UNIFICACIÓN DE CONTRATOS EN PLATAFORMAS DE E-LEARNING PARA ARQUITECTURAS DE INTEGRACIÓN

FRANCISCO HUERTAS FERRER

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA, FACULTAD DE INFORMÁTICA,
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster Sistemas Inteligentes

07/09/2012

Director:

Antonio Navarro Martín

Autorización de Difusión

FRANCISCO HUERTAS FERRER

07/09/2012

El/la abajo firmante, matriculado/a en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: “Unificación de contratos en plataformas de e-learning para arquitecturas de integración”, realizado durante el curso académico 2011-2012 bajo la dirección de Antonio Navarro Martín en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

La integración de plataformas de e-learning se ha convertido hoy en día en una necesidad patente. Sin embargo los mecanismos de integración de estas plataformas en muchas ocasiones no están bien descritos, mientras que en otras ocasiones son insuficientes. Este trabajo analiza la disponibilidad de servicios web en tres de las principales plataformas de e-learning y desarrolla un conjunto de servicios web unificados con el fin de facilitar el uso de dichas plataformas en una arquitectura de integración.

Palabras clave

LMS, SOA, servicios web, Blackboard Learn, Moodle, Sakai

Resumen en inglés

Integration of e-learning platforms is a key issue today. However, the integration mechanisms included in these platforms are now well described in some cases, and are not enough in other cases. This work analyses the availability of web services in three of the most important e-learning platforms, and develops a set of unified web services to ease the use of these platforms in integration architectures.

Keywords

LMS, SOA, web services, Blackboard Learn, Moodle, Sakai

Índice de contenidos

Índice de figuras

Índice de Tablas

1 Introducción

En los últimos años, el e-learning ha tomado un papel relevante en el contexto educativo. Este tipo de aprendizaje cubre una amplia gama de aplicaciones y procesos de aprendizaje basados en computadoras o web, aulas virtuales, colaboración digital, etc. Los canales de comunicación en los que se apoya el e-learning son varios incluyendo Internet, intranet/extranet, radiodifusión, TV, suporte multimedia, etc. [1].

El éxito del e-learning ha promovido la aparición de campus virtuales, plataformas que integran procesos de comunicación e intercambio de información, procesos de gestión y administración, procesos de investigación, etc. Estas plataformas integran gran cantidad de tecnologías de la información.

Originalmente los campus virtuales estaban basados en una sola plataforma e-learning o *Learning Management System* (LMS). Sin embargo, en la actualidad, los campus virtuales utilizados por grandes instituciones se han convertido en complejas aplicaciones de dimensión industrial [2], con importantes necesidades de integración entre distintos LMSs.

El uso de distintos LMSs en un mismo campus virtual proporciona diferentes beneficios:

- Permite que cada usuario utilice la plataforma de e-learning que mejor se adecúa a sus necesidades.
- Elimina la dependencia de la institución educativa con respecto a un único proveedor de servicios.
- Favorece el cambio de plataforma, al poner al servicio de los usuarios diversas plataformas de e-learning con las que pueden experimentar.
- Permite tener una experiencia práctica con las posibilidades reales de cada plataforma, permitiendo prestar mayor atención a la que mejor se adapta al mayor número de usuarios, y seleccionar versiones futuras de cada plataforma desde la experiencia de la plataforma actual.

Sin embargo, la presencia de diversas plataformas simultáneas también presenta algunos inconvenientes:

- Se hace necesario el desarrollo de software que facilite el uso integrado de las diferentes plataformas.
- Requiere un mayor esfuerzo de mantenimiento hardware y software.

- Requiere un mayor esfuerzo de formación entre los usuarios de las plataformas.

Sin embargo, nuestra experiencia en el Campus Virtual de la UCM, nos ha demostrado que las ventajas de contar con distintas plataformas superan sus inconvenientes [3].

Este trabajo se centra en proporcionar una infraestructura software para el uso integrado de distintas plataformas de e-learning en el contexto de un campus virtual.

En general, para conseguir esta integración, las plataformas e-learning caben al menos utilizar dos aproximaciones:

- *Interfaces de programación de aplicaciones* (API), que ofrezcan la funcionalidad de la plataforma e-learning a aplicaciones implementadas en el mismo lenguaje de programación que la propia plataforma.
- *Servicios web* que además de ofrecer funcionalidades que la plataforma posee, favorecen la integración entre plataformas heterogéneas.

Esta última línea de desarrollo, mediante servicios web, ofrece, por tanto, un nivel más de integración debido a la abstracción, no solo del lenguaje de la plataforma, sino también por su capacidad de ofrecer las funcionalidades entre plataformas mediante un canal de red como internet.

Sin embargo, exceptuando proyectos independientes a las plataformas [4], la evolución de la integración está dirigida a un ámbito en que la plataforma educativa representa el centro de dicha integración, y son las aplicaciones externas las que se adaptan a cada una la plataformas mediante las API's o servicios web que, en particular, cada plataforma ofrece.

Las aportaciones de este proyecto buscan dar un paso más en la integración entre LMS y plantean los siguientes objetivos:

- Analizar los servicios web disponibles en tres de las principales plataformas de e-e-learning del mercado (Blackboard Learn 9.1, Moodle 2.0 y Sakai 2.7) con el fin de conocer el estado de implementación de los mecanismos de integración SOA de dichas plataformas.
- Crear un mecanismo de intercomunicación entre plataformas basado en un conjunto de servicios web. Estos servicios web representan un grupo de funcionalidades que permiten interactuar con las principales plataformas de e-learning. Este esfuerzo permitirá una unificación de contratos en plataformas de e-learning para arquitecturas de integración.

- Implementar los servicios web canónicos definidos en este trabajo en tres de los LMS más importantes del mercado (Blackboard Learn 9.1, Moodle 2.0 y Sakai 2.7).
- Crear un entorno que, mediante los servicios web creados, sea capaz de interactuar entre las distintas plataformas indistintamente. Es decir, tal y como se define en [3], crear un *campus virtual de marca blanca*. Dicho campus virtual aísla al usuario de un LMS concreto, recubriendo a dicho LMS con una interfaz de usuario común que implementa sus funciones de e-learning en base a los servicios web canónicos definidos en esta memoria.

Hasta cierto punto, parte de los objetivos de este trabajo fueron abordados en un trabajo de la asignatura Sistemas Informáticos dirigido por el profesor A. Navarro en el curso pasado [5]. Con respecto a dicho trabajo, el presente trabajo:

- Contextualiza el trabajo, comparándolo con el proyecto de mayor relevancia en el área: el Campus Project[6].
- Aumenta y homogeneiza los servicios web canónicos.
- Mejora sensiblemente la gestión de usuarios y sesiones.
- Utiliza una implementación basada en servicios web de las plataformas, en vez de en sus APIs.
- Proporciona una implementación para Blackboard Learn 9.1, y mejora las disponibles para Moodel 2.0 y Sakai 2.7.

De hecho, parte de estos resultados ya han sido publicados en un congreso internacional con actas publicadas por IEEE [7] y seleccionado posteriormente para su publicación en una revista internacional [8].

Cabe destacar que este trabajo se ha realizado en el contexto del proyecto de investigación *Arquitecturas Avanzadas en Campus Virtuales*, AACV (TIN2009-14317-C03-01), que busca el desarrollo de una infraestructura software para la construcción de campus virtuales de nueva generación. Dos puntos fundamentales de este proyecto son el uso de SOA para independizar a los campus virtuales de sus plataformas de e-learning, y el uso de MDA para favorecer la evolución de dichos campus virtuales. Por tanto, el trabajo recogido en esta memoria será aprovechado directamente por el proyecto AACV.

Capítulo 1 Introducción

La presente memoria se organiza en diversas secciones. La sección 2 muestra el trabajo relacionado, incluyendo una introducción a SOA, servicios web, OKI y un análisis de la disponibilidad de servicios web en diversos LMSs. La sección 3 analiza diversas arquitecturas propuestas para campus virtuales, proponiendo la que se utilizará en esta memoria, y definiendo los servicios web canónicos utilizados en dicha arquitectura. La sección 4 compara el trabajo realizado en esta memoria con el Campus Project. Finalmente, la sección 5 presenta las conclusiones y el trabajo futuro.

2 Trabajo relacionado

La integración entre plataformas e-Learning se ha convertido en un campo de investigación a causa de la gran cantidad de plataformas que existen actualmente [3] [7] [8]. Así, surgen proyectos que buscan ofrecer alternativas de integración que ayuden a las instituciones a la hora migrar a plataformas más indicadas o usar varias plataformas simultáneamente.

Estos proyectos están fuertemente ligadas con arquitecturas multicapas [9] y orientadas a servicios [10]. El concepto de utilizar capas de servicios para dar soporte al negocio de las plataformas, se adapta a las necesidades de integración planteadas.

2.1 Arquitectura Orientada a Servicios de Cliente

La arquitectura orientada a servicios (*Service Oriented Architecture*, SOA)[10] es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

SOA es una arquitectura desacoplada de componentes de software que proveen funciones específicas (proveedor) y que pueden ser invocadas por otros componentes (consumidor) independientemente de la plataforma en que se encuentren ambos.

Esta arquitectura permite descomponer las funcionalidades que una organización requiere en partes reutilizables, haciendo más sencillo e independiente su manejo. Un servicio puede contener otros servicios, convirtiéndolo en un modelo iterativo.

SOA se define como un tipo de sistema distribuido, conjunto de agentes software que colaboran entre sí para implementar una determinada funcionalidad, ya que los agentes rara vez trabajan en el mismo entorno, necesitando alguna forma de comunicación entre ellos. En SOA los agentes son servicios que realizan una operación determinada y que puede ser invocado desde afuera del contexto de una gran aplicación [11].

2.1.1 Capas SOA

SOA define las siguientes capas de software, que pueden verse en la Figura 2.1 [10][11]:

- Capa 1: Sistemas operacionales. Esta capa contiene todas las aplicaciones existentes tanto *Enterprise Resource Planning* (ERP), Client Relasonship

Management (CRM), sistemas orientados a objetos, aplicaciones de inteligencia de negocio, etc., SOA puede reutilizarlos e integrarlos.

- Capa 2: Componentes empresariales. Estos componentes realizan la funcionalidad y mantenimiento de la calidad del servicio que se brinda.
- Capa 3: Servicios. Los servicios de negocio residen en esta capa. Pueden ser descubiertos o pueden ser enlazados estáticamente y después invocados en servicios compuestos. Esta capa de exposición de servicios también permite tomar componentes que se encuentran en la capa 2.
- Capa 4: Composición de procesos de negocio. A partir de los servicios de la capa 3 esta capa define la composición de procesos de negocio. Los servicios, para ejecutar un proceso de negocios son introducidos a través de orquestación y coreografía, conceptos que serán explicados más adelante.
- Capa 5: Presentación o acceso. Esta capa se consideraba fuera del ámbito de la SOA, pero estándares como *Web Services for Remote Portlets* versión 2.0 [12].
- Capa 6: Integración de servicios (ESB). Posibilita la integración de servicios a través de la introducción de un conjunto fiable de capacidades.
- Capa 7: Calidad del servicio. Esta capa brinda las capacidades necesarias para monitorizar, gestionar y mantener propiedades de calidad del servicio, como seguridad, ejecución y disponibilidad.

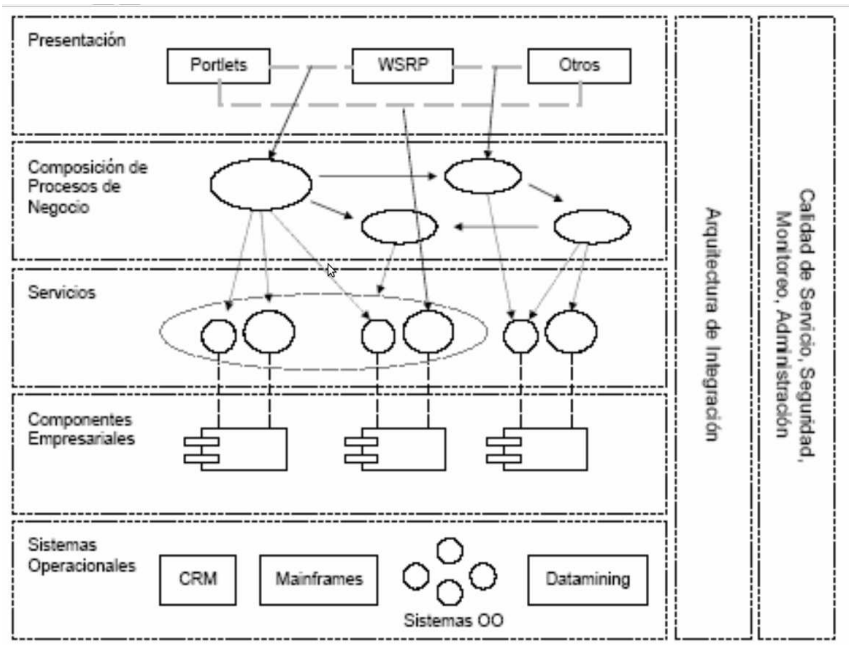


Figura 2.1 Capas de SOA.

2.1.2 Propiedades de SOA

A continuación se analizan las principales propiedades que definen una arquitectura SOA son [10][11].

Orientación a la conversación. El foco de atención no está en los nodos sino en los mensajes que se intercambian entre los mismos.

Vista Lógica: El servicio es una abstracción (vista lógica) de los programas, bases de datos, procesos de negocio, etc.

Orientación a Mensajes: El servicio se define formalmente en términos de los mensajes que se intercambian entre agentes proveedores y solicitantes. Esto posibilita que se incorpore un componente "decorando" estos componentes con software de gestión y conversión.

Abstracción del agente. La estructura interna del agente (lenguaje de programación, BD, etc.) se abstrae en SOA, así un nodo es una entidad computacional que participa en conversaciones con otros nodos. No es necesario conocer las particularidades del lenguaje de implementación. Esto evita problemas arquitectónicos derivados de la necesidad de conocer determinados sistemas a nivel estructural.

Metadatos. SOA se asocian con metadatos, los mismos son descripciones acerca de la forma y tipo de los elementos que transportan los mensajes, el orden de los mensajes, el significado de los mensajes, etc.

Orientación a la Internet: Los servicios tienden a usarse a través de la red, aunque este no es un requisito absoluto.

Granularidad: Los servicios tienden a usar un pequeño número de operaciones con mensajes complejos.

Neutralidad de Plataforma: Los mensajes se envían en un formato estándar y neutral a la plataforma, utilizando XML como mecanismo de comunicación entre plataformas.

2.2 SOA en aplicaciones web: servicios web.

La *World Wide Web* (WWW) o simplemente web presenta un sistema de distribución de información accesible a través de Internet. Las aplicaciones web representan los componentes software de Internet. SOA es un tipo de arquitectura que, implantada en aplicaciones web, ofrece sus funcionalidades a través de la red. El componente con el que ofrece los servicios se denomina servicio web.

Un servicio web [13] se define, según el Word Wide Web Consortium, como un componente software identificado por una URI [14], cuyas interfaces y vinculaciones son capaces de ser definidas, descritas y descubiertas como artefactos XML [15]. Los Servicios Web soportan la interacción con otros agentes software mediante el intercambio de mensajes basado en XML a través de protocolos basados en Internet.

Los servicios web ofrecen, frente a otro tipo de proveedores de funcionalidad de aplicaciones, un método de comunicación desvinculados del lenguaje de programación y que permite arquitecturas distribuidas. También cubren la necesidad de las aplicaciones web de ofrecer su funcionalidad mediante componentes externos y ocultar los detalles de implementación. Y gracias a este tipo de arquitectura se pueden crear plataformas modulares y es posible extender las funcionalidades de las aplicaciones. Los servicios web, además, facilitan la integración entre aplicaciones web.

2.2.1 Tipos de servicios web

Existen distintas formas de implementar los servicios web. Hay dos características principales a la hora de clasificar estas implementaciones: la forma de invocar el servicio y el mecanismo de codificación de información. Así, algunas implementaciones utilizan directamente conexiones HTTP para invocar los servicios, mientras que otras permiten invocar directamente los servicios en términos del lenguaje del cliente utilizando *proxys* [16]. En lo referente a la representación de información, algunas implementaciones codifican los datos en formato XML, que es enviado directamente entre cliente y servidor, mientras que otras utilizan mecanismos de conversión XML a objetos para manejar directamente los datos en el lenguaje del cliente. Entre estas implementaciones cabe destacar:

- *XML-Remote Procedure Call* (XML-RPC) [17]: Esta tecnología usa XML y HTTP para tener acceso a los servicios ofrecidos por el servidor. La comunicación se realiza mediante mensajes que son enviados entre los procesos del cliente y el servidor.
- *Simple Object Access Protocol* (SOAP) [18]: Esta tecnología surge al añadir caracteristicas más avanzadas a XML-RPC, comparte por tanto su método de comunicación, añadiendo la posibilidad de definir estructuras avanzadas, descripciones de los servicios mediante Web Services Description Language (WSDL).
- *Representational State Transfer* (REST) [19]: Los servicios web basados en esta arquitectura intentan emular al protocolo HTTP o similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar (por ejemplo GET, PUT, PUSH y DELETE). Este tipo de servicios web se centra en interactuar con recursos con estado.

2.3 Open Knowledge Initiative (OKI)

El proyecto OKI [4] fue patrocinado por la Fundación Mellon y liderado por el Instituto Tecnológico de Massachussets (*Massachussets Institute of Technology*, MIT 2002), en colaboración con la Universidad de Stanford y la participación de varias universidades americanas, entre ellas la Universidad de Dartmouth, la Universidad de Harvard, la Universidad

de Carolina del Norte, la Universidad de Michigan, la Universidad de Pensilvania y la Universidad de Wisconsin.

El objetivo principal del proyecto fue cubrir la necesidad básica para la comunidad educativa en la Web: diseñar y desarrollar una arquitectura abierta y extensible para las plataformas educativas.

Como resultado del proyecto se especificó un marco común para la arquitectura de las plataformas educativas definido mediante interfaces de programación denominadas Open Service Interface Definitions (OSID) [20].

La arquitectura de OKI se caracteriza por tener dos niveles de interfaces independientes, compuestas de un conjunto de servicios básicos que pueden ser implementados mediante APIs o servicios web. La primera de estas capas se denomina *common services* y da soporte al conjunto de servicios básicos o de infraestructura de una plataforma (administrativos, de autenticación, de definición de roles de usuario, etc.). La segunda capa ofrece servicios de tipo educativo y define el conjunto de servicios que tienen que ver con la función educativa de la plataforma. Esta capa tiene como objetivo conseguir un conjunto que ofrezca una funcionalidad que no esté ligada a una plataforma concreta, sino crear que sea una capa de enlace que independice el software educativo de la infraestructura, de tal manera que los cambios en un módulo determinado no afecten al resto.

Al definir las plataforma educativa como un conjunto de servicios básicos definidos en la interfaz OSID e implementados mediante API’s o servicios web independientes, se adopta una arquitectura SOA logrando interoperabilidad entre aplicaciones a través de esta interfaz común que definen un conjunto de operaciones.

2.3.1 Definición de servicios OSID

OKI define un conjunto de servicios OSID, estos servicios pueden ser implementados en cualquier lenguaje mientras respeten la definición dada por OKI. A continuación se va a dar una definición general de los OSID [20].

Agent

El *agent* OSID es el encargado de definir agentes como usuarios y grupos. Recibe este nombre porque define como crear, modificar, eliminar y acceder a los agentes. Un grupo es un

agente que puede contener otros agentes, incluyendo grupos. Este servicio también define cómo se crea y eliminan un grupo, así como la gestión de los miembros que pertenecen a él.

Assement

El *Assement* OSID da apoyo a la creación, organización, administración, evaluación, almacenamiento y recuperación de evaluaciones. Las evaluaciones se organizan en secciones y las secciones en elementos. Una vez realizada, la evaluación puede ser publicada, en cuyo caso se queda pendiente de lectura por parte de un agente (Estudiante). Cada evaluación, la Sección o elemento puede tener una calificación.

El componente *AssessmentManager* realiza un seguimiento de las evaluaciones, así como los tipos soportados para la implementación en particular.

Se pueden administrar los componentes de este servicio desde cualquier otro a cualquier nivel. Por ejemplo, una aplicación puede administrar el contenido de una evaluación, el conjunto de preguntas, etc. El componente *PublishedAssessment* puede incluir opcionalmente referencias a al componente CourseSection perteneciente a CourseManagement OSID y al componente *GradableObject* perteneciente al *grading* OSID. *Grading* OSID es el servicio apropiado para almacenar el conjunto de evaluaciones.

Authentication

El *Authentication* OSID realiza principalmente la invocación del proceso de autenticación. La implementación de este servicio es la responsable de reunir la información que sea adecuada para realizar la autenticación. El servicio también permite comprobar si un usuario está autenticado, obtener el identificador correspondiente al agente del usuario autenticado y los tipos de autenticación posibles, y la eliminación de una autenticación previa.

Los demás servicios pueden interactuar con la información y recursos sobre los que se requiere algún tipo de control de acceso.

Authorization

Los servicios disponen de funcionalidad que pueden involucrar a contenido y operaciones restringidas. *authorization* OSID proporciona una manera de definir quién está autorizado para acceder a los servicios y cuando puede hacerlo.

Cada agente tiene asociada una a más autorizaciones, que representa las funciones de usuario en el sistema. Antes de realizar una operación, las implementaciones consultan si existe una autorización que permita realizar la operación. Si el agente está autorizado, la operación se puede realizar. Si no es así, la operación no está permitida.

CourseManagement

El *CourseManagement* OSID realiza principalmente la creación y gestión de cursos representados por el componente *CourseCatalog*. Los tipos de cursos que se pueden crear en el componente son:

- *CanonicalCourses*, representan la descripción de los cursos generales que son independientes a los periodos lectivos y permanecen aunque estos periodos finalicen.
- *CourseOfferings* representan la relación entre los cursos descritos en *CanonicalCourses* y un periodo lectivo concreto al que están asociados. Estos cursos desaparecen cuando acaba el periodo lectivo al que están asociados.
- *CourseSections* representan los diferentes grupos que puede tener un *CourseOfferings*, Tienen asociados el lugar donde se producen, calendario, lista de estudiantes, etc.

Dictionary

El *Dictionary OSID* soporta la creación y completado de diccionarios compuesto por pares de variable y valor. Cada diccionario posee un nombre y una descripción, así como un dominio del diccionario. Varios diccionarios pueden compartir el mismo nombre si se encuentran en dominios diferentes. Este servicio puede ser utilizado para la localización y el mapeo de un conjunto de valores de un dominio entre dominios.

Los usos de *Dictionary OSID* son variados y es utilizado por el resto de servicios, como los valores de retorno de los servicios, este puede ser distinto dependiendo del dominio que se esté utilizando.

Filing

El *Filing OSID* proporciona almacenamiento independiente de la plataforma y la gestión de estos archivos y directorios. Los archivos y directorios pueden estar asociados con los

metadatos del propietario, tipo MIME, cuotas y versiones. Las implementaciones pueden ser construidas utilizando los sistemas de archivos, bases de datos, o una combinación de ambas.

