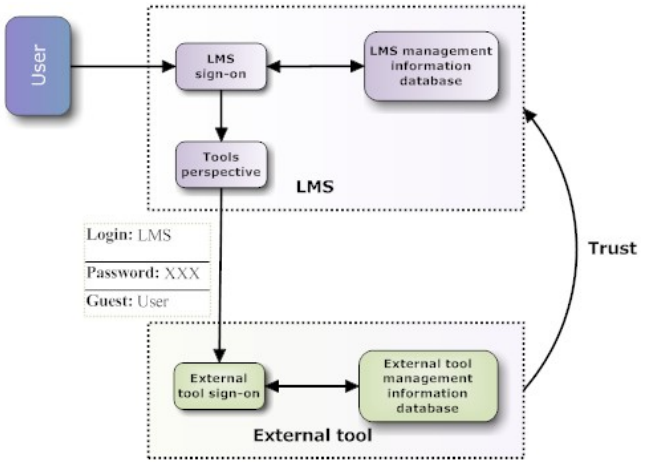

Fig. 6. Procedimiento de single sign-on.

pondiente en la herramienta.

E. Protocolo de autentificación

A pesar de su rigidez, las arquitecturas denominadas "LMS monolítico sin soporte de extensiones" y "LMS monolítico con soporte de extensiones" en la Figura 1 ofrecen una característica deseable. Una vez el usuario ha accedido al LMS no son necesarias autentificaciones adicionales para usar las herramientas. Esto es obvio, pues son parte del sistema al que el usuario acaba de acceder. Este procedimiento de autentificación, conocido como *single sign-on*, permite a los usuarios acceder a muchos servicios con un único conjunto de credenciales, liberándolos de la necesidad de recordar múltiples contraseñas y reduciendo el tiempo empleado para reintroducirlas.

Uno de los principales objetivos de una arquitectura de integración fuerte es obtener una integración transparente de diferentes sistemas, produciendo la ilusión de que son sólo uno. Por tanto, si no se implementase ningún procedimiento de *single sign-on* nuestros esfuerzos habrían sido en vano.

Nuestra solución se muestra en la Figura 6. Antes de que el usuario intente acceder, el LMS ha creado una cuenta de usuario para su uso propio en cada una de las herramientas que vayan a ser integradas. Cuando un usuario quiera acceder a la herramienta, el LMS envía a ésta un mensaje (apropiadamente asegurado empleando técnicas criptográficas) consistente en los siguientes campos: el nombre de usuario de la cuenta del LMS, su contraseña, el identificador de un invitado autorizado (el usuario), e información adicional sobre las credenciales de acceso (e.g. un periodo temporal de validez). La herramienta acepta esta invitación del usuario, pues mantiene una relación de confianza con el LMS. Nótese que durante este proceso no se envía información sensible relacionada con el usuario.

Finalmente la herramienta accede a la "External tool management information database" buscando una correspondencia entre el nombre contenido en el campo "Guest" y alguna instancia existente. Si se encuentra, el usuario final puede iniciar o retomar sus actividades.

F. Protocolo de gestión de instancias

Uno de los cometidos del componente de Herramienta es manejar las instancias de las herramientas. Entendemos por instancias un proceso en ejecución cuyos datos sólo pueden ser accedidos por usuarios autorizados. Dependiendo de la herramienta y de la unidad didáctica existen diferentes posibilidades en lo relativo a la gestión de instancias:

- Cada usuario es asignado a una nueva instancia por el componente de Herramientas en tiempo de ejecución. Es el caso más habitual.
- Todos los usuarios son asignados a la misma instancia. Por ejemplo, en el caso de una herramienta de preguntas frecuentes todos los usuarios deben ser asignados a la misma instancia.
- Los usuarios son asignados a diferentes instancias dependiendo de su pertenencia a un cierto grupo. Por ejemplo, los usuarios de cada grupo deben tener asignada una misma sala de chat para poder comunicarse.

Cuando un usuario (un único estudiante o un grupo) solicita una herramienta, el componente de Herramientas debe obtener una referencia a una instancia existente o bien crear una nueva.