Grading

El Grading OSID proporciona una manera para gestionar grados, esto incluye operaciones de caracterización, almacenamiento y recuperación. Un grado está caracterizado por cuatro elementos: valor (*GradeValue*), tipo (*GradeType*), Nivel (*GradeScale*) y puntuación (ScoringDefinition). Estos cuatro elementos proporcionan una manera genérica y flexible para caracterizar un grado. El Servicio también dispone de gestión de ficheros (*GradeRecords*), que relaciona la información del grado, el estudiante inscrito en el grado y si está graduado. Este servicio también incluye los métodos iterar a través de los elementos del grado con el apoyo de una aplicación particular.

La forma de evaluar un grado generalmente es distinta a la ofrecida por el *Assment* OSID, sin embargo puede utilizarse este servicio para puntuar al grado creando una relación entre *Assement* OSID y *Grading* OSID.

Hierachy

Hierachy OSID proporciona un medio de crear y recorrer distintos tipos de estructuras jerárquicas como árboles, bosques, grafos dirigidos con varios padres, y gráficos cíclicos dirigidos, etc.

ID

Id OSID da soporte para gestionar identificadores de los distintos objetos gestionados por la plataforma.

Logging

Logging OSID proporciona un medio de la lectura y la escritura de entradas en los registros. El servicio soporta diferentes formatos y prioridades en las entradas del registro. Las aplicaciones pueden utilizar el servicio para registrar la actividad de un sistema de producción, mientras que las otras implementaciones de los servicios pueden utilizar *Logging OSID* para registrar datos detallados durante en el desarrollo de los servicios, identificación de problemas de rendimiento, etc.

Una aplicación concreta puede leer o escribir en más de un registro a la vez. Al igual que varias aplicaciones pueden utilizar en el mismo registro.

Repository

Repository OSID da soporte al almacenamiento y recuperación de contenidos digitales, así como información sobre estos contenidos. Este tipo de contenido incluye: documentos, material del curso, el tema de evaluación, imágenes, video, audio, etc. y residen en repositorios con un nombre y una descripción asociada. Cada repositorio está asociado con un conjunto de contenidos digitales.

La organización de los repositorios depende del componente *RepositoryManager* que realiza un seguimiento de los repositorios y apoya operaciones tales como la búsqueda de contenidos a través de los repositorios. Asociado a cada tipo contenido digital existe una entrada del componente (*RecordStructure*) que define el formato y descripción de la información que del contenido asociado. Los contenidos también pueden ser registros del *Logging OSID*, que son los datos en el formato definido por RecordStructure del activo.

Scheduling

Sheduling OSID proporciona un medio de asociar los agentes con actividades específicas (*ScheduleItems*) que tienen un comienzo y un final determinados por una fecha y una hora. Para cada elemento, el agente o agentes implicados tienen un estado que refleja su nivel de compromiso con la actividad. Es posible tener varias actividades (*ScheduleItems*) de forma concurrente y para encontrar el momento en que los agentes participantes están disponibles.

Sheduling OSID se consulta desde el *CourseManagement*. El OSID proporciona una forma para que las aplicaciones puedan integrar o utilizar un sistema de calendario externa. De esta manera, las aplicaciones puede proporcionar las funcionalidades que poseen otros calendarios externos a la plataforma.

Shared

Shared OSID proporciona la implementación de tipos de clases abstractas y las definiciones de los iteradores de los tipos primitivos.

SQL

SQL OSID proporciona un medio para almacenar, editar, y recuperar datos de las tablas a través de la ejecución de sentencias SQL. Se presupone que los datos se encuentran almacenados en una base de datos relacional.

UserMessaging

Usermessaging OSID proporciona un medio para enviar y recibir mensajes, y suscribirse para recibir mensajes. Cada mensaje tiene un tipo y tema y los mensajes que se reciben pueden ser filtrados por tipo y tema. El tema no tiene por qué ser necesariamente el asunto del mensaje. Los mensajes pueden ser enviados a todos los subscriptores o a un subconjunto específico de ellos.

Usermessaging es un servicio general destinado al usarse con listas de subscripción, direcciones de correo electrónico externa y mensajería instantánea o chat.

Workflow

Workflow OSID proporciona los medios para definir un proceso compuesto de etapas, cada una de ellas posee condiciones de entrada y salida de Estados. Existe un proceso para avanzar en el trabajo desde un estado inicial a un estado final. Este avance se ve afectado por eventos (*WorkEvents*) que se realizan como parte de una etapa, el resultado es un nuevo estado de salida. Los eventos son el resultado de agentes que realizan una función específica en el proceso.

Una implementación del *workflow OSID* soporta un conjunto de condiciones de entrada posibles, los estados de salida y los algoritmos que determinan qué pasos están disponibles para ser aplicadas por los agentes que en cualquier momento específico del proceso.

Parte del servicio tiene por objeto definir procesos y sus pasos. Otras partes del Servicio tienen la intención de capturar los eventos de un usuario del proceso.

El *WorkFlow* OSID proporciona una aplicación o conjunto de aplicaciones con un medio de coordinación y gestión de flujo de trabajo basado en una lógica determinada, entre uno o más actores. La abstracción y la separación del flujo de trabajo de la aplicación permite a la lógica de flujo de trabajo, ser modificado sin cambiar mucho de la aplicación. Herramientas comunes para

la visualización, monitoreo y mantenimiento de flujo de trabajo pueden ser utilizadas en conjunción con la aplicación.

2.3.2 Campus Project

Campus Project es un proyecto de integración de plataformas educativas, su propuesta consiste en la implementación de los servicios descritos en OKI en distintas plataformas educativas. La finalidad es conseguir un nivel de abstracción en las plataformas educativas y, de esta manera, poder crear herramientas que puedan usarse indistintamente en diferentes plataformas.

La arquitectura de Campus Project está compuesta por una capa de denominada OKI BUS, especificada en el proyecto OKI, que sirve de puente entre las plataformas educativas y las herramientas desarrolladas. Las plataformas además necesitan realizar una implementación del OKI BUS para que las herramientas sean capaces de interactuar con la plataforma.

El objetivo, por tanto, de este proyecto está enfocado a la integración entre herramientas educativas con diferentes plataformas e-Learning y no para la integración entre las propias plataformas.

Actualmente el desarrollo se ha realizado centrándose principalmente en la gestión de recursos de la plataforma y en el desarrollo de herramientas para comprobar el correcto funcionamiento de esta gestión. Debido a la importancia del Campus Project, y a la similitud con el proyecto planteado, el capítulo 4 se dedica por completo a comprar dicho proyecto con el trabajo realizado esta memoria.

2.4 Disponibilidad funcionalidad basada en Servicios web en las plataformas e-Learning

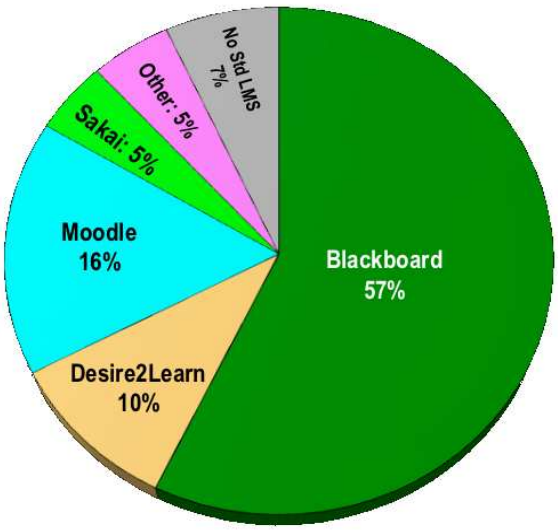
Los servicios web son capaces de abstraer el lenguaje de programación de una plataforma de las herramientas desarrolladas para este. Esta capacidad ofrece un mayor nivel de integración al eliminar la necesidad de desarrollar las herramientas en el mismo lenguaje de programación que la plataforma original.

Sin embargo, los servicios web de los LMS se encuentran en un estado poco maduro de desarrollo. Los servicios ofrecidos, por lo tanto, distan mucho de la funcionalidad ofrecida por las API’s.

Para analizar la disponibilidad de servicios web hay que analizar tres aspectos:

- Arquitectura de los servicios web.
- Recursos accesibles por los servicios web.
- Operaciones que se pueden realizar sobre esos recursos. Estas operaciones pueden tener un soporte total o parcial las operaciones de gestión: alta, baja, modificación y consulta o (*Create, Read, Update and Delete*, CRUD). Además pueden soportar funcionalidad extra similar a la que ofrecida en la aplicación web.

Según el análisis reflejado en la Figura 2.2 tres de las principales plataformas e-Learning del mercado son Blackboard Learn, Moodle y Sakai. En esta sección se va a realizar un análisis de los servicios web ofrecidos por estas plataformas [7] [8]. Para dicho análisis se consideran sus últimas versiones: Blackboard Learn 9.1, Moodle 2.0 y Sakai 2.7.



% by sector	Bb	D2L	Moodle	Sakai
Public Univ.	68.8	5.2	5.2	10.4
Pvt. Univ.	60.0	4.4	8.9	6.7
Public 4-Yr.	64.4	14.9	7.9	2.0
Private 4-Yr.	47.8	---	34.8	5.6
Comm. Colleges	57.0	26.5	5.8	0.8

Figura 2.2 Mercado de las plataformas e-Learning, otoño 2010 [21]

2.4.1 Servicios web Blackboard Learn 9.1

Blackboard es la principal plataforma e-Learning del mercado, Figura 2.2. Esto se traduce en ser la plataforma con mayor experiencia en el sector. La funcionalidad ofrecida es amplia, sencilla y acorde con las necesidades de las instituciones. Una de sus principales desventajas frente a otras alternativas es la alta inversión que supone apostar por esta plataforma

y su uso, por tanto, tiende a ser por instituciones de mayor tamaño y presupuesto como universidades de tamaño medio-grande.

Los servicios web de Blackboard destacan por ser los que ofrecen una funcionalidad más amplia y una arquitectura más clara, separándolos por funcionalidad y contenido administrado. La funcionalidad ofrecida es, sin embargo, muy inferior a la ofrecida por la API de la plataforma.

Arquitectura de los servicios web

Blackboard da prioridad a la funcionalidad de los servicios web frente a su usabilidad, esto se ve reflejado en la arquitectura de los mismos. Esta arquitectura está basada en un conjunto de servicios web con funcionalidades previamente definidas, y usa tecnología basada en el protocolo SOAP para la comunicación, dejando de lado otros protocolos como XML-RPC, AMF o REST. Esta característica resta usabilidad y obliga a adaptarse a dicho protocolo. Sin embargo SOAP satisfaces los requisitos de funcionalidad de los servicios web y no les resta funcionalidad.

Por otro lado, Blackboard es una aplicación comercial de código cerrado. Por lo tanto se desconoce la arquitectura interna de los propios servicios web, conociendo únicamente las descripciones de los servicios web que están disponibles en la plataforma.

Los servicios web disponibles en Blacboard Learn 9.1 son:

- *Announcement:* Permite gestionar los anuncios de las asignaturas.
- *Calendar:* Contiene las operaciones de administración de los eventos del calendario.
- *Content:* Ofrece funcionalidad de manipulación de los contenidos y recursos de los cursos.
- *Context:* Gestiona las sesiones.
- *Course:* Dispone de las operaciones de gestión para el acceso y creación de cursos.
- *CourseMembership:* Permite Gestionar la matriculación de los participantes en los diferentes cursos.
- *Gradebook:* Permite administrar las calificaciones de los cursos.
- *NotificationDistributorOperations:* Permite manejar las notificaciones.

- *User:* Contiene las operaciones de gestión de usuarios.
  Util: Este servicio web posee operaciones secundarias de acceso y de configuración global.

Tabla 2.1 Localización de los servicios web Blackboard Learn 9.1

Servicio Web	Dirección
Announcement	http://your.bb.server/webapps/ws/services/Announcement.WS
Calendar	http://your.bb.server/webapps/ws/services/Calendar.WS
Content	http://your.bb.server/webapps/ws/services/Content.WS
Context	http://your.bb.server/webapps/ws/services/Context.WS
Course	http://your.bb.server/webapps/ws/services/Course.WS
CourseMemebership	http://your.bb.server/webapps/ws/services/CourseMembership.WS
Gradebook	http://your.bb.server/webapps/ws/services/Gradebook.WS
User	http://your.bb.server/webapps/ws/services/User.WS
Util	http://your.bb.server/webapps/ws/services/Util.WS

Los servicios web deben ser activados manualmente por el administrador del sistema y la localización de cada servicio viene indicada en la Tabla 2.1.

2.4.2 Servicios web Moodle 2.0

Moodle es actualmente la principal plataforma libre existente y la segunda más extendida entre las plataformas de gestión de contenidos educativos como muestra la Figura 2.2. La filosofía de esta plataforma se basa en los conceptos de funcionalidad y usabilidad. Los bajos requisitos para la implantación del software hace que sea muy accesibles para cualquier centro educativo que quiera utilizar una plataforma e-Learning y la ha convertido en una de las plataformas más populares y usadas.

Los servicios web en esta plataforma han sido incluidos en la última versión de la plataforma hasta la fecha (Moodle 2.0). Posee una arquitectura bien definida, sin embargo, la funcionalidad que ofrece es aún bastante escasa en comparación con otras plataformas e-Learning del sector.

Arquitectura de los servicios web

Bajo la filosofía de usabilidad, Moodle ofrece una arquitectura de servicios web distinta a al resto de plataformas que hay en el mercado. La arquitectura dista de la filosofía clásica, donde

existe un conjunto de servicios web previamente definidos. Moodle, como muestra la Figura 2.3, implementa una serie de funcionalidades en librerías, generalmente existe una librería por cada recurso utilizado. Una vez desplegada la plataforma sobre el servidor, el administrador de dicha plataforma puede agregar los servicios web que desee pudiendo decidir a quien se ofrece dicho servicio y que las funcionalidades ofrecidas del conjunto de funcionalidades previamente implementadas. Esta arquitectura es muy versátil y se adapta muy bien a las necesidades de los campus virtuales que usan Moodle.

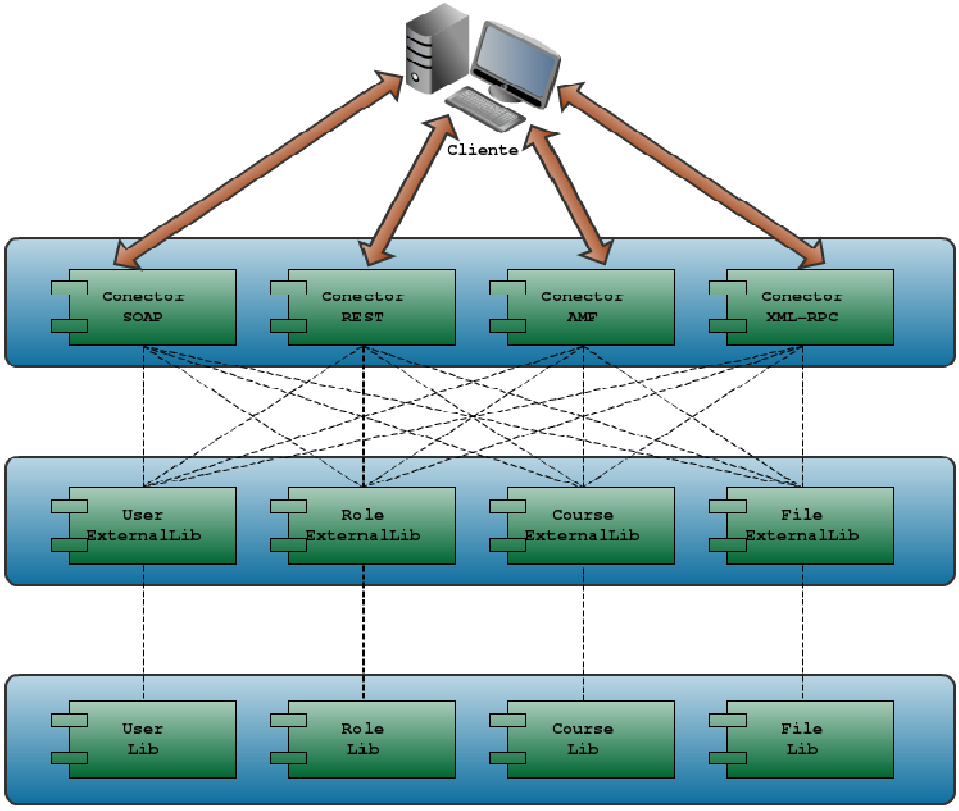


Figura 2.3 Arquitectura de los servicios web de Moodle

Continuando con la filosofía de usabilidad, Moodle puede publicar los servicios web utilizando cuatro de las tecnologías más importantes: SOAP, AMF, REST y XML-RPC. Sin embargo solo es posible utilizar una de ellas a la a vez.

La arquitectura está organizada en los siguientes componentes:

- *Conector:* Esta clase se encarga de publicar los servicios web. Existe un conector por cada una de las tecnologías soportadas por Moodle y utiliza las librerías "*ExternalLib*" para publicar los servicios.

- *ExternalLib*: Contiene el conjunto de operaciones que los servicios web pueden incluir. Estas librerías están divididas por recurso gestionado. Los recursos que actualmente tienen librería externa son usuarios, roles, cursos y gestión básica de ficheros.
- *InternalLib*: La implementación de las operaciones se realiza mediante esta librería que representa el API de la plataforma.

La Tabla 2.2 indica que operaciones implementadas hasta la fecha y que, por tanto, se pueden incluir en los servicios web de Moodle 2.0.

Tabla 2.2 Operaciones soportadas por los servicios web de Moodle 2.0

Modulo	**Operación**	**Descripción**
User	moodle_user_create_users	Crea un usuario
	moodle_user_delete_users	Elimina un usuario
	moodle_user_get_users_by_id	Obtiene la información de un usuario
	moodle_user_update_users	Modifica un usuario
Role	moodle_role_assign	Asigna un rol a un usuario
	moodle_role_unassign	Elimina un rol a un usuario
Group	moodle_group_add_groupmembers	Añade un usuario a un grupo
	moodle_group_create_groups	Crea un grupo
	moodle_group_delete_groupmembers	Elimina un usuario de un grupo
	moodle_group_delete_groups	Elimina un grupo
	moodle_group_get_course_groups	Obtiene los grupos de un curso
	moodle_group_get_groupmembers	Obtiene los usuarios de un grupo
	moodle_group_get_groups	Obtiene los grupos de la plataforma
Course	moodle_course_create_courses	Crea un curso
	moodle_course_get_courses	Obtiene todos los cursos de la plataforma
	moodle_enrol_get_enrolled_users	Matricula a un usuario en un curso
Resource	moodle_file_get_files	Obtiene un recurso

2.4.3 Servicios web Sakai 2.7

Sakai es una plataforma que surge en 2004 bajo la iniciativa gran cantidad de centros educativos del mundo entre los que destacan, Universidad de Míchigan, la Universidad de Indiana, Instituto Tecnológico de Massachussets y la Universidad de Standford, y como respuesta a la necesidad de más alternativas libres del mercado de los plataformas educativas.

Otros centros que se encuentran implicados en el proyecto son la Universidad de Cape Town (Sudáfrica), la Universidad de Cambridge, la Universidad de Toronto, la Universidad Politécnica de Valencia y la Universidad del Valle de Guatemala.

Sakai está implementada en Java y se distribuye bajo licencia educativa libre y de código abierto ECL. Es una plataforma educativa joven en el mercado pero se encuentra en un estado de desarrollo muy maduro y es capaz de competir con otras plataformas e-Learning como Moodle o Blackboard.

Los servicios web ofrecidos desarrollados en Sakai se encuentran en un estado de desarrollo muy avanzado y, aunque lejos de ofrecer la misma funcionalidad que la aplicación, se pueden tiene soporte para operaciones avanzadas sobre la mayoría de los recursos de la plataforma.

Arquitectura

Sakai ofrece una arquitectura clásica de servicios web basada en un conjunto de servicios web estáticos. La plataforma soporta las tecnologías SOAP y REST para los servicios web y, aunque la funcionalidad sea prácticamente equivalente, la arquitectura difiere un poco según el protocolo usado. A continuación se describen dichas arquitecturas en base a los protocolos.

Arquitectura SOAP:

Sakai implementa los servicios web SOAP mediante el framework *Apache Axis* [22], lo que implica que usa la API JAX-RPC [17]. Como puede verse en la Figura 2.4, la arquitectura de Sakai consta de un conjunto de servicios web estáticos, las operaciones que se agrupan en cada servicio web están divididas según su funcionalidad, administración de la plataforma; gestión de sesiones, usuarios y recursos, etc.

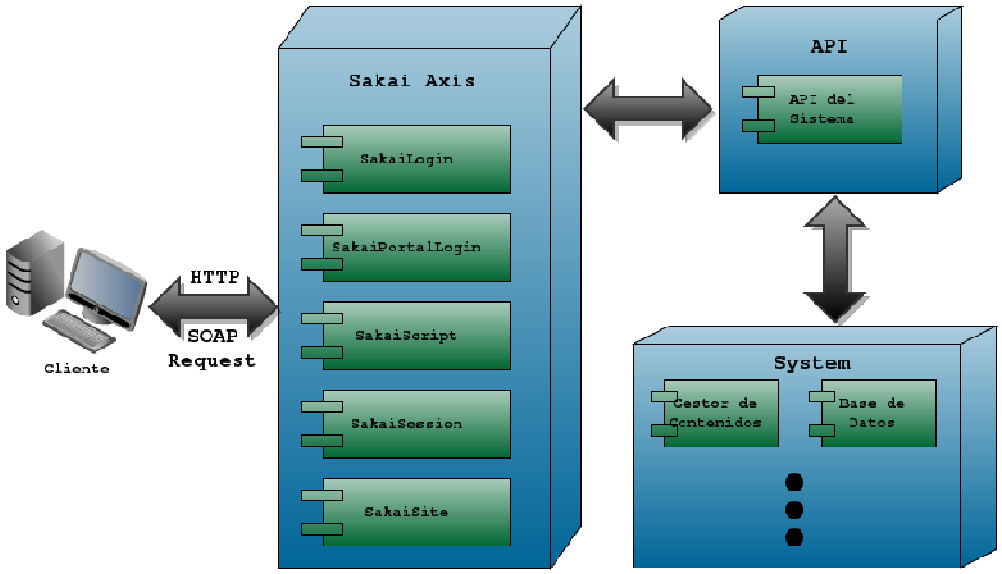


Figura 2.4 Arquitectura SOAP-Axis de Sakai

Tabla 2.3 Localización de los servicios web de Sakai 2.7

Servicio Web	**Dirección**
SakaiLogin	http://your.sakai.server/sakai-axis/SakaiLogin.jws
SakaiPortalLogin	http://your.sakai.server/sakai-axis/SakaiPortalLogin.jws
SakaiScript	http://your.sakai.server/sakai-axis/SakaiScript.jws
SakaiSession	http://your.sakai.server/sakai-axis/SakaiSession.jws
SakaiSite	http://your.sakai.server/sakai-axis/services/SakaiSite.jws

Los servicios web disponibles en Sakai con el protocolo SOAP son:

- *SakaiLogin*: Este servicio es el responsable de la gestión del inicio de sesión. Debe ser invocado para usar el resto de servicios.
- *SakaiPortalLogin*: Servicio necesario para la integración con uPortal. Framework usado por Sakai.
- *SakaiScript*: Servicio web que contiene las operaciones de gestión de recursos de la plataforma como usuarios, cursos, roles, etc. Este servicio web es el pilar de la gestión de la plataforma. Figura 2.5.