G. Protocolo de gestión de sesiones

Otro aspecto importante es la persistencia de los datos entre diferentes sesiones con las herramientas. Un usuario debe ser capaz de retomar su trabajo en el mismo estado en que estaba cuando concluyó la última sesión (e.g. en el caso de un editor de texto debe ser posible continuar editando el mismo texto que en la sesión anterior). Adicionalmente, el LMS debe permitir al usuario que retome su trabajo empleando una herramienta diferente (pero equivalente) a la inicial. Este requisito se introduce para potenciar la capacidad del LMS de tratar con problemas de disponibilidad en las herramientas.

Por tanto, la solución adoptada ha sido el almacenamiento del lado del LMS de los datos de sesión. Esta solución tiene la ventaja de que el LMS tiene control sobre los datos de sesión, evitando de esta forma pérdidas de datos debidos a problemas de disponibilidad de las herramientas. No obstante, esto implica que tanto los LMS como las herramientas de terceros deben interactuar con el fin de almacenar y recuperar los datos de sesión. Para tratar con los problemas de disponibilidad de las herramientas se transfieren copias de los datos de éstas al LMS, periódicamente o al final de cada sesión. Estas copias pueden ser restauradas en la herramienta en caso de pérdida de datos.

VI. Juntando todo

El proceso que tiene lugar desde el punto de vista de los estudiantes cuando usan esta arquitectura comienza cuando éstos se ha autentificado en el LMS y solicita la página Web correspondiente a una parte concreta de una materia (e.g. una práctica especifica).

Supongamos que el estudiante quiere realizar una práctica de la asignatura "Dispositivos Electrónicos" con una herramienta de esquemáticos, para estudiar el factor de ruido de un amplificador operacional. La herramienta de esquemáticos

es accesible como un *Web Service* ajeno al LMS. Cuando al usuario se le muestra la página correspondiente a la práctica también se le proporciona un hiperenlace a la herramienta. El LMS captura el clic sobre el enlace, y procede al procedimiento de *single sign-on* descrito en la Sección V-E. El usuario puede tener o no una instancia en la herramienta, pero en todo caso el protocolo de gestión de instancias hace el trabajo duro por él. Cuando el usuario está correctamente autentificado en la herramienta el LMS recibe un aviso, lo que le permite acceder a la página de la herramienta vía cURL y emplazarla en la página del LMS.

El trabajo con la herramienta tiene lugar a través de su interfaz gráfica de usuario. Sus elementos gráficos (botones para instanciar amplificadores operacionales, para ejecutar una simulación, etc.) están asociados a funciones JavaScript responsables de invocar métodos remotos en el *Web Service*. En la comunicación entre el usuario final y el *Web Service* el LMS no toma parte; no obstante, el LMS recibe eventos de la herramienta conteniendo información relativa a qué hace el usuario, que usa para alterar su comportamiento de forma controlada.

Cuando la tarea está completada, el usuario invoca el correspondiente método "Logout" con el que la herramienta termina la sesión y envía el correspondiente evento al LMS. Este evento permite a éste no sólo borrar la interfaz de la herramienta del árbol DOM de la página de la unidad didáctica, sino también solicitar una copia de los documentos que el usuario ha producido en la última sesión. El estudiante puede navegar de nuevo en la lista en árbol de lecciones y prácticas listada en el LMS, y repetir el proceso con otra actividad y otra herramienta.

VII. Conclusiones

Los LMSs actuales están jugando un papel importante a la hora de proporcionar acceso a contenidos educativos por todo el mundo. No obstante, sus posibilidades están limitadas debido al problema "una talla no sirve a todos". Estas limitaciones han sido el punto de partida de nuestro trabajo. A diferencia de soluciones actuales, que sólo permiten integración suave de herramientas de terceros, nuestra arquitectura permite diferentes niveles de integración.

Nuestra propuesta no sólo implica el diseño de nuevos sistemas de e-learning, sino también un modelo de negocio completamente nuevo en el que el desarrollo de LMSs y herramientas educativas pueden seguir caminos separados (aunque complementarios). En último término este modelo de negocio implica nuevas oportunidades para ofrecer a los estudiantes una educación mejor y más aplicada. Se trata de una aproximación prometedora para solventar las carencias de iniciativas existentes en el campo de la integración de herramientas de terceros, pues proporciona algunas ventajas:

- En primer lugar, los desarrolladores software pueden especializarse y centrar sus esfuerzos bien en el LMS, bien en las herramientas externas. Esto implica costes de desarrollo menores y un menor tiempo en el lanzamiento de nuevas versiones.
- En segundo lugar, permite desarrollar herramientas ad-hoc para una unidad didáctica concreta, y usarlas en diferentes LMSs.
- En tercer lugar, los profesores pueden elegir las herramientas más adecuadas para las unidades didácticas de entre un amplio número de herramientas, pues ya no serían exclusivas de un LMS concreto.
- Finalmente, hace posible construir LMSs que soporten un mayor número de usuarios, pues la carga computacional estaría dispersa entre los servidores del LMS y de las herramientas.

Este modelo de negocio está todavía en sus primeros pasos, pues aún está por ser fundada una comunidad mundial de desarrolladores de herramientas externas. No obstante, esperamos que nuestra arquitectura pueda ser útil para aumentar la adaptabilidad y extensibilidad de los LMSs.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia bajo la subvención TIN2007-68125-C02-02, y por la Consellería de Innovación e Industria bajo la subvención PGIDIT06PXIB32 2270PR. Asimismo, los autores quieren agradecer al Ministerio de Ciencia e Innovación de España y al Plan Nacional Español I+D+I 2008-2011 el apoyo a este artículo dentro del proyecto RedOBER - Proyecto TSI2007-31091-E Objetos Educativos Reutilizables (para el EEES en las especialidades de las Tecnologías de la Información y las Comunicaciones), y a la acción de coordinación del CYTED código 508AC0341 "Software Libre en Teleformación".

References

[1] Sitio Web de Moodle. Accedido en Febrero de 2009 en: http://moodle.org/
[2] Sitio Web de Blackboard. Accedido en Febrero de 2009 en: http://www.blackboard.com/
[3] Especificación de widgets del W3C. Accedido en Febrero de 2009 at: http://www.w3.org/TR/widgets/
[4] S. Weber, L. Thomas, E. Ras, "Investigating the suitability of mashups for informal learning and personal knowledge management", *Proceedings of 1st Workshop MUPPLE'08, Maastritch, 2008.*
[5] Demostración alt-i-lab 2005. Accedido en Febrero de 2009 en: http://zope.cetis.ac.uk/content2/20050901184010
[6] M. Kyng, "Computers and Design in Context", The MIT Press, 1997.
[7] C. Severance, "Functionality mash-up - Evolving to the next generation of learning management systems". Accedido en Febrero de 2009 en: http://www.ja-sig.org/wiki/download/attachments/13567279/2008-04-30-jasig-severance.pdf?version=1
[8] H. Vogten, H. Martens, R. Nadolski, C. Tattersall, P. van Rosmalen, R. Koper, "CopperCore Service Integration - Integrating IMS Learning Design and IMS Question and Test Interoperability". Accedido en Febrero de 2009 en: http://ieeexplore.ieee.org/iel5/10997/34637/01652450.pdf
[9] Especificación IMS Tools Interoperability. Accedido en Febrero de 2009 en: http://www.imsglobal.org/ti/index.html
[10] M. Caeiro, "PoEML: A separation-of-concerns proposal to instructional design", *Handbook of visual languages for instructional design: theory and practices*. Editado por L. Botturi y T. Stubbs, IGI Global, 2007.
[11] Sitio Web de cURL. Accedido en Febrero de 2009 en: http://es.php.net/curl
[12] J. Fontenla, "Introducción a una ontologia de herramientas educativas". Documento interno.
[13] G. Gross, "Google, IBM Promote Cloud Computing", *PC World, 2007.*
[14] F. Coyle, "XML, Web Services and the data revolution", Addison-Wesley Professional.
[15] Sitio Web de PHP. Accedido en Febrero de 2009 en: http://www.php.net/