- *SakaiSession*: Servicio que informa de la sesión activa.
- *SakaiSigning*: Este servicio permite a aplicaciones externas para verificar usuarios.
- *SakaiSite*: Servicio para administrar el sitio.

Estos servicios se encuentran organizados según muestra la Tabla 2.3.

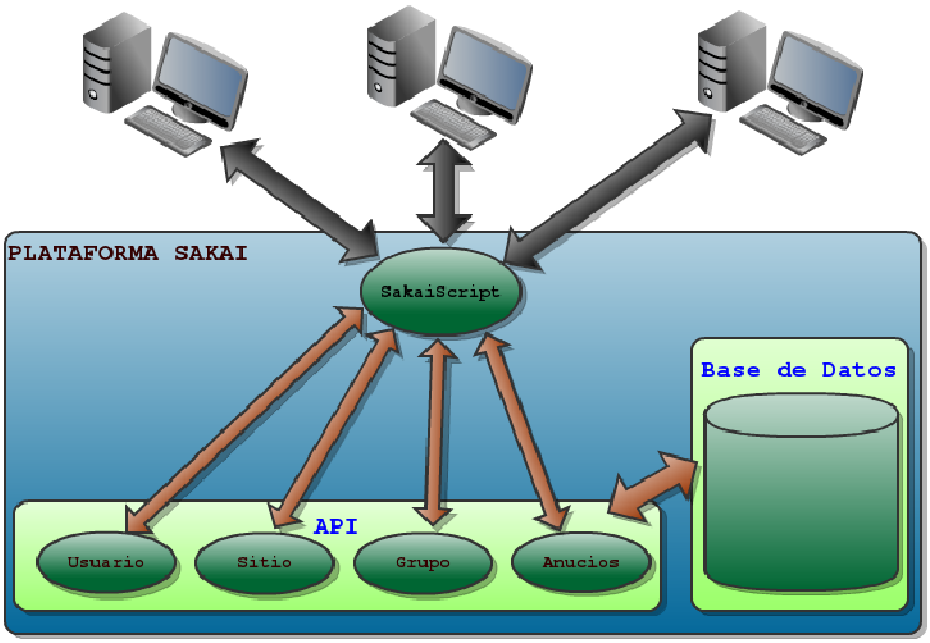


Figura 2.5 Servicio Web de Manipulación de datos

Arquitectura RESTful:

Los servicios web basados en RESTful se introdujeron después de los basados en SOAP. A diferencia de SOAP, en lugar de enviar mensajes XML a través de HTTP a una URL que apunta a un servicio, RESTful envía a una dirección URL que incluye información sobre la entidad sobre la que se van a realizar las operaciones.

Un ejemplo de esta dirección es:

http://your.sakai.server/direct/addressbook_item/15.

La arquitectura de estos servicios posee un componente por cada elemento del sistema así como un componente que ayuda a administrar los componentes del sistema este componente recibe el nombre de *entity broker*. Sobre cada uno de estos componentes, dependiendo del estado

de desarrollo de los servicios, se podrá realizar los métodos HTTP POST, GET, PUT y DELETE.

2.4.4 Estado de madurez de los servicios web en las plataformas.

Esta sección analiza y compara los servicios web en las tres plataformas anteriores.

Este análisis considera las principales funcionalidades usadas para gestionar las plataformas como gestión de sesión, usuarios o cursos, bajo la perspectiva de las funcionalidades de gestión, estas son, alta, baja, modificación y consulta (Create, Read, Update, Delete, CRUD).

La disponibilidad las operaciones CRUD para cada una de las funcionalidades se clasifica en:

- Soporte Completo: Las operaciones CRUD se encuentran implementadas. Además se han implementado adicionales que también se encuentran implementadas en la plataforma.
- Soportado: Se han implementado las operaciones CRUD.
- Soporte parcial: Solo parte de las operaciones CRUD han sido implementadas.
- Sin Soporte: No se han implementado servicios web para esa funcionalidad.

Además, se han analizado los protocolos soportados por las distintas plataformas, este análisis se muestra en la Tabla 2.4. La Tabla 2.5 analiza la disponibilidad de operaciones en función de la plataforma.

Tabla 2.4 Protocolos soportados por las plataformas e-Learning

Servicio Web	**Plataforma**		
	BlackBoard	**Moodle**	**Sakai**
SOAP	Soportado	Soportado	Soportado
RESTful	No soportado	Soportado	Soportado
XML-RPC	No soportado	Soportado	No soportado
AMF	No soportado	Soportado	No soportado

Como resultado se ha observado como Sakai y Blackboard, en las versiones analizadas, se encuentran en un estado de desarrollo mejor que Moodle. Cabe destacar que la plataforma Sakai dispone de un mayor soporte para servicios basados en RESTful.

También cabe destacar que, sin embargo, la mayor parte de los servicios implementados se centran en la gestión de usuarios, cursos y sesión. Existe poco o ningún soporte para servicios que engloban otras áreas, tales como calendario, anuncios, etc.

Por otro lado, dentro de la cantidad de protocolos soportados hay que destacar el esfuerzo de la plataforma Moodle que dispone de soporte para una mayor cantidad de protocolos.

Tabla 2.5 Disponibilidad de servicios web en plataformas e-Learning

Servicio Web	**Plataforma**			
	BlackBoard	**Moodle**	**Sakai (REST)**	**Sakai (SOAP)**
Sesión	Soporte Completo	Soporte Completo	Soporte Completo	Soporte Completo
Usuario	Soportado	Soportado	Soportado	Soportado
Grupos	Soportado	Soportado	Soportado	Soportado
Rol	Soportado	Soporte Parcial	Soportado	Soportado
Matriculación	Soportado	Soportado	Soportado	Soportado
Curso	Soportado	Soporte Parcial	Soportado	Soportado
Recurso	Soportado	Soporte Parcial	No soportado	No soportado
Anuncio	Soporte Completo	No soportado	No soportado	Soporte Completo
Foro	No Soportado	No soportado	No soportado	No soportado
Calendario	Soportado	No soportado	No soportado	Soporte Parcial
Notificación	No soportado	No soportado	Soporte Parcial	No soportado
Correo Int.	No soportado	No soportado	No soportado	No soportado
Correo Ext.	No soportado	No soportado	No soportado	No soportado
Blog	No soportado	No soportado	No soportado	No soportado
Calificación	No soportado	No soportado	Soporte Parcial	No soportado
Mensajería	No soportado	No soportado	Soporte Parcial	No soportado
Votaciones	No soportado	No soportado	Soportado	No soportado
Tareas	No soportado	No soportado	Soporte Parcial	No soportado

3 Arquitectura de la solución

En este capítulo se plantea una solución para integrar tres de las principales plataformas educativas existentes: Blackboard, Moodle y Sakai. De esta manera se pretende conseguir un marco común entre ellas que facilite el uso de las distintas plataformas dentro de un mismo campus virtual, y facilite la migración de contenidos entre LMS.

Para plantear la solución primero se estudiarán las distintas alternativas arquitectónicas, tanto existentes, como modelos arquitectónicos nuevos, analizando las características, ventajas e inconvenientes de cada una.

3.1 Alternativas planteadas

Las distintas arquitecturas que puede presentar un campus virtual dependen del nivel de integración entre las plataformas educativas y funcionalidad que se quiere usar de estas. Algunas arquitecturas de integración están siendo utilizadas por los centros educativos actualmente, otras son todavía experimentales.

3.1.1 Campus Virtual con un LMS explícito

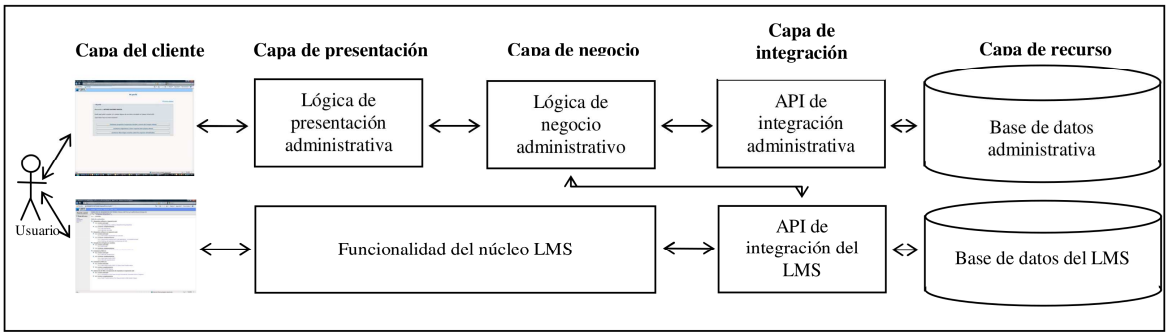


Figura 3.1 Arquitectura antigua del campus virtual de la UCM

Muchos de los centros que se deciden a utilizar plataformas educativas no disponen de los recursos o la experiencia necesaria para el uso de varias plataformas educativas, el primer reto al que se enfrentan estos centros es elegir que plataforma utilizar. Esta decisión viene determinada por la funcionalidad que ofrece la plataforma y costes de implantación y mantenimiento.

Estos centros utilizan arquitecturas centradas únicamente en un LMS y es esta plataforma la que determina la arquitectura del campus virtual y no existe, por lo tanto, ningún tipo de integración con otros LMSs.

Son estos centros los que utilizan esta arquitectura, que se basa en una sola plataforma, ligándose a ella y adaptándose a su uso. La arquitectura de esta solución es sencilla y se basa en la proporcionada por la propia arquitectura de la plataforma.

Estas arquitecturas pueden contar con una capa administrativa. Esta capa es la encargada de realizar las gestiones de usuarios. Las sesiones se realizan frente a ella y es esta capa la que inicia sesión con la LMS. Esta capa cuenta con una BBDD administrativa que almacena información de sesión de los usuarios.

Las ventajas de esta solución de campus virtual son:

- La implantación de la plataforma es la más sencilla de todas las arquitecturas planteadas en este punto al no existir ningún tipo de integración.
- La gestión de la plataforma se centra en la implantación. No es necesario conocer en profundidad la arquitectura del LMS y las labores pueden centrarse únicamente en labores de mantenimiento.
- Existen empresas que platean soluciones integrales de plataformas, esto permite a los centros educativos, externalizar la plataforma.
- Se convierte en una solución óptima para centros educativos con niveles presupuestales ajustados.

Las principales desventajas son:

- La elección de una única plataforma educativa es complicada, hay que sopesar las distintas alternativas, son sus características. Esta elección es más difícil si es la primera vez que se va a implantar una plataforma.
- Cambiar de solución educativa es muy complicado y conllevar complicadas operaciones de migración de datos y cursos. Estas operaciones, además no pueden hacerse sobre asignaturas en curso.
- Vincula el campus virtual a un proveedor fijo, lo que puede crear problemas con gestión de licencias y/o mantenimiento.

En general esta alternativa es óptima para centros educativos pequeños y con poca carga sobre las plataformas educativas, no es la mejor opción si se piensan utilizar características avanzadas de las plataformas.

En la primera etapa del e-Learning de la Universidad Complutense de Madrid, Figura 3.1, usó una arquitectura similar agregando por separando la gestión administrativa de la plataforma. El LMS que utilizaba era WebCT 4.1, un LMS superado hoy en día, y los procesos de adaptación a nuevas soluciones han sido costosos tanto en esfuerzo, como en tiempo, ya que se han tenido que invertir varios cursos para su sustitución.

3.1.2 Campus Virtual con varios LMSs explícitos y una página de integración

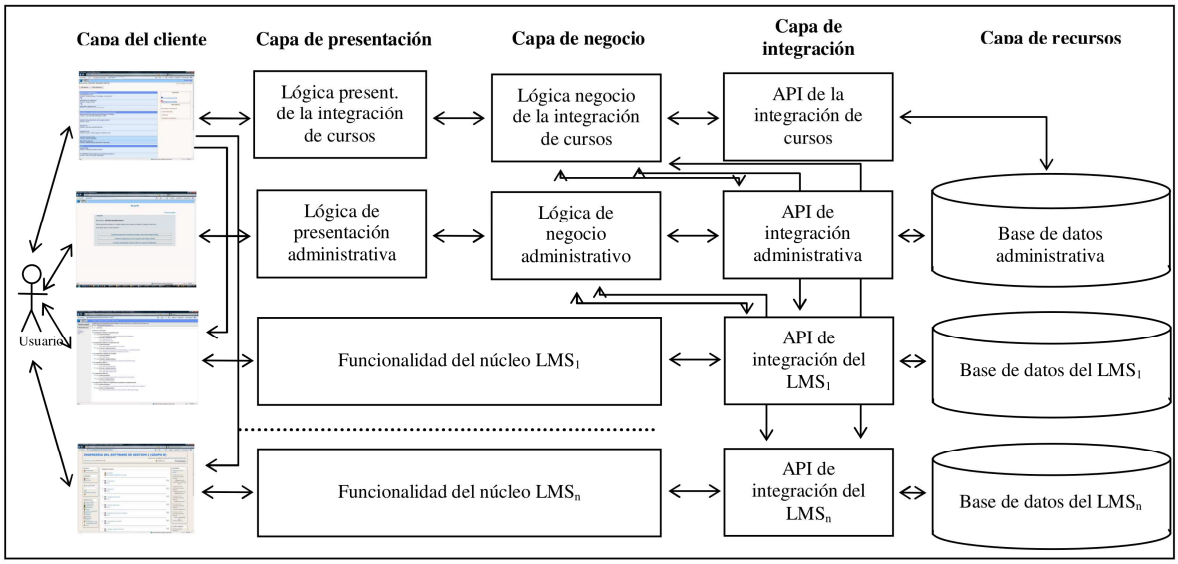


Figura 3.2 Arquitectura actual de la UCM

El principal problema de usar un único LMS es el nivel de dependencia de la plataforma y la dificultad intrínseca de cambiar a nuevas plataformas. Este problema hace que se planteen nuevas arquitecturas que permitan utilizar diferentes plataformas explícitamente.

Con esta idea, aparece una arquitectura que utiliza niveles básicos de integración. Esta arquitectura se caracteriza por el uso de varias plataformas que gestionan cursos, cada curso está se encuentra en una plataforma, y de una página que gestiona el acceso a cada curso, Figura 3.2. En este tipo de arquitectura se puede diferenciar 3 niveles. El primer nivel, la gestión administrativa, ya se encuentra planteada en la arquitectura descrita en el apartado 3.1.1, y ahora

es un elemento indispensable. Esta capa tiene un papel más importante en esta arquitectura ya que es la encargada de toda la gestión e interacción de los usuarios con las plataformas.

Aparece un segundo nivel completamente nuevo la gestión de cursos: la integración de plataformas. Esta integración aparece por una doble necesidad, la necesidad de conocer las asignaturas de cada alumno y conocer en que plataforma está alojada cada curso. Gracias a este nivel la primera interacción se realiza antes del acceso a las plataformas y será cuando el alumno quiera acceder a una asignatura, cuando entre en funcionamiento el campus virtual.

El último nivel se encuentran las plataformas e-Learning, a este nivel se accede cuando el usuario quiere acceder a un curso en concreto, accediendo a la plataforma que aloja dicha asignatura y al curso en concreto. Una vez accedida a la asignatura, toda la interacción con el curso queda delegada al LMS y por lo tanto, las capas de integración de usuarios y cursos dejan de actuar.

Esta arquitectura, por lo tanto, permite utilizar varias plataformas educativas al mismo tiempo con un doble objetivo: no depender de una única plataforma educativa y utilizar características avanzadas de cada plataforma. Estas características hacen que esta arquitectura no está orientada a centros educativos que tengan bajos presupuestos para el área del campus virtual o que no usen características avanzadas de las plataformas.

En esta arquitectura, el componente de la base de datos juega un papel mucho más importante, ya que es la que mantiene la información actualizada de todos los usuarios, cursos matriculados y que plataforma aloja cada curso, y por tanto interactúa con el LMS activo en ese momento. La arquitectura planteada en el apartado anterior, solo almacenaba información de usuarios.

Esta arquitectura destaca por:

- Presentar una forma sencilla y rápida de implementar la integración entre LMSs, ya que la lógica de los cursos sigue dependiendo completamente de cada plataforma de e-learning.
- Las dependencias de los centros educativos con las plataformas utilizadas es mucho menor, restringiéndose únicamente a cada curso y periodo lectivo del mismo, puesto que migrar asignaturas entre periodos lectivos es mucho más sencillo, al menos desde el punto de vista de instalación y puesta en marcha de una plataforma de e-learning. Evidentemente, para la migración de contenidos

concretos entre plataformas distintas, sería necesario el desarrollo, o especialización, de software específico para la migración de contenidos entre plataformas.

- Se pueden usar las herramientas de los distintos LMSs, aprovechando las características avanzadas de las que dispone cada plataforma.

Sin embargo la integración es únicamente entre las capas de gestión de usuarios y cursos, la integración a nivel plataforma no existe y, pese a esto, la complejidad de la comunicación entre las capas de integración y los campus virtuales es bastante elevada.

Además cada curso utiliza un único LMS, por lo que para utilizar características avanzadas de las plataformas, requiere duplicar los cursos, con muchos problemas de sincronización de información, ya que esto no está contemplado.

La Universidad Complutense de Madrid implementa actualmente esta arquitectura como muestra la Figura 3.2. Sin embargo ha presentado problemas en la capa de gestión de cursos y la comunicación con las diferentes plataformas. Los principales problemas han sido causados por integrar plataformas que ya daban problemas antes de su integración (WebCT 4.1). Por lo tanto, la aplicación de integración, aunque desarrollada y plenamente funcional, en la actualidad, no se utiliza en el Campus Virtual de la UCM.

Al delegar la funcionalidad del campus virtual a las propias plataformas y al existir únicamente integración a nivel de administración, la complejidad de esta no es muy alta y lo que permite que puedan convivir varias plataformas. Sin embargo, para el alumno puede resultar problemático el uso de varias plataformas e-Learning, ya que al ser funcionalmente independientes, el alumno está obligado a utilizar cada una de manera independiente. Otro inconveniente de esta aproximación es no poder usar las características de todas las plataformas en un mismo curso.

Uno de los principales problemas de integración de distintas plataformas es su heterogeneidad. Cada plataforma dispone distintas funcionalidades, e incluso, para las mismas operaciones y contenidos, el modo de representarlos depende de la plataforma utilizada.

Por tanto, el proceso de integración lleva consigo la unificación tanto de la funcionalidad, como del contenido de la plataforma, y es el principal punto de conflicto de las arquitecturas de integración. Esta integración puede plantearse desde, al menos, dos puntos de vista diferentes, dependiendo de quien realice la unificación del contenido de las plataformas.

3.1.3 Campus Virtual de marca blanca con un único LMS funcionando

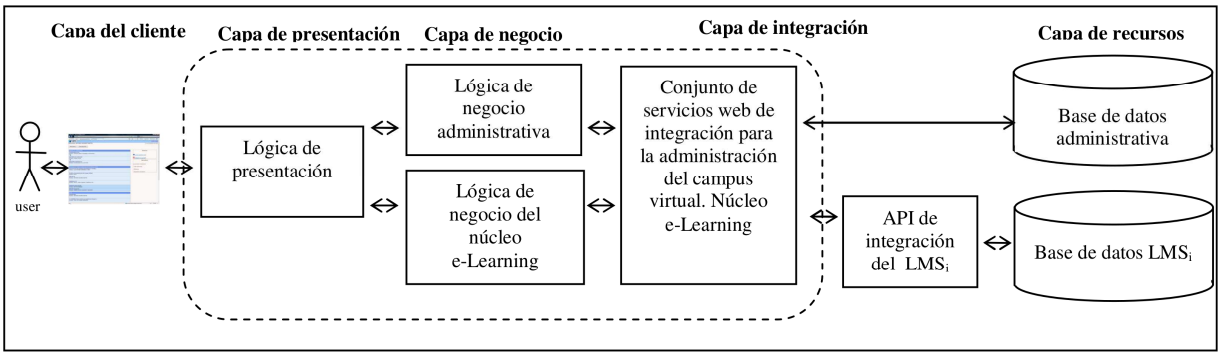


Figura 3.3 Arquitectura de integración basada en servicios web, un LMS funcionando.

En el apartado anterior, la integración se ha planteado mediante una página común, y está página es la que se ha adaptado a cada una de las plataformas presentando el contenido de las plataformas. Sin embargo, la integración puede enfocarse desde otro punto de vista, extendiendo las plataformas con un conjunto de operaciones similares expuestas para que puedan usarse desde la capa de integración de una arquitectura multicapa. Este conjunto de operaciones definirá el grado de integración entre las plataformas. Dicho conjunto puede incluir operaciones de gestión de cursos y usuarios, explicadas en el apartado anterior, e incluir operaciones sobre los cursos, pudiendo acceder a los contenidos de los cursos de las asignaturas sin necesidad de acceder a la aplicación web.

Bajo este conjunto de operaciones puede crearse una arquitectura que, utilizando estas operaciones, pueda comunicarse con cada una de las plataformas sin conocer las implementaciones concretas de las operaciones.

La arquitectura planteada usa este concepto de operaciones comunes mediante su implementación con servicios web. Estos servicios web son usados por un cliente, el cual denominaremos *campus virtual de marca blanca* [3], que es capaz de interactuar con las distintas plataformas sin conocer exactamente que plataforma está utilizando. El *campus virtual de marca blanca* usa únicamente un solo LMS simultáneamente.

Los servicios web, planteados en el apartado 3.2.1, definen una funcionalidad y nivel de integración entre plataformas. Los niveles de integración pueden incluir la adaptación de la gestión planteada en el apartado anterior, o disponer de todas las funcionalidades de la plataforma.

En esta arquitectura planteada, la gestión de cursos sigue estando en una capa distinta, ya que al poder usar únicamente una plataforma simultáneamente, es necesario conocer en que plataforma está alojada el curso al que se quiere acceder.

El campus virtual que adquiera esta plataforma gana un grado de independencia mayor, ya que para integrar una plataforma solo se necesita desarrollar estos servicios web. Además, la integración de plataformas nuevas causa un menor impacto sobre los usuarios ya que no cambia la presentación de los servicios.

Esta arquitectura mantiene la funcionalidad de las plataformas lo que permite que puedan acceder a ellas mediante la página web.

Sin embargo cada curso sigue teniendo una plataforma asociada y es incapaz de integrar las funcionalidades de varias plataformas.

Generalizar las operaciones, tampoco es algo trivial, debido a que las características que diferencian las plataformas desaparezcan en las implementaciones desarrolladas, convirtiendo a las plataformas en contenedores con funcionalidades similar. Esto complica el desarrollo de los servicios web, que deben generalizar las operaciones sin que eso conlleve perder la esencia de cada plataforma.

Este tipo de arquitectura ha sido el que se ha desarrollado en el proyecto y presenta el primer paso para integrar las plataformas completamente, incluso en el mismo curso.

3.1.4 Campus Virtual de marca blanca basado en distintos servicios proporcionados por diferentes LMSs funcionando de manera simultánea

El siguiente nivel de integración consiste en ampliar la arquitectura del campus virtual descrita en el apartado anterior. La finalidad es poder usar varias plataformas simultáneamente en un curso.

En esta arquitectura, los cursos están gestionados completamente por el cliente de campus virtual. Este campus virtual de marca blanca, mediante un conjunto de servicios descriptos en el apartado anterior, utiliza las herramientas de las distintas plataformas para cada curso. El objetivo es utilizar las características avanzadas de cada LMS, de esta manera se elimina la restricción que existía en la arquitectura descrita en el apartado anterior.

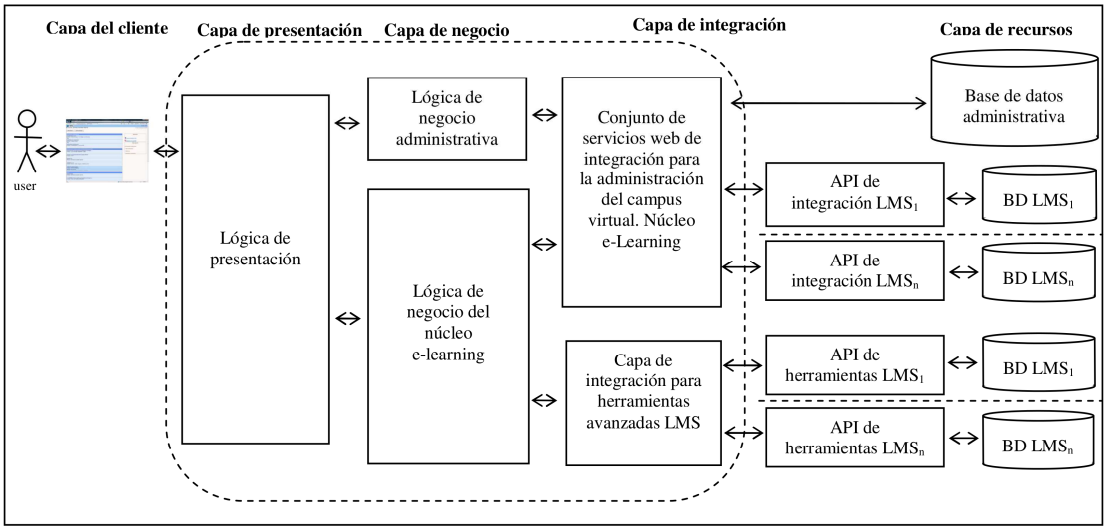


Figura 3.4 Arquitectura integración basada en servicios web, varios LMS funcionando

Una de las características que diferencia esta arquitectura de la anterior es que, al estar el curso dividido entre las plataformas, si se acede a los cursos mediante la aplicación web la información que contiene puede no ser la completa. Esta característica obliga a adaptar cada una de las herramientas que se quieran usar en el campus virtual de marca blanca, ya que estas herramientas existen solo en una de las plataformas o son demasiado singulares para adaptarse entre sí.

Esta arquitectura, por tanto, es la más avanzada que las descritas y la integración se realiza en dos niveles. El primer nivel incluye herramientas incluyen las operaciones de gestión de los recursos por parte del campus virtual de marca blanca, estas operaciones son las mismas que las planteadas en el apartado 3.1.3; el segundo nivel de integración engloba aquellas herramientas de las plataformas que no realizan operaciones de gestión propiamente dicho, para este nivel es necesario crear una interfaz común de integración para que sean integradas. Comparando esta arquitectura con la planteada en el apartado 3.1.3, ambas plataformas comparten la parte de integración entre plataformas, el conjunto de servicios web de integración. Además añade servicios web para integrar las herramientas de las plataformas.

Por lo tanto este campus virtual permite el uso de cualquier plataforma que disponga de los servicios necesarios para administrar la plataforma, servicios descritos en el apartado anterior.

Esta arquitectura elimina además la mayoría de las restricciones de integración entre plataformas, la única restricción aplicable es la impuesta por el uso de herramientas específicas de un campus virtual.

El concepto de migración entre plataformas LMS es distinto, esto se debe a que las plataformas son simplemente componentes reemplazables de la arquitectura siempre que cumplan los requisitos de los servicios comunes ofrecidos y se crean mecanismos potentes de migración entre ellas que permiten tener un mismo curso replicado y utilizado en varias plataformas de manera simultánea.

Con esta arquitectura cada curso puede disponer de las herramientas disponibles independientemente de la plataforma que la ofrezca.

Esta integración es trasparente para el usuario que utilice el campus virtual ya que es este el que se encarga de interactuar entre las distintas plataformas, las herramientas disponibles y los usuarios finales.

Sin embargo esta arquitectura obliga a implementar complejas capas de integración para que el campus virtual sea capaz de interactuar con las distintas plataformas. Esta capa de integración tiene que comportarse de una manera similar en cada una de las plataformas.

Otra de las dificultades que conlleva la implantación de esta arquitectura es el coste de la adaptación de las herramientas concretas de cada plataforma.

Esta plataforma, por tanto, está únicamente indicada para centros educativos que usan el campus virtual, no solo como una herramienta de apoyo a los cursos, sino que la educación e-Learning tiene un papel crucial en estos. Los centros educativos que, sin necesitar estas características, implanten esta arquitectura en sus campus virtuales, incrementaran su complejidad aprovechando las ventajas de esta plataforma.

3.2 Diseño

Las arquitecturas planteadas en el apartado anterior presentan distintas alternativas para integrar plataformas educativas. Las alternativas planteadas en los apartados 3.1.3 y 3.1.4 son las alternativas que presenta y nivel de integración mayor. La alternativa planteada en el apartado 3.1.3, Campus Virtual de marca blanca con un único LMS funcionando, representa la arquitectura previa para la alternativa planteada en el apartado 3.1.4, Campus Virtual de marca blanca basado en distintos servicios proporcionados por diferentes LMSs funcionando de manera

simultánea. Además, representa una alternativa con ventajas en integración y migración de datos sin perder la funcionalidad de las plataformas.

Es por esto por lo que en este trabajo se ha desarrollado la arquitectura 3.1.3 para la integración de plataformas. Esta arquitectura de la solución para la integración de plataformas e-Learning, como se indica en el apartado 3.1.3, está basada en un campus virtual de marca blanca y en un conjunto de servicios web que denominamos *servicios web canónicos* implementados en las diferentes plataformas. Estas plataformas alojan los diferentes cursos, pero estos cursos no se encuentran en más de una plataforma a la vez.

Esta arquitectura, por tanto, define en primer lugar el conjunto de servicios web canónicos, su funcionalidad y operaciones. En segundo lugar, se definirá la arquitectura del cliente del campus virtual de marca blanca que usa dicho servicios. Por último la arquitectura concreta de la implementación de los servicios web en cada una de las plataformas que ha sido posible realizar

3.2.1 Servicios web canónicos

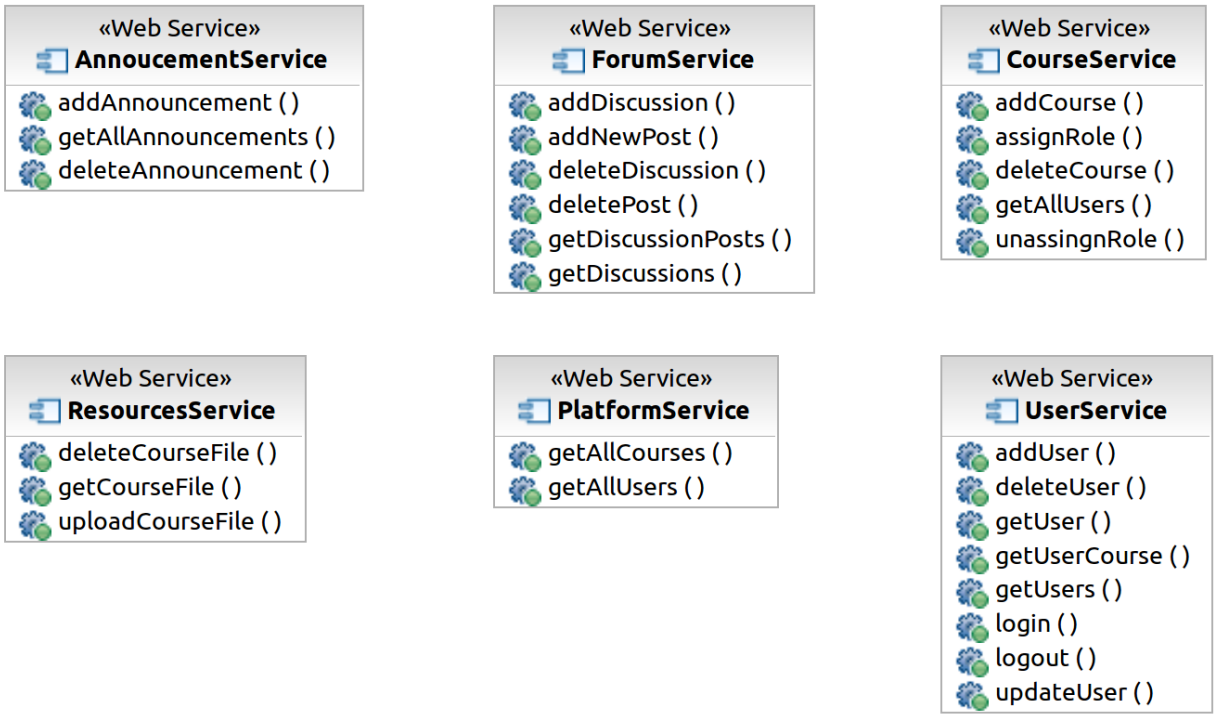


Figura 3.5 Servicios web canónicos y sus operaciones

Los servicios web nos permiten exponer la funcionalidad de una plataforma independientemente de la implementación que la plataforma haya en las capas inferiores. Esto nos permite abstraer los detalles de implementación de la plataforma y su lenguaje de programación original. También permite un uso distribuido de las plataformas.

En la arquitectura de la solución, los servicios web son los encargados de la integración entre las distintas plataformas de e-Learning, y el campus virtual de marca blanca, mediante la implementación de estos en todas las plataformas. Todas las implementaciones implementan las mismas interfaces y es por esto por lo que el campus virtual de marca blanca puede usarlos para su integración.

El conjunto de estos servicios dispone de funcionalidad para administrar la plataforma, gestionar cursos y contenidos de los cursos. El conjunto de servicios web cubre operaciones sobre:

- Identificación: Identificación de los usuarios que quieran usar los servicios
- Plataforma: Consulta de información general de la plataforma
- Usuarios: Gestión y consulta de usuarios.
- Curso: Acceso y gestión de los cursos de la plataforma y Gestión de los participantes.
- Anuncios: Gestión del sistema de anuncios de un curso.
- Foro: Gestión del sistema de foros.
- Recursos: Acceso a los recursos digitales de los cursos

Definición de los servicios

Para agrupar las operaciones en servicios web, se han tenido en cuenta el recurso al que hacen referencia, de tal manera que todas las operaciones referentes a los foros, por ejemplo, se han agrupado en un mismo servicio web. Los servicios web resultantes son:

- Servicio web usuario (UserService).
- Servicio web cursos (CourseService).
- Servicio web foro (ForumService).
- Servicio web plataforma (PlatformService).
- Servicio web recursos (ResourceService).

- Servicio web anuncios (AnnouncementeService).

Todos los servicios web excepto login, en los servicios de usuario, usan dos atributos para identificar al usuario, como se describe en el apartado de Control de acceso y seguridad, página 44.

A continuación se describen las interfaces propuestos en este trabajo, que representan la unificación de contratos en plataformas de e-learning para arquitecturas de integración.

Servicio Web de usuarios (UserService)

La finalidad de este servicio web es la gestión de usuarios y su identificación, las funcionalidades que ofrece son:

- `login(name, key, publicKey)`: Identifica a un usuario con su usuario y contraseña. La clave pública generada por el cliente, utilizada en el control de acceso, página 44.
- `logout(session, certificate)`: Finaliza la sesión de un usuario previamente iniciada con la operación `login`.
- `addUser(name, surname, email, username, password, type, session, certificate)`: Añade un usuario al sistema. Solo existen restricciones en el tipo de usuario, este pude tener los valores “student”, “teaching assistant”, “instructor”. La operación asigna un identificador único al usuario añadido. La operación devuelve el resultado de la misma.
- `deleteUser(id, session, certificate)`: se elimina el usuario identificado con el id del sistema. El resultado de la operación nos indica si la eliminación ha efectuada correctamente.
- `updateUser(id, name, lastname, email, password, type, session, certificate)`: Actualiza los datos del usuario del identificador que se facilita, las restricciones son las mismas que las que tiene la operación `addUser`. Esta operación devuelve el resultado de la misma.
- `getUser(id, session, certificate)`: obtiene la información del usuario del identificador. El resultado de la operación es la información del usuario.
- `getAllUser(session, certificate)`: esta operación devuelve la lista de todos los usuarios del sistema.

Servicio web Curso (CourseService)

Este servicio gestiona la información referente a los cursos, las funcionalidades ofrecidas son:

- `addCourse(number, shortName, fullName, sumary, category, session, certificate)`: Añade un curso a la plataforma, la categoría está restringida a las categoría definidas previamente en la plataforma y no puede existir dos asignaturas con el mismo número nombre corto. La operación crea los elementos de foros y anuncios del curso y asigna un identificador al curso. El resultado de la operación informa del éxito de la misma.
- `deleteCourse(id, session, certificate)`: Elimina el curso identificado por el identificador facilitado. El resultado de la operación informa del éxito de la misma.
- `getAllUsers(id, session, certificate)`: Esta operación lista todos los usuarios de un curso.
- `assigRole(courseID, UserID, Role, session, certificate)`: Matricula a un usuario en un curso, el rol en este curso debe ser “student”, “teaching assistant”, “instructor”. El resultado de la operación nos indica si ha sido posible asignar el rol satisfactoriamente.
- `unAssignRole(courseID, UserID, Role, session, certificate)`: Elimina un rol concreto de un usuario que cursa una asignatura. El rol debe ser uno de los tres posibles roles, “student”, “teaching assistant”, “instructor”. El resultado informa del éxito o fracaso de la operación.

Servicio web de anuncio (AnnouncementService)

La finalidad de este servicio web es la gestión de anuncios de una asignatura. Las funcionalidades que ofrece son:

- `getAllAnnouncement(courseid, session, certificate)`: Devuelve una lista con todos los anuncios de un curso.
- `addAnnouncement(courseid, title, text, session, certificate)`: Añade un anuncio a la asignatura con el id especificado. Al

añadir un anuncio se le asigna un identificador. El resultado informa del éxito o fracaso de la operación.

- `deleteAnnouncement(announceid, courseid, session, certificate)`: Elimina el anuncio identificado por el id de una asignatura. La operación informa del éxito o fracaso de la operación.

Servicio web de foro (ForumService)

Este servicio gestiona los foros, las funcionalidades ofrecidas son:

- `addDiscussion(title, message, courseid, session, certificate)`: Añade una discusión a un foro. Se asigna un identificador a cada discusión añadida.
- `addNewPost(discussionid, postid, title, message, courseid, session, certificate)`: Añade una nueva entrada a una discusión. El post debe estar asociada a una discusión y a una entrada de post padre. Al añadir un post se le asigna un identificador. El resultado de la operación informa del éxito de la misma.
- `deleteDiscussion(discussionId, courseId, session, certificate)`: Elimina una discusión identificada por identificador y todas sus entradas. La operación devuelve el éxito o fracaso de la misma.
- `deletePost(discussionId, postId, courseId, session, certificate)`: Elimina una entrada de una discusión. La operación informa del éxito o fracaso de la operación.
- `getDiscussions(courseId, session, certificate)`: Obtiene todas las discusiones del foro de una asignatura, esta operación no obtiene los posts de las discusiones.
- `getDiscussionPosts(discussionId, courseId, session, certificate)`: Devuelve todos los posts las entradas de una discusión.

Servicio web de plataforma (PlatformService)

Gestión general de una plataforma:

- `getAllUsers(session, certificate)`: Obtiene un listado de todas los usuarios registrados en la plataforma.
- `getAllCourses(session, certificate)`: Obtiene la lista de las asignaturas impartidas en la plataforma.

Servicio web de recursos (ResourceService)

Gestiona los archivos de las asignaturas

- `getCourseFiles(courseId, path, session, certificate)`: Obtiene todos los archivos de una asignatura junto con la información de acceso al recurso.
- `uploadCourseFile(courseId, userId, stream, name, session, certificate)`: Añade un recurso a una asignatura. El éxito de la operación es devuelta.
- `deleteCourseFile(courseId, name,session, certificate)`: Elimina un recurso de una asignatura. La operación devuelve el éxito o fracaso de la misma.

Protocolo de comunicación

El tipo de operaciones que van a soportar los servicios web canónicos engloban operaciones CRUD sobre parte de los recursos de las plataformas, también se van a realizar operaciones que modifican el estado de ciertos elementos del sistema. Estas características hacen que, si bien se podrían usar cualquier protocolo, los basados en RPC sean más apropiados. En concreto, el protocolo SOAP, gracias a que los servicios se encuentran descritos mediante WSDL, facilita la implementación en varias plataformas simultáneamente.

Estructura de los datos intercambiados

A la hora de elegir las estructuras de datos que utilizados en el software desarrollado se ha buscado en primer lugar un lenguaje que sea capaz de representar toda la información y al mismo tiempo la estructura de datos la soporten el mayor número de lenguajes de programación.

La segunda restricción ha limitado el tipo de estructura de datos que se podía utilizar. En concreto, el lenguaje en que está desarrollado Moodle, php, y el framework de SOAP para este lenguaje nusoap[23], tienen problemas a la hora de representar estructuras avanzadas de datos. Por este motivo se han utilizado cadenas de texto para intercambiar información y almacenar dentro de estas cadenas, estructuras de datos más avanzadas. El único atributo que no se ha representado de manera distinta ha sido el que se encuentra en la operación de *upload, stream,* en el servicio web de gestión de recursos. Para este atributo se utiliza un flujo binario de datos. Esta estructura datos no es avanzada y los lenguajes de programación los manipulan como arrays de bytes.

En el intercambio de datos existen dos tipos de datos, los atributos que se envían a los servicios web y el retorno de las operaciones. Para el primer tipo, al poder usar varios atributos, se han utilizado cadenas de texto sin restricciones, y si había algún tipo de restricción en los valores que podía tomar la entrada de la operación, como en el caso de los roles dentro de una asignatura, se ha comprobado en la operación del servicio web.

Tabla 3.1 Información de vuelta por los servicios web

Entidad	**Campos obligatorios**	**Campos opcionales**
Usuario	- userId - userName - name - eMail	- lastName
Anuncio	- ID - title - body	
Curso	- ID - category - name	- fullName - startName
Discusión	- ID - title - author - posts	- created - changed
Post	-ID - title - body - parent - discussion - author	- created - changed

La información devuelta por el servicio, sin embargo, necesita una estructura que permita representar toda la información en una sola cadena de texto. Por ellos se ha utilizado un lenguaje basado en el metamodelo de lenguaje de marcas extensible (*eXtensible Markup Language*, XML).

La estructura de datos de los contenidos en la plataforma y la cantidad de información representada es distinta dependiendo del LMS. El lenguaje que se ha utilizado para representar esta información cuando es devuelta por las operaciones, tiene en cuenta esto y no limita la cantidad de información que se puede incluir. Los elementos devueltos por las operaciones sin embargo si tienen un mínimo de información que deben contener. Estos elementos son:

- *Usuarios*: Representa la información de un usuario, los atributos de este campo son id de usuario (`userId`), nombre de usuario (`userName`), nombre real del usuario (`name`) y correo electrónico (`eMail`). La información adicional, en la implementación que se ha realizado existe información adicional del apellido (`lastName`), ya que en alguna de las plataformas separa estos dos campos.
- *Anuncio*: Representa los anuncios que están publicados en las asignaturas. Los atributos representados son, el identificador (`ID`), título del anuncio (`title`) y cuerpo del anuncio (`body`).
- *Curso*: Representa la información de cada curso. La información que contiene es. Identificador (`ID`), categoría (`category`) y nombre (`name`), además puede incluir como nombre completo del curso (`fullName`) y fecha de comienzo (`startName`).
- *Discusión*: Representa la información de las discusiones. El conjunto de información que contiene engloba el identificador (`ID`), título de la discusión (`title`), autor de la discusión (`author`). Además puede incluir información sobre el momento de creación (`created`) y última modificación (`changed`).
- *Post*: Representa las entradas de una discusión. Incluye el id de la discusión (`ID`), título (`title`), cuerpo (`body`), autor (`autor`), entrada anterior (`parent`) y discusión asociada (`discussion`). Opcionalmente puede existir información sobre el momento de creación (`created`) y última modificación (`changed`).

Esta información, además, puede estar representada como como una lista de elementos, Figura 3.6, o como elemento único, Figura 3.7. Los elementos están anidados en una etiqueta "*item*", si la información es una lista, los elementos están anidados en la etiqueta "*list*".

```
<list>
  <item>
    <userId>4</userId>
    <eid>user1</eid>
    <displayName>user1</displayName>
    <displayLastName>user1</displayLastName>
    <email>user1@campusvirtual.test</email>
  </item>
  <item>
    <userId>6</userId>
    <eid>profe1</eid>
    <displayName>Profe</displayName>
    <displayLastName>Primero</displayLastName>
    <email>profe1@campusvirtual.test</email>
  </item>
  <item>
    <userId>8</userId>
    <eid>alumno1</eid>
    <displayName>Alumno</displayName>
    <displayLastName>Primero</displayLastName>
    <email>alumno1@campusvirtual.test</email>
  </item>
</list>
```

Figura 3.6 Información de una lista de usuarios.

```
<item>
    <userId>4</userId>
    <eid>user1</eid>
    <displayName>user1</displayName>
    <displayLastName>user1</displayLastName>
    <email>user1@campusvirtual.test</email>
</item>
```

Figura 3.7 Información de un usuario.

Control de acceso y seguridad

El control de acceso y seguridad en las plataformas e-Learning es un punto importante. Un fallo en alguno de estos puntos puede comprometer la integridad de la plataforma y de la información que contiene. En este ámbito existen dos puntos importantes para controlar la integridad de acceso y seguridad de la plataforma:

- El acceso indebido a la plataforma. Este control de acceso se realiza mediante un sistema de sesiones, cada vez que se inicia sesión el cliente envía un clave pública aleatoria. El cliente recibe un identificador de sesión vinculado al certificado, estas dos partes son necesarias para que el usuario registrado pueda usar las operaciones de los servicios web. Cuando una operación es invocada desde el cliente, este envía tanto el identificador de sesión como un certificado que autentifica al cliente.
- Robo de información por parte de terceros. La confidencialidad de la información puede verse comprometida si existe una persona infiltrada en el canal de trasmisión. Es por esto que las comunicaciones deben hacerse por medio de un canal seguro. Este tipo de seguridad, al igual que la gran mayoría de las plataformas e-Learning, no está contemplado directamente en la plataforma, delegando al uso de conexiones seguras, como HTTPS, al usuario que quiera implantar esta plataforma.

Información de fallo de servicio

Tabla 3.2 Códigos de error

Código numérico	**Código**	**Descripción**
-1	ERROR_GEN	Error distinto genérico
-2	ERROR_PERMISSION	No se poseen permisos para realizar la operación.
-3	ERROR_NO_SESSION	No existe la sesión
-4	ERROR_INCORRECT_SESSION	La sesión con la que se intenta acceder al servicio es incorrecta
-5	ERROR_SESSION_TIMEOUT	Ha caducado la sesión con la que se intenta acceder al servicio
-6	ERROR_ITEM_NOT_FOUND	El identificador del objeto al que se intenta acceder no se ha encontrado.
-7	ERROR_INCORRECT_ROL	El rol que se ha intentado asignar a un usuario es incorrecto
-8	ERROR_DISCUSSION	La discusión no ha podido crearse correctamente

Las operaciones de los servicios web pueden fallar por distintos motivos como problemas de seguridad o nivel de acceso, información incorrecta, etc. Cuando una operación falla por cualquier motivo, la información referente a dicho fallo es devuelta al cliente para que este, si lo ve oportuno gestione dicho error.

```
<result>
  <code>-5</code>
  <message>Session ID=356</message>
  <session>356</session>
</result>
```

Figura 3.8 Mensaje de error de sesión caducada.

Al igual que en la estructura de datos devueltos por las operaciones, la información del error es representada mediante XML. La información posee un código de error (code) que lo identifica, Tabla 3.2, y opcionalmente puede incluir información adicional como mensaje (message). La Figura 3.8 muestra un error de sesión caducada.

3.2.2 Cliente de los Servicios Web Canónicos: Campus Virtual de Marca blanca

En cliente representa la pieza del diseño encargada de la interacción entre las plataformas y los clientes, y se comporta, por tanto, como cliente de los servicios web ofrecidos por las distintas plataformas y como servidor de los clientes web que quieran acceder a este campus.

La arquitectura del cliente, Figura 3.9, basada en la arquitectura expuesta en el apartado 3.1.3, se compone de tres componentes básicos de lógica de negocio. Cada uno de estos componentes se corresponde con una aplicación del campus virtual: gestión de usuarios, gestión de cursos cursos e integracón de plataformas. La arquitectura dispone de una base de datos que contiene información de usuarios y cursos y es usada por el resto de componentes. Cabe destacar en que en la Figura 3.9 aparecen elementos de la capa de integración para componentes de negocio,y a su vez hay un componente de negocio que se llama “integración”, estando referida esta palabra al uso conjunto de plataformas de e-learning, y no a una capa de la arquitectura multicapa [9].

La lógica de negocio administrativa es la encargada de gestionar los usuarios de la plataforma del campus virtual de marca blanca. Esta funcionalidad incluye comprobar la coherencia de los datos de los usuarios en las distintas plataformas. La integración con el resto de las plataformas se realiza mediante la lógica de negocio de integración donde se realizan las

llamadas a los servicios web canónicos. Esta capa utiliza la base de datos donde tiene almacenado todo el contenido administrativo.

La lógica de negocio de cursos se encarga de gestionar los cursos de la plataforma. Al igual que la lógica de negocio administrativa, esta capa gestiona la coherencia de datos a nivel de cursos, lo que incluye información de que cursos existen y que plataformas tiene alojada cada curso. En la base de datos se encuentra toda la información de los cursos. Para realizar la gestión utiliza los servicios web canónicos proporcionados por la lógica de integración.

La lógica de integración es la encargada de ofrecer la funcionalidad de las plataformas y es la que realiza las llamadas a los servicios web canónicos de las distintas plataformas. Este componente ha sido diseñado con una arquitectura de modelo de negocio. La implementación final se ha hecho con una factoría abstracta por cada servicio web que provee de las interfaces de los servicios web a los componentes que las requieran, Figura 3.10. Los servicios web han sido desarrollados con *Apache Axis 2* framework de java para el desarrollo de servicios web. Este proporciona un completo modelo de objetos y una arquitectura modular que facilita la tarea de añadir funcionalidad y dar soporte para nuevos servicios Web. *Apache Axis 2* usa la tecnología Java-WS como paquete de desarrollo de servicios web.

La interfaz gráfica (Graphic User Interface, GUI) se ha desarrollado mediante JSP que nos permite usar la tecnología java para generar páginas web dinámicamente.

Somos conscientes de que esta sección proporciona una visión arquitectónica de alto nivel del campus virtual construido sobre servicios web, en vez de una visión detallada en términos de diagramas de clase y secuencia UML. No se ha incluido dicha descripción en aras de una mayor concisión, y porque la principal aportación de este trabajo se centra en la definición de los servicios web canónicos y sus implementaciones, en vez de en los campus virtuales construidos sobre dichos servicios web canónicos.

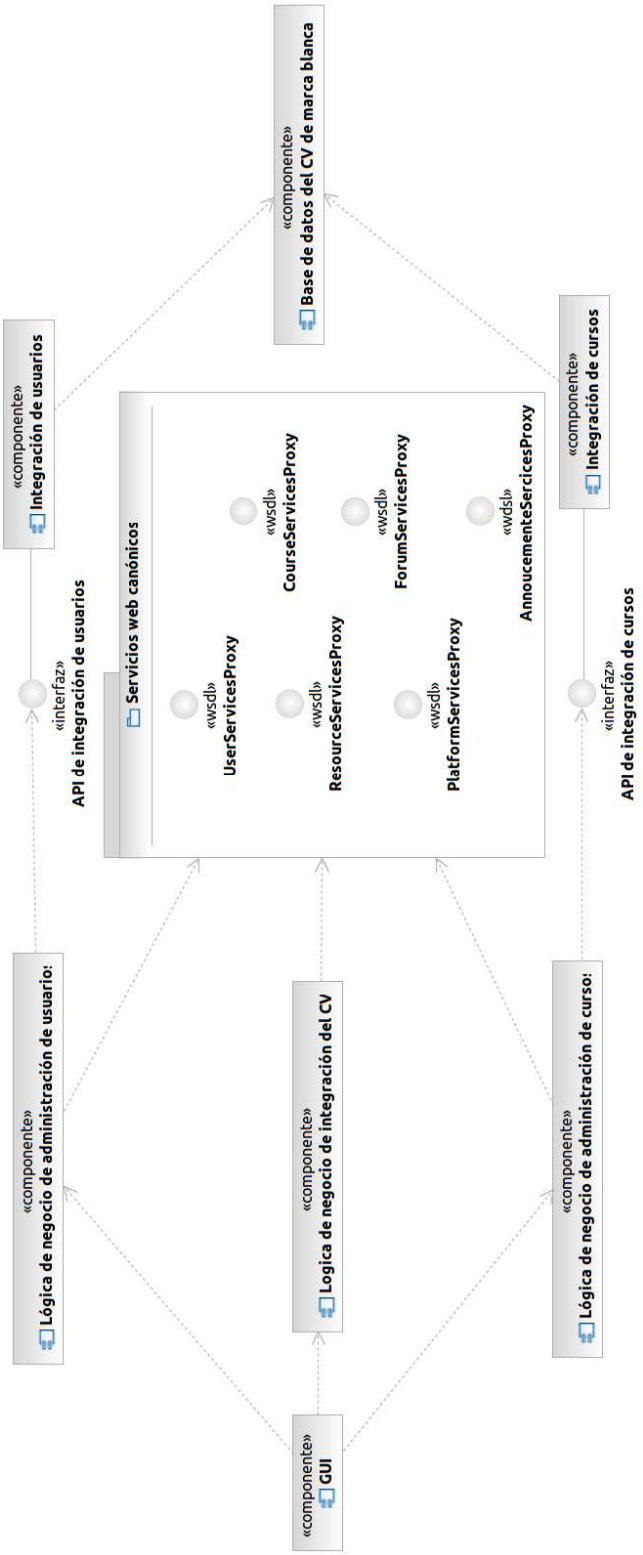


Figura 3.9 Arquitectura del cliente de campus virtual de marca blanca.

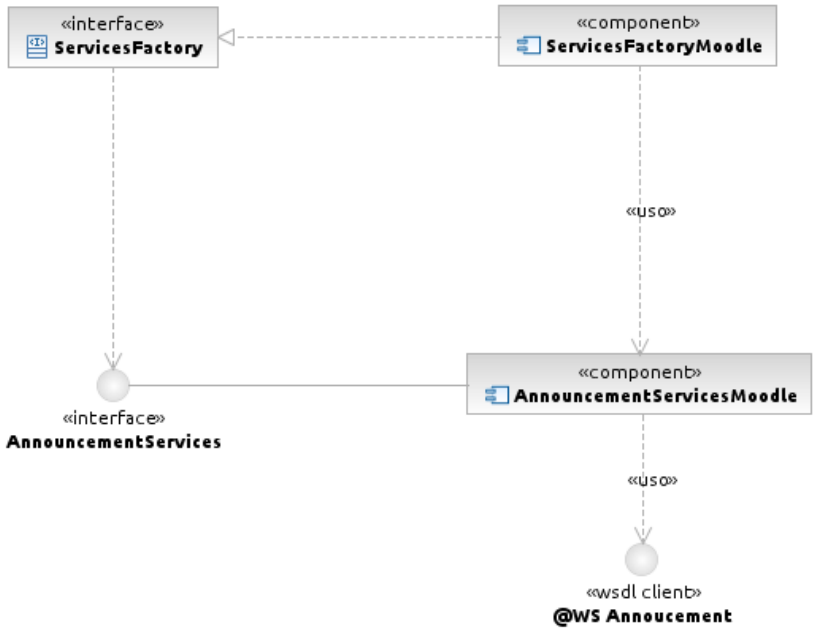


Figura 3.10 Lógica de integración. Factoría de creación de servicios

3.2.3 Proveedor de Servicios Web Canónicos: Plataformas e-Learning

Los servicios web canónicos propuestos, sección 3.2.1, definen un modelo de conjunto de operaciones. La definición de estos servicios web no solo se limita a las operaciones, sino que especifica además el comportamiento y resultado de las mismas.

Las plataformas sobre las que se va a implementar estos servicios web canónicos, aunque son todas LMS, son bastante heterogéneas. Por tanto la implementación de estos servicios web será muy distinta en cada una de ellas. Además, también habrá que realizar trasformaciones de los datos que las distintas plataformas manejan para adaptarlos a la estructura de datos de los servicios web canónicos definidos en esta memoria, página 41.

Para implementar los servicios web canónicos se ha utilizado las herramientas que ofrece la plataforma. Se ha dado prioridad al uso de los servicios web frente a las APIs y, de esta manera, poder implantar los servicios web desarrollados en máquinas distintas. Se ha optado por esta solución en base a: (i) la experiencia nos ha demostrado que los servicios web soportados por las plataformas son más estables que las funciones de las APIs entre distintas versiones de la misma plataforma; y (ii) la composición de servicios web es una práctica bastante extendida y recomendada en SOA (Erl, 2008).

Tal y como hemos comentado en secciones previas de esta memoria, las plataformas e-learning donde se han implantado estos servicios web son Blackboard Learn 9.1, Moodle 2.0 y Sakai 2.7.

A continuación se describe la implementación de los servicios web canónicos propuestos en esta memoria según las distintas plataformas de e-learning consideradas.

Blackboard Learn 9.1

Blackboard dispone de un conjunto de servicios web propios bastante amplio (apartado 2.4.1). Esto ha permitido desarrollar los servicios web canónicos utilizando los proporcionados por la plataforma, Figura 3.11. Las operaciones sobre los foros no se encuentran soportadas y por lo tanto ha sido necesario recurrir a la API de la plataforma para su implementación.

De esta forma la implementación de los servicios web para Blackboard se limita básicamente a *adaptadores* [16] de los servicios web de Blackboard.

La Tabla 3.3 muestra, la dependencias de cada operación de los servicios web canónicos con respecto a los servicios web o el API de Blackboard Learn 9.1. Entre paréntesis aparece el servicio web o componente de la API en donde se encuentra la dependencia.

Despliegue

El despliegue de los servicios en Blackboard necesita un servidor de aplicaciones Java EE que soporte el framework *Apache Axis 2*, Los servicios no necesitan ser desplegados en la misma plataforma donde esté instalada la plataforma aunque es recomendable ya que reduce las latencias de espera para el cliente.

Las pruebas han sido realizadas con un servidor alternativo al de la plataforma donde se encontraban alojados los servicios web canónicos.

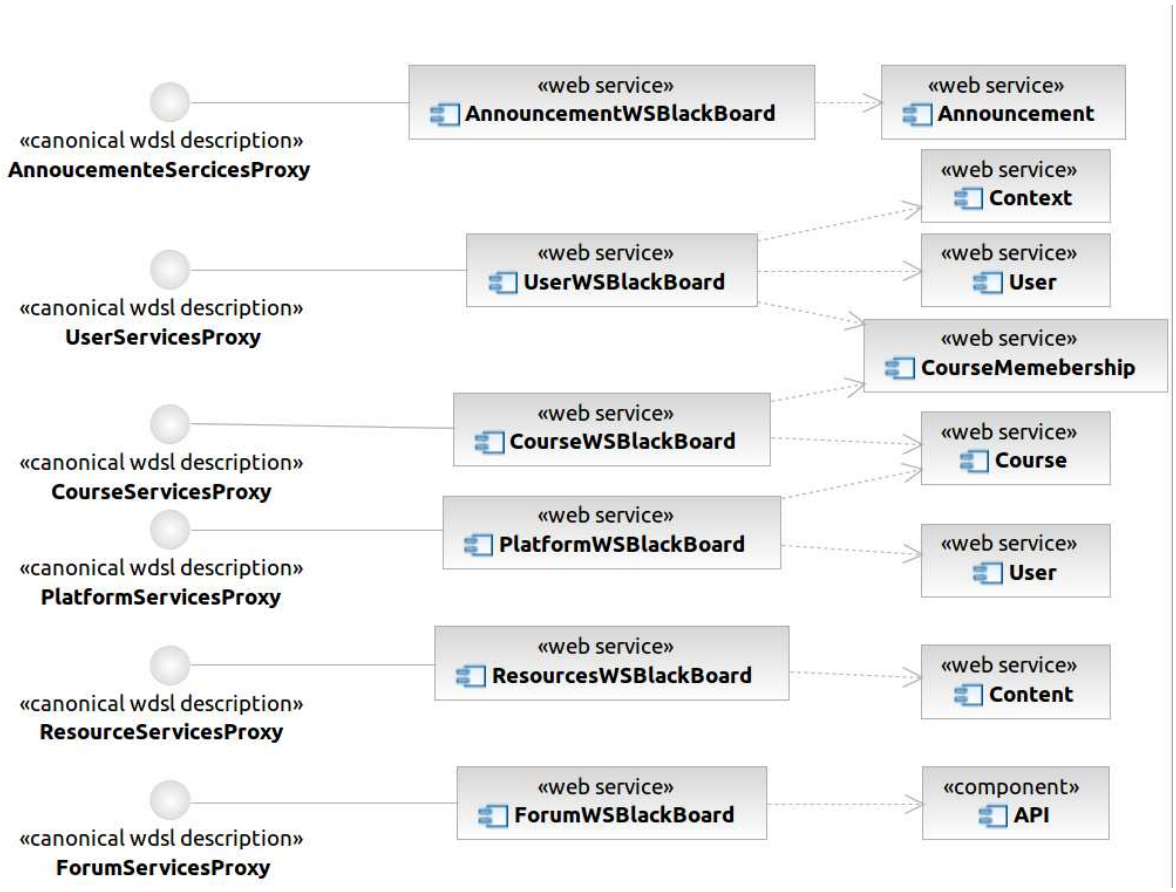


Figura 3.11 Implementación de los servicios web canónicos en Blackboard Learn 9.1

Tabla 3.3 dependencia de las operaciones de los servicios web de Blackboard

Servicio web	Operación	Dependencias servicios web Blackboard
AnnouncementWSBlackboard	getAllAnnouncement	createCourseAnnouncement (Announcement)
	addAnnouncement	deleteCourseAnnouncement (Announcement)
	deleteAnnouncement	getCourseAnnouncements (Announcement)
CourseWSBlackboard	addCourse	CreateCourse (Course)
	deleteCourse	getCourse (Course) deleteCourse (Course)
	getAllUsers	getCourseCategory (Course) getCourseCategoryMembership
	assigRole	getCourseCategory (Course) getUser (User) saveUser(User) saveCorseCategoryMembership (Course)
	unAssignRole	getCourseCategory (Course) getUser (User) saveUser(User) saveCorseCategoryMembership (Course)
ForumWSBlackboard	addDiscussion	getUser (User) getCourse (Course) createNewDiscussionCourse (Forum API)
	addNewPost	getUser (User) getCourse (Course) getDiscussionCourse (Forum API)
	deleteDiscussion	getUser (User) getCourseCategoryMembership (Course) deleteDiscussionCourse (Forum API)
	deletePost	getUser (User) getCourseCategoryMembership (Course) deletePostCourse (Forum API)
	getDiscussions	getCourses (Course) getDiscussionsCourses (Forum API)
	getDiscussionPosts	getCourse (Course) getDiscussion (Forum API) getDiscussionPosts (Forum API)
PlatformWSBlackboard	getAllUsers	getUsers (User)
	getAllCourses	getCourses (Course)
ResourceWSBlackbaod	getCourseFiles	getContentFiles (Content)
	uploadCourseFile	addContentFile (Content)
	deleteCourseFile	deleteContentFile (Content)
UserWSBlackboard	Login	login (Context)
	Logount	logout (Context)
	addUser	inicializeUserWS (User) saveUser (User)

	deleteUser	deleteUser (User)
	updateUser	gateUser (User) saveUser (User)
	getUser	getUser (User)
	getAllUser	getUser (User)

Moodle 2.0

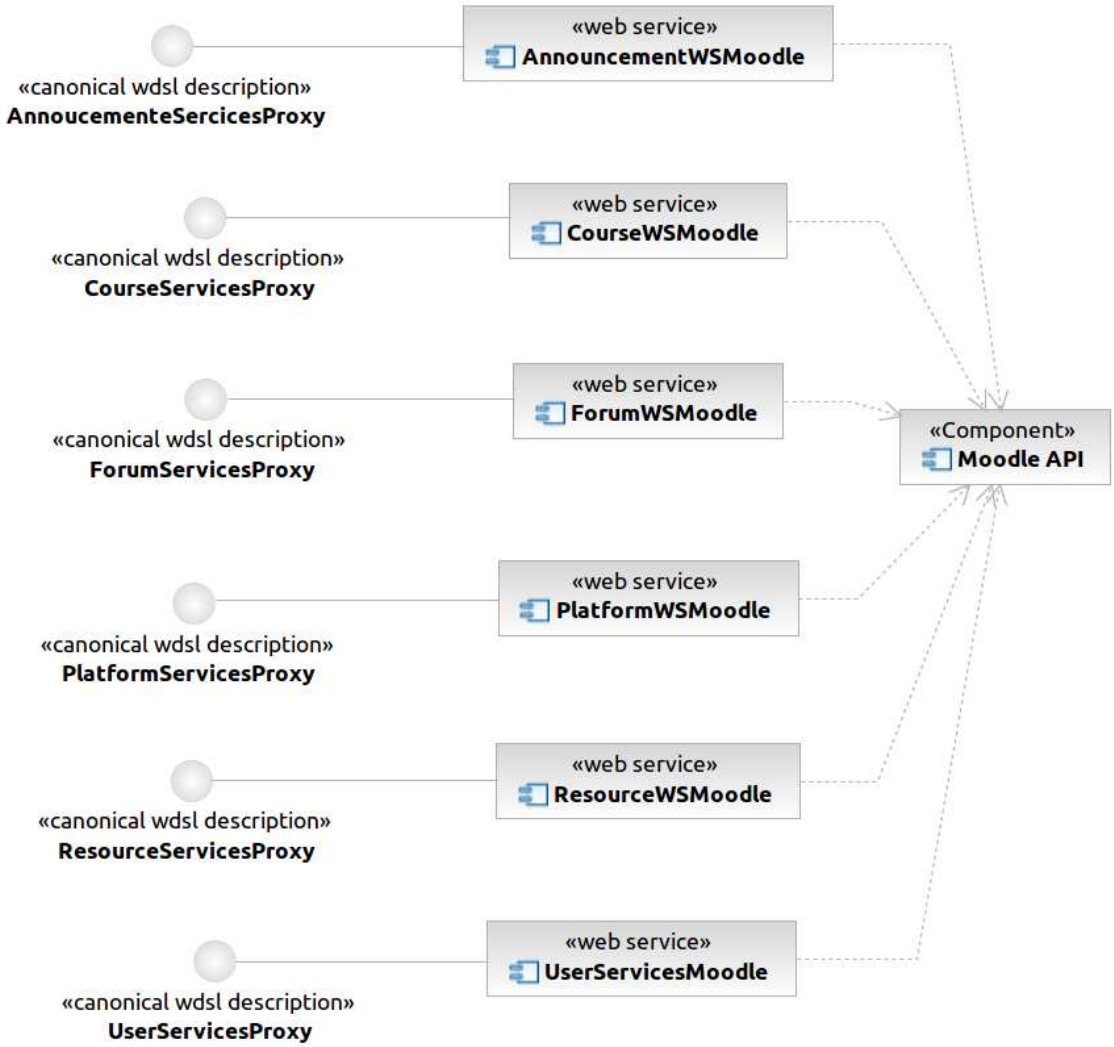


Figura 3.12 Implementación de los servicios web canónicos en Moodle 2.0

Al contrario que Blackboard, el conjunto de servicios web ofrecido por Moodle 2.0 no satisface las necesidades de implementación de los servicios web definidos en esta memoria. Frente a esto se han presentado dos alternativas, aumentar el repertorio de operaciones en la

arquitectura de los servicios web de la plataforma o implementar los servicios web canónicos directamente con la API, Figura 3.12.

Tabla 3.4 Dependencia de las operaciones de los servicios web de Moodle

Servicio web	Operación	Dependencias de librerías de la API
AnnouncementWSMoodle	getAllAnnouncement	forum: obtener anuncios
	addAnnouncement	fórum: añadir anuncio
	deleteAnnouncement	fórum: eliminar anuncio
CourseWSMoodle	addCourse	course: crear el curso forum: crear foro de anuncios y general
	deleteCourse	course: delete course
	getAllUsers	access: Lista de usuario
	assigRole	access: Asignación de rol enrollib: Matricular usuario
	unAssignRole	access: Eliminar el rol del usuario enrollib: Elimina la matricula del usuario.
ForumWSMoodle	addDiscussion	forum: añadir descusión
	addNewPost	forum: obtener discussion y añadir post
	deleteDiscussion	forum: eliminar descusión y post
	deletePost	forum: eliminar post
	getDiscussions	forum: obtener discusión
	getDiscussionPosts	forum: obtener discusión y post
PlatformWSMoodle	getAllUsers	access: listar usuarios
	getAllCourses	data: obtener lista de cursos
ResourceWSMoodle	getCourseFiles	filestorage: obtener las ficheros
	uploadCourseFile	filestorage: crear fichero course: asociar fichero a cursos
	deleteCourseFile	filestorage: eliminar ficheros course: eliminar asociación de ficheros a cursos
UserWSMoodle	Login	moodle: autentificación
	Logout	
	addUser	access: añadir usuario, añadir un rol al usuario
	deleteUser	access: eliminar usuario
	updateUser	access: actualizar información y rol
	getUser	access: obtener usuario
	getAllUser	access: listar usuarios

La primera alternativa es más costosa, puesto que desarrollar operaciones para los servicios web con la arquitectura que Moodle requiere la adaptación de las propias librerías mediante las librerías externas de la plataforma, Figura 2.3. La filosofía de servicios web dinámicos, además, no se adapta al concepto de servicio web canónico.

La segunda alternativa, pese a obligar a desplegar los servicios web en la plataforma anfitriona, nos permite crear directamente los servicios web necesarios. Se reduce el coste de desarrollo y se desarrollan directamente los servicios web canónicos. Esta es, por tanto, la solución elegida en este trabajo. La Tabla 3.4 muestra las dependencias entre los distintos componentes de la API de Moodle y las operaciones de los servicios web canónicos.

Despliegue

Los servicios web de Moodle han sido utilizados con el framework SOAP para php, nusoap. Este framework no añade requisitos adicionales al servidor donde se encuentra Moodle si bien los servicios deben ser desplegados en la plataforma donde se encuentra el servidor.

En la aplicación desarrollada, los servicios web se han desplegado en la ruta WS de la ruta raíz del servidor.

Sakai 2.7

Sakai 2.7 dispone de una arquitectura similar a Blackboard, es decir un conjunto de servicios web estáticos, con estos servicios web se pueden desarrollar gran parte de los servicios web canónicos.

Sin embargo estos servicios web poseen menos funcionalidad que los de la plataforma Blackboard y, por tanto, es necesario ampliar la funcionalidad que la plataforma ofrece.

Para ampliar esta funcionalidad existen dos alternativas, utilizar la API para las operaciones de los servicios web canónicos que necesiten funcionalidad no ofrecida por los servicios web de la plataforma o extender los servicios web de Sakai para que soporten la funcionalidad que se necesita.

Al contar Sakai con un conjunto de servicios web estáticos, desarrollar las operaciones necesarias por los servicios web canónicos mediante la extensión de los servicios propios de la plataforma y utilizar la API tienen costes de desarrollo similar.

Sin embargo la primera opción nos obliga a implantar los servicios web directamente en la plataforma anfitriona, mientras que crear extensión de los servicios web existentes no. Por este motivo se ha optado por extender los servicios web.

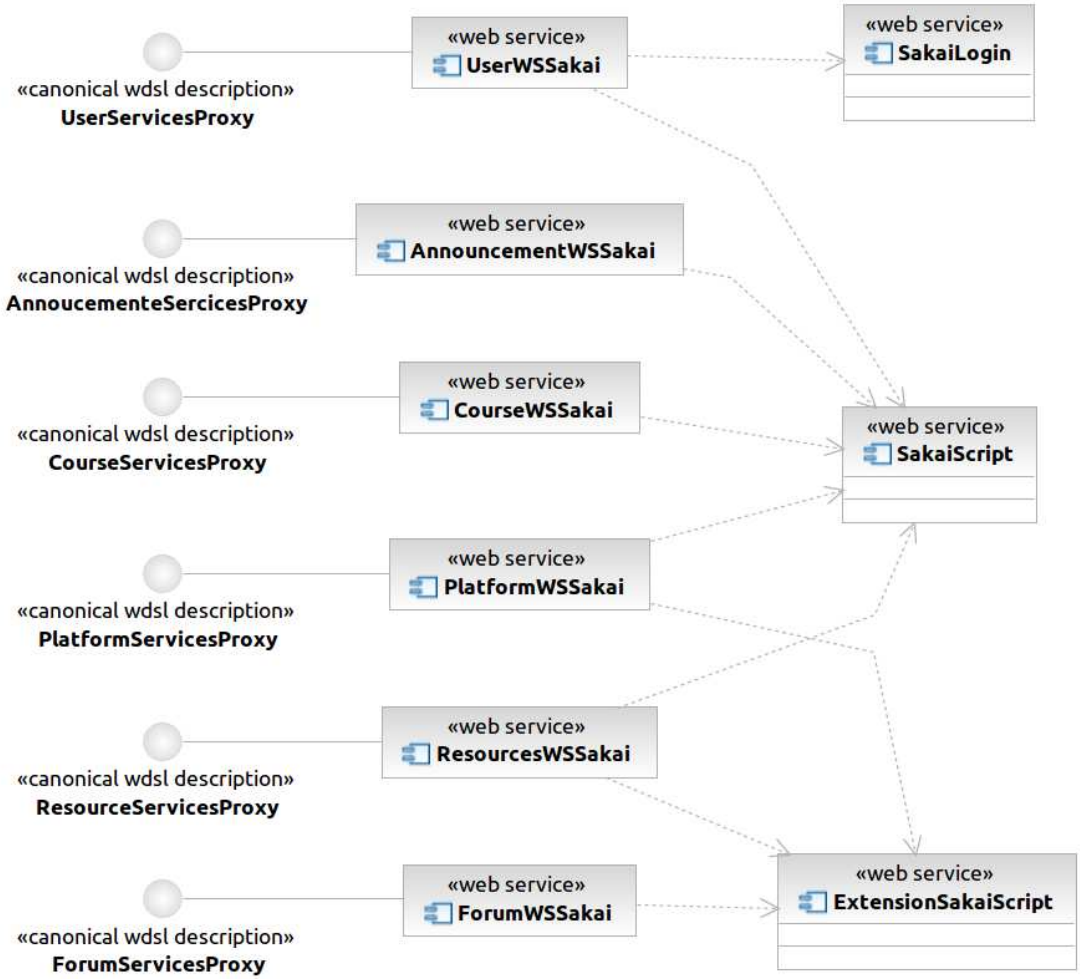


Figura 3.13 Implementación de los servicios web canónicos en Sakai 2.7

La funcionalidad que se va a añadir a los servicios web de Sakai es la encargada de gestionar diversos contenidos de la plataforma, gestionada por el servicio web SakaiScript. Por este motivo y para no romper la distribución de las operaciones de los servicios web, he creado un nuevo servicio, ExtensionSakaiScript, encargado de extender la funcionalidad del servicio web SakaiScript. Al crear un servicio web nuevo no hemos modificado las clases originales de la plataforma facilitando. Esto facilita la adaptación de estos servicios a las nuevas versiones de la plataforma.

Tabla 3.5 Dependencia de las operaciones de los servicios web de Sakai

Servicio web	Operación	Dependencias de las operaciones de los SW
AnnouncementWSMoodle	getAllAnnouncement	ExtensionSakaiScript: getAnnouncements
	addAnnouncement	ExtensionSakaiScript: addAnnouncement
	deleteAnnouncement	ExtensionSakaiScript: deleteAnnouncment
CourseWSMoodle	addCourse	SakaiScript: addNewSite, addNewPageToSite, addNewToolToPage ExtensionSakaiScript: createForum
	deleteCourse	SakaiScript: removeSite
	getAllUsers	SakaiScript: addMemberToSiteWithRole
	assigRole	SakaiScript: removeMemberFromSite
	unAssignRole	ExtensionSakaiScript: getMembersFromSite
ForumWSMoodle	addDiscussion	ExtensionSakaiScript: addForumDiscussion
	addNewPost	ExtensionSakaiScript: addForumPost
	deleteDiscussion	ExtensionSakaiScript: deleteForumDiscussion
	deletePost	ExtensionSakaiScript: deleteForumPost
	getDiscussions	ExtensionSakaiScript: getDiscussions
	getDiscussionPosts	ExtensionSakaiScript: getDiscussionPosts
PlatformWSMoodle	getAllUsers	SakaiScript: getAllUsers
	getAllCourses	ExtensionSakaiScript: getAllSubjects
ResourceWSMoodle	getCourseFiles	ExtensionSakaiScript: getRessource
	uploadCourseFile	ExtensionSakaiScript: addResource
	deleteCourseFile	ExtensionSakaiScript: deleteResource
UserWSMoodle	Login	SakaiLogin: login
	Logout	SakaiLogin: logout
	addUser	SakaiScript: addNewUser
	deleteUser	SakaiScript: removeUser
	updateUser	SakaiScript: changeUserInfo
	getUser	SakaiScript: getUserInfo
	getAllUser	SakaiScript: getAllUsers

La Tabla 3.5 muestra las dependencias entre las operaciones de los servicios web canónicos y los servicios de la plataforma. La Tabla 3.6 muestra las dependencias entre las operaciones de la extensión de los servicios web y la API de la plataforma.

Tabla 3.6 Dependencias entre las operaciones de la extensión de los SW y la API de Sakai

Operación de ExtensionSakaiScript	Dependencias de la API de Sakai
getAllSubjects	SiteService
getSubjects	SiteService
getMemebersFromSite	SiteService, Member
getAnnouncements	AnnouncementService
addAnnouncemenet	AnnouncementService
removeAnnouncement	AnnouncementService
createForum	MessageForumsForumManager, SiteService
addForumDiscussion	MessageForumsForumManager, SiteService
addForumPost	MessageForumsForumManager
deleteForumDiscussion	MessageForumsForumManager
deleteForumPost	MessageForumsForumManager
getDiscussions	MessageForumsForumManager, SiteService
getDiscussionPosts	MessageForumsForumManager
addResource	ContentHostingService
saveReseource	ContentHostingService
deleteResource	ContentHostingService

Despliegue

Para desplegar los servicios web canónicos en la plataforma Sakai 2.7 es necesario tener un servidor de aplicaciones Java con soporte al framework *Apache Axis 2[24]*. Al igual que con las servicios desplegados en Blackboard, no es necesario que estos servicios se encuentren en la misma máquina que aloje a la plataforma, pero sigue siendo recomendable para reducir la latencia en las operaciones.

En la aplicación desarrollada, los servicios web canónicos desarrollados han sido probados en la maquina donde se encontraba alojado el servidor, en concreto se encontraba.

4 Comparativa con Campus Project

Los proyectos de Arquitecturas Avanzadas en Campus Virtuales (AACV) y Campus Project presentan dos enfoques a la necesidad de integración entre plataformas educativas heterogéneas. Aunque ambos proyectos tienen objetivos similares, los enfoques con que plantean la solución son diferentes, y esto se refleja en sus elecciones de diseño.

Para comparar los proyectos hay que analizar en primer lugar cual es el enfoque que tienen, en segundo lugar hay que contrastar los aspectos teóricos de ambos proyectos como la arquitectura y el diseño, por último el estado de desarrollo actual y final de ambos proyectos.

4.1 Enfoque de los proyectos

Pese a que ambos proyectos tienen como objetivo la integración de las plataformas educativas heterogéneas, el enfoque de la integración no es la misma. Para comprender esto es necesario hablar de tipos de integración.

Los tipos de integración entre plataforma representan áreas donde las plataformas pueden interactuar entre ellas o con herramientas comunes y la forma en lo hacen. Además, esta integración permite compartir herramientas externas a las plataformas y sus contenidos. También es posible, gracias a la integración, utilizar los elementos compartidos de las plataformas de forma anónima, es decir, sin conocer la plataforma que está actuando bajo estos recursos. Cada tipo de integración actúa, sobre distintos componentes recursos y aspectos de la plataforma.

Campus Project tiene como objetivo presentar un método para la integración de herramientas y extensiones, y permitir de esta manera, que puedan ser utilizadas indistintamente por Moodle y Sakai, las plataformas educativas que se integran en Campus Project. Este tipo de integración prioriza, en primer lugar, las funcionalidades que permitan interactuar con los contenidos de la plataforma. Esto hace que se puedan crear herramientas comunes entre plataformas que extiendan la funcionalidad de la plataforma. La evolución de la integración permitirá a las herramientas poder gestionar cualquier recurso de la plataforma lo que permitirá, además de interactuar con los contenidos, administrar las propias plataformas.

El proyecto de AACV busca la integración de la administración y gestión de cursos usuarios y contenidos de las plataformas. Su objetivo es permitir el uso de las plataformas mediante un único campus virtual, denominado campus virtual de *marca blanca,* y permitir

mecanismos sencillos para realizar migraciones entre plataformas. La evolución de este proyecto consiste en aumentar en primer lugar los recursos gestionados, y en segundo lugar crear mecanismos para poder utilizar herramientas avanzadas de las plataformas.

Ambos proyectos, por tanto, pueden converger y ofrecer la misma funcionalidad desde distintos enfoques, Campus Project mediante la creación de una herramienta que permita gestionar una plataforma y AACV mediante la extensión de funcionalidades de gestión de contenidos del campus virtual de marca blanca.

4.2 Diseño, arquitectura y tecnología

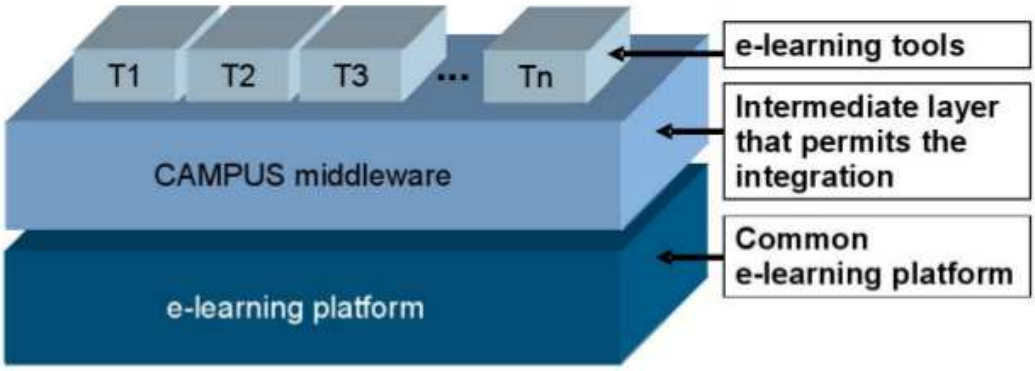


Figura 4.1. Boceto arquitectónico Campus Project [25]

Si se analiza el diseño de la solución de ambos proyectos se pueden diferenciar a varios niveles.

En primer lugar existen diferencias a nivel arquitectónico, estas son debidas a los distintos enfoques de los proyectos, como se ha descrito en el apartado anterior.

Como se describe en el capítulo 3, la arquitectura elegida para el proyecto AACV está compuesta por un conjunto de capas que integran y gestionan diferentes componentes de las plataformas. Estas capas permiten el uso y la administración de las plataformas mediante aplicaciones externas. La arquitectura de la solución incluye además un componente, el campus virtual de marca blanca, encargado de integrar las plataformas mediante las capas de integración.

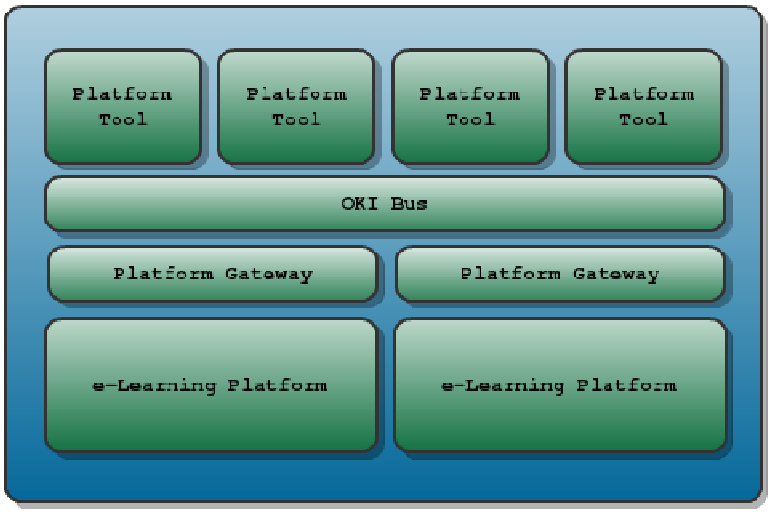


Figura 4.2. Arquitectura de Campus Project con el uso de OKI BUS

La integración de las plataformas en el proyecto Campus Project está basada en una arquitectura compuesta por una capa de operaciones comunes que han de implementarse en todas las plataformas, Figura 4.1 y Figura 4.2. A diferencia de la arquitectura de AACV, no existen distintas capas de arquitectónicas [9] si no que todas las operaciones se encuentran en la misma capa. En el contexto de Campus Project, por tanto, las herramientas de integración de usuarios y cursos que aparecen en AACV, Figura 3.9, son tratadas como cualquier otra herramienta y se incluyen en la capa de integración.

Las diferencias arquitectónicas se encuentran únicamente a nivel teórico y son salvables mediante el desarrollo, en la arquitectura del Campus Project, de herramientas de gestión de administrativa y de cursos.

Una segunda diferencia entre las soluciones consiste en cómo se ha diseñado la capa de integración. Como se ha indicado en el apartado 2.3.2, Campus Project busca implementar el marco de aplicación OSID de OKI, esto conlleva que las operaciones ya están definidas en el framework. El proyecto AACV sin embargo define su propio conjunto de operaciones. La funcionalidad que ofrece OSID y los servicios web canónicos propuestos en este proyecto son similares. OSID ofrece un conjunto de operaciones amplio y estable, los servicios web, sin embargo, al definir las propias operaciones, son más flexibles.

Ambos proyectos tienen en común la elección de servicios web frente a otras tecnologías, para implementar la capa de integración. Esta elección da a las soluciones un grado más de

integración debido a la abstracción del lenguaje de programación en las implementaciones de los clientes de esta capa de integración.

4.3 Estado de desarrollo

Las diferencias que existen a nivel teórico de ambos proyectos son el resultado de diferentes elecciones de diseño de la aplicación. La funcionalidad ofrecida por estas plataformas no está limitada por estas elecciones y ambas pueden converger para ofrecer funcionalidad similar. Sin embargo, el estado de desarrollo de ambos proyectos si presentan una diferencias importantes.

El primer aspecto a analizar se centra en las plataformas sobre las que se han desarrollado los proyectos. Campus Project ha implementado los servicios en las plataformas Sakai y Moodle, que son las principales plataformas educativas de código libre. El proyecto, AACV, además de las plataformas Moodle y Sakai, ha implementado sus servicios sobre la principal plataforma del mercado, Blackboard.

El segundo aspecto que es necesario analizar para ver las diferencias a nivel de desarrollo, es el estado del mismo, es decir, como se encuentran actualmente la implementación de los servicios en ambos proyectos para cada una de las plataformas soportadas.

En el proyecto AACV ha planteado un conjunto de operaciones necesarias para la administración las plataformas educativas y la creación de un campus virtual de marca blanca que, haciendo uso de estas operaciones, sea capaz de presentar una interfaz gráfica común para las plataformas. Para que el campus virtual funcione ha sido necesario implementar todo el marco de operaciones definido en el apartado 3.2.1.

Campus Project busca la creación de herramientas comunes para todas las plataforma, para ello implementa los servicios del framework OKI. Sin embargo no se han implementado todos los OSIDs si no que el desarrollo se ha centrado en aquellos servicios que iban a ser usados por las herramientas diseñadas.

La Tabla 4.1 muestra, los servicios que han sido desarrollados del conjunto de OSIDs expuestos en la sección 2.3.1, así como el estado de desarrollo para cada una de las plataformas sobre la que se desarrolla Campus Project. Puede observarse como el estado de desarrollo de ambas plataformas son distintas.

Tabla 4.1 Estado de las implementaciones OSID finalizadas en Campus Project

OSID	**Sakai**	**Moodle**
Agent	Desarrollado	Desarrollado
Assement	No desarrollado	No desarrollado
Authentication	Desarrollado	Desarrollado
Authorization	Desarrollado	Desarrollado
CourseManagement	Desarrollado	Desarrollado
Dictionary	Desarrollado	Desarrollo incompleto
Filing	No desarrollado	No desarrollado
Grading	No desarrollado	No desarrollado
Hierachy	No desarrollado	No desarrollado
ID	Desarrollado	Desarrollado
Logging	Desarrollado	No desarrollado
Repository	Desarrollado	No desarrollado
Scheduling	No desarrollado	No desarrollado
Shared	Desarrollado	No desarrollado
SQL	No desarrollado	No desarrollado
UserMessaging	No desarrollado	No desarrollado
Workflow	No desarrollado	No desarrollado

Analizando el estado de desarrollo de los proyectos existen un conjunto de funcionalidades básicas en ambas plataformas y que son necesarias para gestionar contenidos más avanzados y que se encuentran implementadas en ambos proyectos para todas las plataformas. Esta funcionalidad incluye gestión de usuarios, autorizaciones, grupos y cursos.

Tabla 4.2 Recursos gestionados por las plataformas

Recurso	**AACV**	**Campus Project**
Anuncios	Soportado	No soportado
Cursos	Soportado	Soportado
Diccionario	No Soportado	Soportado
Foros	Soportado	No soportado
Grupos	Soportado	Soportado
Identificación	Soportado	Soportado
Log	No soportado	Soportado
Usuarios	Soportado	Soportado
Recursos	Soportado	Soportado (Sólo en Sakai)

La gestión de recursos avanzados de las diferentes plataformas, sin embargo, se encuentran, para cada uno de los proyectos, en estados de desarrollo distintos. AACV es capaz de gestionar recursos de foros, anuncios y contenido digital. Campus Project solo es capaz de gestionar recursos digitales en Sakai, en Moodle aún no ha sido desarrollada esta funcionalidad. La Tabla 4.2 muestra un resumen de los recursos que son capaces de gestionar las plataformas.

5 Conclusiones y trabajo futuro

El e-learning va tomando un papel cada vez más relevante en el aprendizaje del ser humano. Contenido digital, enseñanza virtual, anuncios electrónicos, etc. son recursos de este tipo de aprendizaje que son integrados por gran parte de los centros educativos para apoyar o sustituir los recursos didácticos clásicos.

Dentro de los tipos de recursos del e-learning, las plataformas educativas o LMSs representan el núcleo de esta enseñanza. Los LMSs son los encargados de gestionar el material electrónico de aprendizaje, y de esta manera facilitar la labor educativa del personal docente.

Estas plataformas han evolucionado desde su aparición y han pasado de actuar como un sistema de gestión de contenidos a presentar soluciones integrales capaces de gestionar por completo cursos digitales.

En el camino para conseguir que las plataformas se conviertan en gestores de enseñanza virtual, la diversidad de plataformas ha jugado un papel fundamental, favoreciendo un crecimiento de los LMS y una mayor adaptación a las necesidades ofreciendo funcionalidades avanzadas. Esto ha convertido a los LMS en complejas aplicaciones capaces de gestionar no sólo los recursos disponibles en un curso, si no todo los elementos educativos de los cursos e incluso aulas virtuales independientes de la enseñanza clásica.

Sin embargo, la complejidad de estas plataformas y la diversidad de alternativas existentes plantean nuevos problemas. ¿Cómo saber que plataforma es la más indicada para un centro educativo? ¿Qué hacer si son necesarias características avanzadas de varias plataformas distintas? ¿Cómo se puede migrar contenidos entre plataformas? Estos interrogantes plantean cuestiones que, en el estado actual de las plataformas educativas, no son fáciles de resolver y donde una mala elección puede causar graves deficiencias dentro del sistema educativo de un centro o institución.

Ante el reto de plantear solución a los interrogantes aparece un concepto nuevo para este tipo de plataformas que ya ha sido utilizado con anterioridad: integración entre plataformas. Una integración basada en un conjunto de mecanismos comunes capaces de generalizar las funcionalidades similares de todas las plataformas y que permitan el uso de características avanzadas de las plataformas. Es decir, una integración basada en la unificación de contratos en plataformas de e-learning para arquitecturas de integración.

Bajo dicha idea de integración de plataformas, este proyecto ha presentado un marco de integración basado en SOA. Dicho marco consiste en un conjunto de servicios web canónicos que ofrecen la misma funcionalidad en todas las plataformas. Son estos servicios los encargados de adaptar esta funcionalidad a cada una de las plataformas donde se implementan. Este marco ha sido desarrollado, además, en tres de las principales plataformas del mercado: Blackboard Learn 9.1, Moodle 2.0 y Sakai 2.7.

Este marco de adaptación ofrece soluciones a los retos planteados. Gracias a este marco, un usuario no está ligado a una única plataforma, si no que puede interactuar con varias simultáneamente de manera sencilla. Esto permite que puedan ser usadas las características avanzadas de las plataformas. Por último permite al usuario crear mecanismos de integración más sencillos y genéricos entre plataformas.

Dicho marco representa, por lo tanto, un nuevo avance en el desarrollo de las plataformas educativas, en el que los LMSs dejan de ser la pieza central del e-learning para convertirse en una herramienta más de la enseñanza virtual.

Además, este trabajo ha permitido analizar el estado de implementación de los servicios web para tres de las principales plataformas de e-learning. De acuerdo con este análisis se ha podido comprobar que el desarrollo de los servicios web no se encuentran en un estado capaz de cumplir los requisitos de usuarios avanzados. Además, la implementación de funcionalidad de los servicios web no corresponde a menos de la mitad de la funcionalidad ofrecida por las plataformas e-learning.

Blackboard es la plataforma educativa más utilizada y con mayor experiencia y esto se ve reflejado en la implementación de sus servicios. Su catálogo presenta la más amplia funcionalidad. Estos servicios poseen una implementación clásica, al igual que los implementados por Sakai. Sin embargo Blackboard presenta una mayor modularidad en dichos servicios y, como consecuencia, estos servicios son más sencillos de entender y utilizar.

Moodle, por el contrario, presenta un nuevo modelo de servicios web. Al contrario que en las otras plataformas, la arquitectura presenta un modelo dinámico donde el usuario final decide que servicios web utilizar, así como su funcionalidad y control de acceso. Esta arquitectura se compone de un conjunto de operaciones implementadas. Estas operaciones son la base de los servicios web y pueden ser añadidas a dichos servicios definiendo la funcionalidad de los mismos. También es necesario determinar que usuarios o grupos de ellos pueden utilizar el

servicio lo que permite gestionar la política de seguridad de los servicios también dinámicamente. Esta arquitectura añade por tanto una nueva característica de personalización en sus servicios. Sin embargo la funcionalidad ofrecida por la plataforma es aún escasa y apenas si abarca las necesidades básicas de los usuarios. Por último, destacar que la arquitectura de los servicios web de esta plataforma es la más compleja de todas.

Sakai dispone de dos arquitecturas distintas para sus servicios web. En primer lugar dispone de una arquitectura basada en SOAP, en la que los servicios web se clasifican de acuerdo a su funcionalidad. El principal inconveniente de este tipo de servicios es que engloba toda la gestión de recursos en un único servicio web que demasiado extenso. La segunda arquitectura está basada en el protocolo de servicios web RESTful. Esta arquitectura es capaz de gestionar un mayor número de recursos, y además dispone de una organización mucho más razonable.

Si se comparan las interfaces de los servicios web de las tres plataformas analizadas, se puede comprobar que son muy heterogéneas. Esto implica que no pueden ser usadas directamente para la arquitectura propuesta en el proyecto. Moodle, por un lado, presenta una arquitectura completamente diferente y no posee un conjunto de servicios web definidos a priori. Blackboard y Sakai comparten una arquitectura similar, pero las interfaces de las mismas son muy distintos y por tanto la integración no es trivial.

Para resolver los inconvenientes enumerados, este trabajo ha analizado diversas arquitecturas para campus virtuales, implementando aquella más razonable para el estado tecnológico actual. Así, se han definido los servicios web canónicos necesarios para construir un campus virtual de marca blanca que independice al usuario de la plataforma de e-learning subyacente.

Con respecto a proyectos similares, el trabajo realizado en este trabajo incluye la gestión de foros y chats, elementos no implementados por el Campus Project. Además, presenta una filosofía totalmente distinta: en vez de partir de interfaces predefinidos e implementarlos en distintas plataformas, analiza las interfaces disponibles en cada plataforma, generalizándolos en servicios web canónicos. Así, aunque el resultado final es similar, nuestra aproximación es más sencilla y cercana al uso de las plataformas de e-learning de mayor difusión en la actualidad.

Con respecto al trabajo futuro, existen más caminos de investigación en el campo de la integración de plataformas educativas. Estas vías describen métodos para unificar funcionalidad,

crear mecanismos automáticos de migración y uso anónimo de las plataformas. Estos caminos marcan el futuro de la integración y presentan nuevos retos.

Entre estos caminos de integración, la evolución de la arquitectura del proyecto actual planteada en el apartado 3.1.4 presentan la evolución lógica del proyecto. Esta evolución se basa en el uso combinado de distintos LMSs, permitiendo implementar un campus virtual de marca blanca en base a los mejores componentes de cada LMS. Esta arquitectura presenta un gran reto y conlleva el estudio de los resultados para construir un marco de integración total y evitar la creación de una plataforma única que haga uso de las herramientas de otros LMSs.

También cabe destacar que, en el contexto del proyecto AACV, sería muy interesante investigar la implementación de los servicios web canónicos en base a un modelo independiente de la plataforma que fuese traducible a modelos específicos de la plataforma para cada LMS. Esto favorecería la evolución de la implementación de los distintos servicios web canónicos en las nuevas versiones de las plataformas. También favorecería la implementación de dichos servicios web para nuevas plataformas.

Apéndice A. Publicaciones

Web Services Availability in e-learning Platforms

Francisco Huertas
DISIA
Universidad Complutense de Madrid
Madrid, Spain
fhuertas@pas.ucm.es

Antonio Navarro
DISIA
Universidad Complutense de Madrid
Madrid, Spain
anavarro@fdi.ucm.es

Abstract — **Nowadays, integration of e-learning platforms has become a key issue in e-learning. In order to facilitate this integration, most e-learning platforms depict their functionality in terms of web services. However, the availability of these services in every platform is very heterogeneous. In addition, every platform follows its own philosophy when designing its services. This paper analyses three of the most successful e-learning platforms (Blackboard, Moodle and Sakai), identifying their web services, and comparing their readiness for the development of a virtual campus based on these services. The goal of the paper is to facilitate the integration of these platforms in an information technology infrastructure.**

Keywords-LMS; platform integration; Blackboard, Moodle, Sakai

I. INTRODUCTION

In recent years e-learning has had a significant impact in the educational context and it covers a wide set of applications and processes, such as Web-based learning, computer-based learning, virtual classrooms, and digital collaboration. It also includes the delivery of content via Internet, intranet/extranet (LAN/WAN), audio and videotape, satellite broadcasting, interactive TV, CD-ROM, and more [1].

e-learning's success has promoted the appearance of *virtual campuses*: "The virtual campus is a metaphor for the electronic teaching, learning and research environment created by the convergence of several relatively new technologies including, but not restricted to, the Internet, World Wide Web, computer-mediated communication, video conferencing, multimedia, groupware, video-on-demand, desktop publishing, intelligent tutoring systems, and virtual reality [2]. In more recent studies [3, 4, 5, 6] virtual campuses are understood, in a broader sense, as the integration of Information and Communication Technologies in universities at both educational and organizational levels.

Originally, virtual campuses were built on a single e-learning platform, or *Learning Management System* (LMS). However, at present, virtual campuses are evolving towards complex applications built on several e-learning platforms that have to be integrated [7, 8].

In this context, e-learning platforms have evolved in order to facilitate their integration with other applications. This evolution has two different approaches: (i) the inclusion of *Application Program Interfaces* (APIs) to make public the functionalities of the e-learning platform in terms of a code written in the same language in which the e-learning platform has been built; and (ii) web services that allows the integration of e-learning platforms with heterogeneous applications.

This paper analyzes the need for integration of e-learning platforms, as well as the integration facilities provided by three of the most successful e-learning platforms in terms of web services. Thus, Section 2 describes two projects that take advantage of the integration capabilities of e-learning platforms. Sections 3, 4 and 5 describe web services availability in Blackboard, Moodle and Sakai. Section 6 analyzes this availability. Finally, Section 7 presents conclusions and future work.

II. NEED FOR INTEGRATION OF E-LEARNING PLATFORMS

A. VCAA Project

The *Universidad Complutense de Madrid* (UCM) is an old university, founded in 1499, and is currently the largest non-open university in Spain. In the academic year 2009-2010 there were 85,500 students and 6,200 lecturers. In the academic year 2003-2004 the *UCM Virtual Campus* (UCM VC) [9] was set up. The main objective of the project was to place at students and lecturers' disposal all the support that modern information and communication technologies can provide to improve the quality of learning and research activity at the university [10]. The UCM Virtual Campus includes management of the students enrolled in courses and of the content of these courses, as well as facilitating cooperation and communication: work groups, chats, forums, etc. In the present 2010-11 academic year there are 81,000 active students and 3,900 lecturers in the Virtual Campus.

Since its deployment, the UCM VC has had several software architectures for dealing with its e-learning and administrative facilities [7]. At present, the *Virtual Campus Advanced Architectures* (VCAA) project is designing new software architecture for virtual campuses based on *Service-Oriented Architecture* (SOA) [11].

According to this architecture, virtual campuses are built on an integration layer [12] described in terms of abstract interfaces. The e-learning platforms that implement these interfaces can be used to support core e-learning facilities and can be easily interchanged in these virtual campuses. Fig. 1 describes this architecture implementing the SOA architecture in terms of web services.

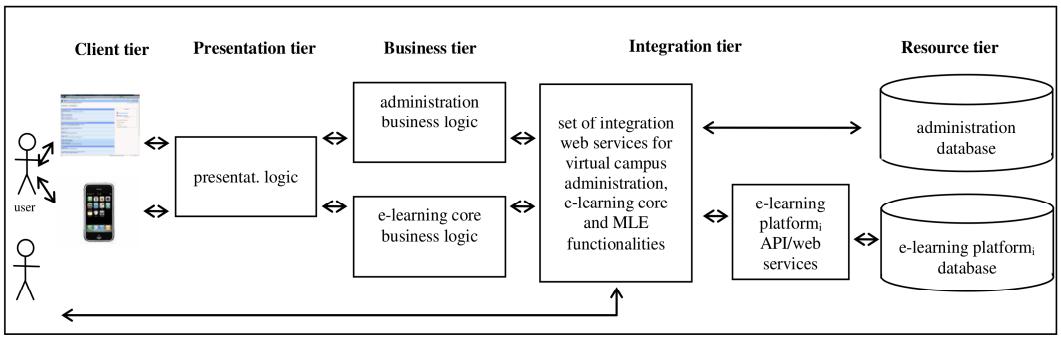


Figure 1. New integration architecture for virtual campuses

The first step for the development of these web services was the analysis of web services availability in e-learning platforms. Precisely, this paper describes such availability.

B. *Campus Project and OKI*

Due to the diversity of platforms and the differences among them, *Campus Project* [8] emerges as a developing community within the area of e-learning. Campus Project is focused on interoperability between systems, ensuring that developments are shared among its members.

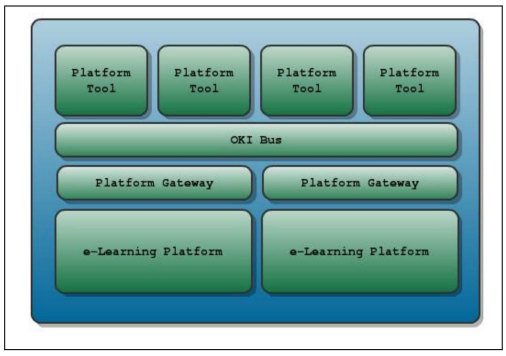


Figure 2. Architecture of Campus Project [13].

Campus Project is based on the assumption that the next step to achieving real interoperability is to adopt a SOA model. When these services implement a clearly-defined interface, it is possible to isolate the interaction mechanisms in a single layer, which provides control over the coupling between the two endpoints. If loose coupling is desired, the layer can be, for example, implemented using web services. Campus Project is this type of architecture (Fig. 2): heterogeneous tools developed in different programming languages that interact with a group of LMS services, but independently of the LMS. This type of architecture could be described as agnostic with respect to the learning tools and LMS used, and coincides with the vision of the VCAA Project.

In order to implement this architecture, Campus Project adopts the *Open Knowledge Initiative* (OKI) proposed by *Massachusetts Institute of Technology* (MIT) [14].

Campus Project is based on OKI, an open and extensible architecture that specifies how the components of LMSs communicate with each other. OKI is specified in the *Open Service Interface Definitions* (OSID) [15], a programmatic interface that describes OKI services.

Although OSID does not aim to provide SOA solutions, the presence of web services in the e-learning platforms can facilitate the implementation of the OSID interfaces in the Campus Project. Therefore, the definition of web services availability in main e-learning platforms can also facilitate the development of the Campus Project.

III. BLACKBOARD LEARN'S WEB SERVICES

Blackboard Learn 9.x [16] is one of the most important e-learning platforms. More than fifty percent of the academic institutions use it as the main LMS [17]. The platform offers many features, and new functionalities can be deployed using its tool called *Building Blocks* [18].

Blackboard's web services prioritize functionality over usability and this makes it different from other e-learning platforms. Thus, Blackboard has the most complete implementation of web services. However, these web services do not include all the functionalities of the e-learning platform deployed as a web application.

A. *Protocols Supported*

Because Blackboard does not prioritize usability, it only implements one web service protocol, *Simple Object Access Protocol* (SOAP) [19]. However, this implementation is enough to support all the implemented functionalities. Therefore, no more protocols are needed.

B. *Architecture of web services*

Blackboard implements architecture similar to Sakai. This architecture is depicted in Fig. 3.

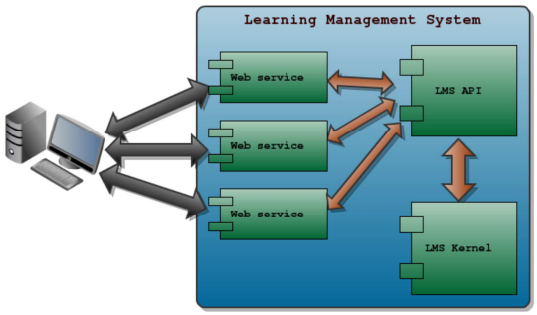


Figure 3. Architecture of web services in Blackboard and Sakai

Blackboard's services are grouped by functionality and resource. These services have basic login and management operations.

Blackboard's web services are:

- *Announcement*: This web service provides methods for creating, modifying and accessing announcements.
- *Calendar*: This web service provides methods for accessing and updating the calendar items in a calendar.
- *Content*: This web service provides methods for creating and accessing content items.
- *Context*: This web service provides the initial methods required for session creation. Therefore it needs to be invoked before any other web service can be used.
- *Course*: This web service interface provides methods for creating and accessing course items.
- *CourseMembership*: This web service provides methods relating to memberships of courses and groups.
- *Gradebook*: This web service provides methods for accessing grade books.
- *NotificationDistributorOperations*: This web service features web service methods for executing notification distributor operations in Blackboard Learn.
- *User*: This web service provides methods for accessing and updating the users, admin users and users' address book entries.
- *Util*: This web service provides secondary methods for accessing and updating global configuration.

C. Security level

To relate a session with web service layer, the LMS usually has a session identifier. This identifier identifies the user in the e-learning platform during a session. This is the principal unsafe point in most e-learning platforms, because if this session identifier is stolen, the user's session can be accessed by hackers.

The majority of LMSs do not implement security in the web service and end-users must implement security policies if needed. However, Blackboard can force the use of *Secure Sockets Layer* (SSL) [20] to access its web services.

IV. Web Services in Moodle

Moodle is currently the main open source e-learning platform and the second most widespread among LMSs [21]. Moodle is implemented in PHP [22], which makes it a highly accessible for any institution that wishes to use a simple LMS. In addition, there are a great many tools deployed by independent developers. Therefore, Moodle has extensive functionality.

Web services in Moodle are implemented following both usability and functionality philosophies. Therefore, Moodle has no static deployment of web services. These services are dynamically deployed and can be adapted to the users' requirements.

However, web service support in Moodle 2.0 is very limited.

A. Protocols Supported

One of the goals of Moodle's web services is usability. Therefore, Moodle implements different web services protocols:

- *XML-Remote Procedure Call* (XML-RPC) [23].
- *Action Message Format* (AMF) [24].
- *Representational State Transfer* (REST) [25].
- *Simple Object Access Protocol* (SOAP).

B. Architecture of web services

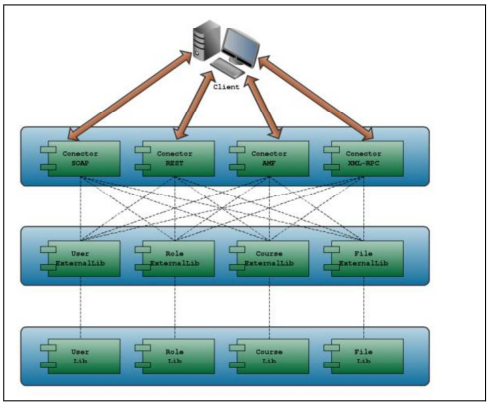


Figure 4. Moodle 2.0 Web service architecture

As Fig. 4 shows, Moodle web services have three tiers that are used to dynamically set up the services:

- *Library/API*. The web services are implemented on this API tier that performs the operations offered by these web services.
- *ExternalLib*. This is the set of the implemented operations that are used by Moodle's web services. It is an extension of available Moodle modules and, therefore, there is an *ExternalLib* for each module. Table I shows the *ExternalLibs* implemented in Moodle 2.0 and their operations.
- *Connectors*. Connectors have two missions: (i) they configure web services according to the user's demands and the operations of the *ExternalLib*; and (ii) they make the web service available. There is a connector per web service protocol (SOAP, REST, AMF and XML-RPC).

TABLE I OPERATIONS SUPPORTED BY EXTERNALLIB IN MOODLE 2.0

Module	Operations
User	moodle_user_create_users moodle_user_delete_users moodle_user_get_users_by_id moodle_user_update_users
Role	moodle_role_assign moodle_role_unassign
Group	moodle_group_add_groupmembers moodle_group_create_groups moodle_group_delete_groupmembers moodle_group_delete_groups moodle_group_get_course_groups moodle_group_get_groupmembers moodle_group_get_groups
Course	moodle_course_create_courses moodle_course_get_courses moodle_enrol_get_enrolled_users
Resources	moodle_file_get_files

C. Access Control

In Moodle, when web services are created, the administrator of the e-learning platform determines their availability to external users. This feature is not as powerful as Blackboard's security policy, but it allows custom levels of access and operations per web service.

V. WEB SERVICES IN SAKAI

Sakai is a modern e-Learning platform promoted by several universities and other institutions [26].

Sakai implements enough web services to fulfill management of the LMS. Unlike Moodle, Sakai's web service architecture follows a classic model. According to this model, Sakai's web services offer all the functionality and cannot be changed without change development. This model offers a list of clearly defined services.

A. Protocols Supported

Sakai web services aim to offer a set of features capable of managing the platform and not adapt the services to all existing technologies. It uses two web services protocols. They offer all the functionality needed by the web services. These are:

- SOAP.
- REST.

B. Architecture of web services

Sakai's web service architecture is based on a classic model, as Fig. 3 shows. There is a set of web services that offer operations to interact with the e-learning platform. The operations of each web service manage similar information.

Sakai's web services architecture has two variants, according to the communication protocol:

1) SOAP Architecture: Sakai uses the *Apache Axis* framework [27] to implement web services. The web services implemented in Sakai are grouped by type of resource managed. They are [28]:

- *SakaiLogin*: These services are responsible for login facilities. Therefore, they need to be invoked before any other service.
- *SakaiPortalLogin*: These services are needed to help connections from *Portal* software such as *uPortal* [29].
- *SakaiScript*: This is a functionally-rich service that includes the main services needed for manipulating users, sites, memberships and permissions on sites.
- *SakaiSession*: This service returns the session information.
- *SakaiSigning*: This service enables external application to verify a user.
- *SakaiSite*: These services allow site handling. It is well worth mentioning that the methods with the word *DOM* [30] are returning strings in a specific *XML* format [31].

2)REST architecture: Sakai's RESTful services are more intuitively described than SOAP services. REST protocol fits in well with the most common application type: *CRUD* (Create, Read, Update, Delete) operations. Each web service manages a specific Sakai resource. They are: connection management, group, group membership, me, presence, search, site, site membership, user, files batch request and activity.

VI. ANALYSIS

This section analyzes and compares the web services availability in the three e-learning platforms analyzed.

This analysis considers the main functionalities needed to use e-learning platforms (e.g. session management, user management, etc.), and analyzes them from the perspective of CRUD operations.

In this analysis four options can be selected for the availability of the CRUD operations for e-learning functionalities:

- *Fully supported*. CRUD operations are implemented. In addition, the operations that allow performance of the same functionalities as the web deployment of the e-learning platforms are also implemented.
- *Supported*. Only CRUD operations are implemented.
- *Poorly supported*. Only part of the CRUD operations is implemented.
- *Not supported*. None of the CRUD operations are implemented.

Table II summarizes this analysis.

According to this analysis, both Blackboard and Sakai are one step ahead of Moodle, although they do not fully implement all the e-learning functionalities used when interacting with e-learning platforms.

In most cases, only web services related to the user, course, announcements and session management are implemented. However those related to calendars, and communications tools (e.g. forums, mail, blog, etc.) are not supported by any e-learning platform.

This is an important drawback because our experience with the UCM VC tells us that communication tools are extensively used by both students and teachers.

TABLE II WEB SERVICE AVAILABILITY IN E-LEARNING PLATFORMS

Web service	***e-learning platform***		
	Blackboard	***Moodle***	***Sakai***
Session	Fully supported	Fully Supported	Fully Supported
User	Supported	Supported	Supported
Role	Supported	Poorly Supported	Supported
Enroll	Fully Supported	Fully Supported	Fully Supported
Course	Supported	Poorly Supported	Supported
Resource	Supported	Poorly Supported	Not Supported
Announcement	Fully Supported	Not Supported	Fully Supported
Forum	Not Supported	Not Supported	Not Supported
Calendar	Supported	Not Supported	Poorly Supported
Notifications	Not Supported	Not Supported	Not Supported
Internal Mail	Not Supported	Not Supported	Not Supported
External Mail	Not Supported	Not Supported	Not Supported
Blog/ Personal Web	Not Supported	Not Supported	Not Supported
Grades	Supported	Not Supported	Not Supported

Finally, the platforms offer the same operations to all protocols supported. Therefore, Blackboard Learn supports SOAP; Moodle supports SOAP, REST, XML-RPC and AMF; and Sakai supports SOAP and REST.

VII. CONCLUSION AND FUTURE WORK

At present, taking into account the requirements of e-learning projects, integration capabilities are needed for most e-learning platforms. Web services enable transparent integration of e-learning platforms in environments such as virtual campuses.

However, according to the analysis carried out in this paper, current implementation of web services does not fulfill the requirements of advanced users. Thus, the implemented functionality in terms of web services is less than half of the functionality offered by the web e-learning platforms.

Blackboard is the most widely used e-learning platform and is also the most experienced. This is reflected in the implementation of its web services, Blackboard offers the greatest functionality implemented as web services. These services have a classical implementation, similar to Sakai. However, Blackboard has increased modularity, and it also has a web service definition per resource or functionality. These features make Blackboard's services very understandable.

Moodle has a characteristic implementation of web services. Unlike other platforms, the web services deployment architecture is dynamic and the end-user creates it. This architecture has a set of operations implemented. These operations can be added to web services and determine all the functionality that they can have. This architecture adds an important personalization feature to Moodle's web services. However, Moodle does not offer enough web services to support the needs of a normal user, although it implements several communication protocols. Perhaps fewer protocols and more web services would be a more balanced approach. In addition, Moodle's architecture for web services is more complex than its counterparts' architecture.

Sakai has a classic web services implementation architecture, classifying them according to their functionality. However Sakai's implementation of web services has a large drawback: all the functions are implemented in a single web service (i.e. a single "*Web Service Description Language (WSDL)*" [32] interface is provided).

Comparing web services' interfaces, they are very heterogeneous. Therefore, they are unsuitable to be directly used in an architecture such as the one promoted by the VCAA Project. Thus, Moodle deployment architecture is completely different and it does not have a stable set of web services. Blackboard and Sakai, despite having similar architectural philosophies, structure their web services in different interfaces. Therefore, their integration with each other is not trivial.

Future work includes the development of a common set of interfaces for e-learning functionalities and their implementation in Blackboard, Moodle and Sakai. The final goal is the development of a virtual campus isolated from its

underlying e-learning platform, as the VCAA Project promotes.

ACKNOWLEDGMENTS

El Ministerio de Educación y Ciencia (TIN2009-14317-C03-01), *La Comunidad Autónoma de Madrid* (S2009/TIC-1650) and *La Universidad Complutense de Madrid* (Group 921340) have supported this work.

REFERENCES

[1] E. Kaplan-Leison E. "ASTD Learning Circuits. Glossary", 2001
http://www.learningcircuits.org/glossary

[2] G.C. Van Dusen G.C. "The Virtual Campus: Technology and Reform in Higher Education", ASHE-ERIC Higher Education Report, Vol. 25, No. 5, 2097.

[3] D.H. Allison, P.B. DeBlois and the EDUCAUSE Current Issues Committee (2008), "Current IT Issues Survey Report", EDUCAUSE Quarterly, Vol. 31, No. 2, 2008
http://www.educause.edu/ir/library/pdf/eqm0823.pdf

[4] R.M. Epper and M. Garn, "The Virtual University in America: Lessons from Research and Experience", EDUCAUSE Centre for Applied Research (ECAR) Research Bulletin, Vol. 2004, No. 2, , 2004
http://www.educause.edu/ir/library/pdf/ERB0402.pdf

[5] PLS RAMBOLL."Studies in the Context of the E-learning Initiative: Virtual Models of European Universities", 2004
http://www.elearningeuropa.info/extras/pdf/virtual_models.pdf

[6] K.C. Green. The 2005 National Survey of Information Technology in U.S. Higher Education: Growing Campus Concern about IT Security; Slow Progress on IT Disaster, 2005

[7] A. Navarro, J. Cristóbal, A. Fernández-Valmayor, C. Fernández, H. Hernanz, S. Guillomía, and F. Buendía, "Towards a New Generation of Virtual Campuses", AICT 2010, 2010

[8] CampusProyect. http://www.campusproject.org

[9] UCM Virtual Campus, https://www.ucm.es/campusvirtual

[10] A. Navarro and A. Fernández-Valmayor, "Conceptualization of Hybrid Websites", Internet Research vol. 17, 2007, pp. 207-228.

[11] T. Erl, "SOA Principles of Service Design". Prentice Hall, 2007

[12] D. Alur, J. Crupi and D. Malks, "Core J2EE Patterns: Best Practices and Design Strategies". 2nd edition. Prentice Hall/Sun Microsystems Press, 2003.

[13] F. Santanach, A. Bertran, C. Ors and M. Gener "Campus Project. Guide for Application Developers", Campus Project, February 2010.

[14] G. Collier and R. Robson; "What is the Open Knowledge Initiative?", Massachusetts Institute of Technology (MIT), September 2002.

[15] "Open Service Interface Definitions – Full Documentation V2.0", Massachusetts Institute of Technology, 2004

[16] Blackboard Learn
http://www.blackboard.com/Platforms/Learn/Overview.aspx

[17] "Managing Online Education", The Campus Computing Project, November 2010.
http://www.campuscomputing.net/sites/www.campuscomputing.net/files/ManagingOnlineEd2010-ExecSummaryGraphics_1.pdf

[18] Greg Ritter "Blackboard, Building Blocks and Libraries" . 2003
http://library.blackboard.com/docs/BuildingBlocks/Bb_datasheet_Building_Blocks_Libraries.pdf

[19] N. Mitra and Y. Lafon "SOAP Version 1.2 Part 0: Primer"; W3C, 27 April 2007.

[20] T. Dierks and E. Rescorla "The Transport Layer Security (TLS) Protocol Version 1.2". RFC 5246, IETF, August 2008, Updated by RFCs 5746, 5878

[21] Campus Computing Project. Study of the role of eLearning. http://www.campuscomputing.net/

[22] PHP: Hypertext Processor. http://www.php.net

[23] D. Winer, "XML-RPC Specification", XML-RPC.com, Jun 2099.

[24] "AMF 3 Specification", Adobe Systems Inc, 2006.

[25] L. Richardson and S. Ruby, "RESTful Web Services", O'Reilly, May-2007

[26] S. Lonn "Sakai 2009 Multi-Institutional Survey Initiative (MISI)", Sakai project, 2009.

[27] "Web Services – Axis" Apache Web Service Project, Last Version 1.4 April 2006.

[28] A.M. Berg and M. Korcuska; "Sakai Web Services: Connecting to the Enterprise"; Open Source , June 2009.

[29] J.A. Lewis "Sakai and uPortal integration", 3 July 2008
https://confluence.sakaiproject.org/download/attachments/39616558/Sakai+uPortal+Integration.pdf

[30] P. Le Hégaret, R. Whitmer and L. Wood, "Document Object Model (DOM)". W3C DOM IG, Jan 2005.

[31] M. Murata, S. St. Laurent and Kohn, "D.: XML Media Types"; RFC 3023, IETF, January 2001

[32] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana; "Web Services Description Language (WSDL) 1.1", W3C, March 2001

Integration Mechanisms in e-learning Platforms

Francisco Huertas
DISIA
Universidad Complutense de Madrid
Madrid, Spain
fhuertas@pas.ucm.es

Antonio Navarro
DISIA
Universidad Complutense de Madrid
Madrid, Spain
anavarro@fdi.ucm.es

Abstract — **Nowadays, integration of e-learning platforms has become a key issue in e-learning. In order to facilitate this integration, most e-learning platforms depict their functionality in terms of APIs and/or web services. Usually, APIs expose the most important functions in platforms. However, the availability of web services in every platform is very heterogeneous. In addition, every platform follows its own philosophy when designing its services. This paper analyses three of the most successful e-learning platforms (Blackboard, Moodle and Sakai), identifying their APIs and web services, and comparing their readiness for the development of a virtual campus based on these services. The goal of the paper is to facilitate the integration of these platforms in an information technology infrastructure.**

Keywords-LMS; platform integration; Blackboard, Moodle, Sakai

I. INTRODUCTION

In recent years e-learning has had a significant impact in the educational context and it covers a wide set of applications and processes, such as Web-based learning, computer-based learning, virtual classrooms, and digital collaboration. It also includes the delivery of content via Internet, intranet/extranet (LAN/WAN), audio and videotape, satellite broadcasting, interactive TV, CD-ROM, and more [1].

e-learning's success has promoted the appearance of *virtual campuses*: "The virtual campus is a metaphor for the electronic teaching, learning and research environment created by the convergence of several relatively new technologies including, but not restricted to, the Internet, World Wide Web, computer-mediated communication, video conferencing, multimedia, groupware, video-on-demand, desktop publishing, intelligent tutoring systems, and virtual reality [2]. In more recent studies [3, 4, 5, 6] virtual campuses are understood, in a broader sense, as the integration of Information and Communication Technologies in universities at both educational and organizational levels.

Originally, virtual campuses were built on a single e-learning platform, or *Learning Management System* (LMS). However, at present, virtual campuses are evolving towards complex applications built on several e-learning platforms that have to be integrated [7, 8].

In this context, e-learning platforms have evolved in order to facilitate their integration with other applications. This evolution has two different approaches: (i) the inclusion of *Application Program Interfaces* (APIs) to make public the functionalities of the e-learning platform in terms of a code written in the same language in which the e-learning platform has been built; and (ii) web services that allows the integration of e-learning platforms with heterogeneous applications.

This paper, an extended version of [9], analyzes the need for integration of e-learning platforms, as well as the integration facilities provided by three of the most successful e-learning platforms in terms of their APIs and web services. Thus, Section 2 describes two projects that take advantage of the integration capabilities of e-learning platforms. Sections 3, 4 and 5 describe web services availability in Blackboard, Moodle and Sakai. Section 6 analyzes this availability, comparing web services functionalities with APIs functionalities. Finally, Section 7 presents conclusions and future work.

II. NEED FOR INTEGRATION OF E-LEARNING PLATFORMS

A. VCAA Project

The *Universidad Complutense de Madrid* (UCM) is an old university, founded in 1499, and is currently the largest non-open university in Spain. In the academic year 2010-2011 there were 83,700 students and 6,200 lecturers. In the academic year 2003-2004 the *UCM Virtual Campus* (UCM VC) [10] was set up. The main objective of the project was to place at students and lecturers' disposal all the support that modern information and communication technologies can provide to improve the quality of learning and research activity at the university [11]. The UCM Virtual Campus includes management of the students enrolled in courses and of the content of these courses, as well as facilitating cooperation and communication: work groups, chats, forums, etc. In the present 2011-12 academic year there are 81,000 active students and 4,000 lecturers in the Virtual Campus.

Since its deployment, the UCM VC has had several software architectures for dealing with its e-learning and administrative facilities [7]. At present, the *Virtual Campus Advanced Architectures* (VCAA) project is designing new software architecture for virtual campuses based on *Service-Oriented Architecture* (SOA) [12].

According to this architecture, virtual campuses are built on an integration layer [13] described in terms of abstract interfaces. The e-learning platforms that implement these interfaces can be used to support core e-learning facilities and can be easily interchanged in these virtual campuses. Fig. 1

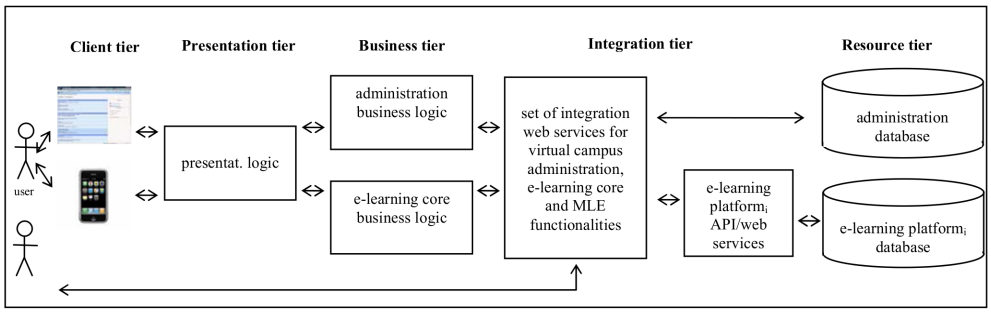


Figure 1. New integration architecture for virtual campuses [7]

describes this architecture implementing the SOA architecture in terms of web services.

The first step for the development of these web services was the analysis of web services availability in e-learning platforms. Precisely, this paper describes such availability, as well as the availability of other integration devices such as APIs.

B. *Campus Project and OKI*

Due to the diversity of platforms and the differences among them, *Campus Project* [8] emerges as a developing community within the area of e-learning. Campus Project is focused on interoperability between systems, ensuring that developments are shared among its members.

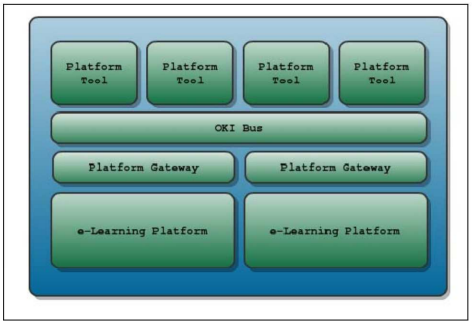


Figure 2. Architecture of Campus Project [14].

Campus Project is based on the assumption that the next step to achieving real interoperability is to adopt a SOA model. When these services implement a clearly-defined interface, it is possible to isolate the interaction mechanisms in a single layer, which provides control over the coupling between the two endpoints. If loose coupling is desired, the layer can be, for example, implemented using web services. Campus Project is this type of architecture (Fig. 2): heterogeneous tools developed in different programming languages that interact with a group of LMS services, but independently of the LMS. This type of architecture could be described as agnostic with respect to the learning tools and LMS used, and coincides with the vision of the VCAA Project.

In order to implement this architecture, Campus Project adopts the *Open Knowledge Initiative* (OKI) proposed by *Massachusetts Institute of Technology* (MIT) [15].

Campus Project is based on OKI, an open and extensible architecture that specifies how the components of LMSs communicate with each other. OKI is specified in the *Open Service Interface Definitions* (OSID) [16], a programmatic interface that describes OKI services.

Although OSID does not aim to provide SOA solutions, the presence of web services in the e-learning platforms can facilitate the implementation of the OSID interfaces in the Campus Project. Therefore, the definition of web services availability in main e-learning platforms can also facilitate the development of the Campus Project.

III. BLACKBOARD LEARN'S WEB SERVICES

Blackboard Learn 9.x [17] is one of the most important e-learning platforms. More than fifty percent of the academic institutions use it as the main LMS [18]. The platform offers many features, and new functionalities can be deployed using its tool called *Building Blocks* [19].

Blackboard's web services prioritize functionality over usability and this makes it different from other e-learning platforms. Thus, Blackboard has the most complete implementation of web services. However, these web services do not include all the functionalities of the e-learning platform deployed as a web application.

A. *Protocols Supported*

Because Blackboard does not prioritize usability, it only implements one web service protocol, *Simple Object Access Protocol* (SOAP) [20]. However, this implementation is enough to support all the implemented functionalities. Therefore, no more protocols are needed.

B. *Architecture of web services*

Blackboard implements architecture similar to Sakai. This architecture is depicted in Fig. 3.

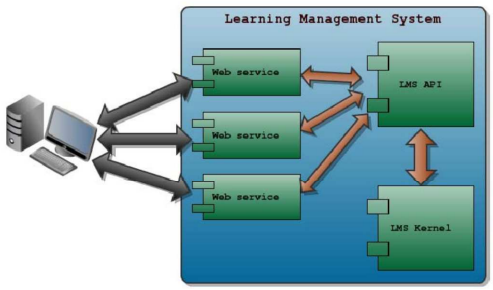


Figure 3. Architecture of web services in Blackboard and Sakai

Blackboard's services are grouped by functionality and resource. These services have basic login and management operations.

Blackboard's web services are:

- *Announcement*: This web service provides methods for creating, modifying and accessing announcements.
- *Calendar*: This web service provides methods for accessing and updating the calendar items in a calendar.
- *Content*: This web service provides methods for creating and accessing content items.
- *Context*: This web service provides the initial methods required for session creation. Therefore it needs to be invoked before any other web service can be used.
- *Course*: This web service interface provides methods for creating and accessing course items.
- *CourseMembership*: This web service provides methods relating to memberships of courses and groups.
- *Gradebook*: This web service provides methods for accessing grade books.
- *NotificationDistributorOperations*: This web service features web service methods for executing notification distributor operations in Blackboard Learn.
- *User*: This web service provides methods for accessing and updating the users, admin users and users' address book entries.
- *Util*: This web service provides secondary methods for accessing and updating global configuration.

C. *Security level*

To relate a session with web service layer, the LMS usually has a session identifier. This identifier identifies the user in the e-learning platform during a session. This is the principal unsafe point in most e-learning platforms, because if this session identifier is stolen, the user's session can be accessed by hackers.

The majority of LMSs do not implement security in the web service and end-users must implement security policies if needed. However, Blackboard can force the use of *Secure Sockets Layer* (SSL) [21] to access its web services.

IV. WEB SERVICES IN MOODLE

Moodle is currently the main open source e-learning platform and the second most widespread among LMSs [22]. Moodle is implemented in PHP [23], which makes it a highly accessible for any institution that wishes to use a simple LMS. In addition, there are a great many tools deployed by independent developers. Therefore, Moodle has extensive functionality.

Web services in Moodle are implemented following both usability and functionality philosophies. Therefore, Moodle has no static deployment of web services. These services are dynamically deployed and can be adapted to the users' requirements.

However, web service support in Moodle 2.0 is very limited.

A. *Protocols Supported*

One of the goals of Moodle's web services is usability. Therefore, Moodle implements different web services protocols:

- *XML-Remote Procedure Call* (XML-RPC) [24].
- *Action Message Format* (AMF) [25].
- *Representational State Transfer* (REST) [26].
- *Simple Object Access Protocol* (SOAP).

B. *Architecture of web services*

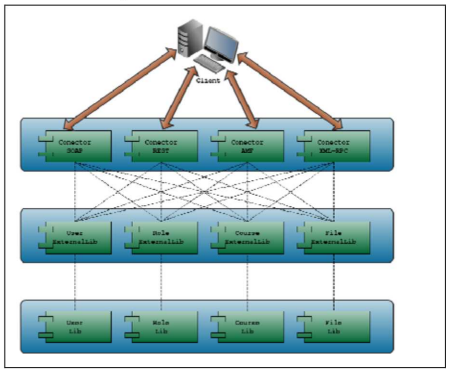


Figure 4. Moodle 2.0 Web service architecture

As Fig. 4 shows, Moodle web services have three tiers that are used to dynamically set up the services:

- *Library/API*. The web services are implemented on this API tier that performs the operations offered by these web services.
- *ExternalLib*. This is the set of the implemented operations that are used by Moodle's web services. It is an extension of available Moodle modules and, therefore, there is an *ExternalLib* for each module. Table I shows the *ExternalLibs* implemented in Moodle 2.0 and their operations.
- *Connectors*. Connectors have two missions: (i) they configure web services according to the user's demands and the operations of the *ExternalLib*; and (ii) they make the web service available. There is a connector per web service protocol (SOAP, REST, AMF and XML-RPC).

TABLE I OPERATIONS SUPPORTED BY EXTERNALLIB IN MOODLE 2.0

Resource	Operations
User	moodle_user_create_users moodle_user_delete_users moodle_user_get_users_by_id moodle_user_update_users
Role	moodle_role_assign moodle_role_unassign
Group	moodle_group_add_groupmembers moodle_group_create_groups moodle_group_delete_groupmembers moodle_group_delete_groups moodle_group_get_course_groups moodle_group_get_groupmembers moodle_group_get_groups
Course	moodle_course_create_courses moodle_course_get_courses moodle_enrol_get_enrolled_users
Resources	moodle_file_get_files

C. Access Control

In Moodle, when web services are created, the administrator of the e-learning platform determines their availability to external users. This feature is not as powerful as Blackboard's security policy, but it allows custom levels of access and operations per web service.

V. WEB SERVICES IN SAKAI

Sakai is a modern e-Learning platform promoted by several universities and other institutions [27].

Sakai implements enough web services to fulfill management of the LMS. Unlike Moodle, Sakai's web service architecture follows a classic model. According to this model, Sakai's web services offer all the functionality and cannot be changed without change development. This model offers a list of clearly defined services.

A. Protocols Supported

Sakai web services aim to offer a set of features capable of managing the platform and not adapt the services to all existing technologies. It uses two web services protocols. They offer all the functionality needed by the web services. These are:

- SOAP.
- REST.

B. Architecture of web services

Sakai's web service architecture is based on a classic model, as Fig. 3 shows. There is a set of web services that offer operations to interact with the e-learning platform. The operations of each web service manage similar information.

Sakai's web services architecture has two variants, according to the communication protocol:

1) SOAP Architecture: Sakai uses the *Apache Axis* framework [28] to implement web services. The web services implemented in Sakai are grouped by type of resource managed. They are [29]:

- *SakaiLogin*: These services are responsible for login facilities. Therefore, they need to be invoked before any other service.
- *SakaiPortalLogin*: These services are needed to help connections from *Portal* software such as *uPortal* [30].
- *SakaiScript*: This is a functionally-rich service that includes the main services needed for manipulating users, sites, memberships and permissions on sites.
- *SakaiSession*: This service returns the session information.
- *SakaiSigning*: This service enables external application to verify a user.
- *SakaiSite*: These services allow site handling. It is well worth mentioning that the methods with the word *DOM* [31] are returning strings in a specific *XML* format [32].

2) REST architecture: Sakai's RESTful services are more intuitively described than SOAP services. REST protocol fits in well with the most common application type: *CRUD* (Create, Read, Update, Delete) operations. Each web service manages a specific Sakai resource. They are: connection management, group, group membership, me, presence, search, site, site membership, user, files batch request and activity.

VI. ANALYSIS

A. Web Services Availability in e-learning platforms

This section analyzes and compares the web services availability in the three e-learning platforms analyzed.

This analysis considers the main functionalities needed to use e-learning platforms (e.g. session management, user management, etc.), and analyzes them from the perspective of CRUD operations.

In this analysis four options can be selected for the availability of the CRUD operations for e-learning functionalities:

- *Fully supported*. CRUD operations are implemented. In addition, the operations that allow performance of the same functionalities as the web deployment of the e-learning platforms are also implemented.
- *Supported*. Only CRUD operations are implemented.
- *Poorly supported*. Only part of the CRUD operations is implemented.
- *Not supported*. None of the CRUD operations are implemented.

Table II summarizes this analysis.

According to this analysis, both Blackboard and Sakai are one step ahead of Moodle, although they do not fully implement all the e-learning functionalities used when interacting with e-learning platforms.

In most cases, only web services related to the user, course, announcements and session management are implemented. However those related to calendars, and communications tools (e.g. forums, mail, blog, etc.) are not supported by any e-learning platform.

This is an important drawback because our experience with the UCM VC tells us that communication tools are extensively used by both students and teachers.

TABLE II WEB SERVICE AVAILABILITY IN E-LEARNING PLATFORMS

Web service	*e-learning platform*		
	Blackboard	***Moodle***	***Sakai***
Session	Fully supported	Fully Supported	Fully Supported
User	Supported	Supported	Supported
Role	Supported	Poorly Supported	Supported
Enroll	Fully Supported	Fully Supported	Fully Supported
Course	Supported	Poorly Supported	Supported
Resource	Supported	Poorly Supported	Not Supported
Announcement	Fully Supported	Not Supported	Not Supported
Forum	Not Supported	Not Supported	Not Supported
Calendar	Supported	Not Supported	Poorly Supported
Notifications	Not Supported	Not Supported	Not Supported
Internal Mail	Not Supported	Not Supported	Not Supported
External Mail	Not Supported	Not Supported	Not Supported
Blog/ Personal Web	Not Supported	Not Supported	Not Supported
Grades	Supported	Not Supported	Not Supported

Finally, the platforms offer the same operations to all protocols supported. Therefore, Blackboard Learn supports SOAP; Moodle supports SOAP, REST, XML-RPC and AMF; and Sakai supports SOAP and REST.

B. APIs and web services

This section compares web services with APIs in the three platforms: Blackboard, Moodle and Sakai. This analysis reviews APIs functionalties, and whether they can be accessed by external applications not deployed in the same machine where the e-learning platform runs, although some functions are only naturally used in the context of the platform's web tier (e.g. visual configuration of the user interface).

APIs can be used to extend the basic functionalities provided by the platform (e.g. a new plugin), or to expose the functionalities to external applications (e.g. a web service). However, web services are usually intended to expose the platform functionalities to external applications.

Therefore, APIs have more functionality than web services included in e-learning platforms. For example, functions to configure web application of the platform need no to be offered in the form of web services. Other example is the support classes including in API libraries, such as special data structures used to manage platform dataset (e.g. a set of cites). However, it can be useful to import these classes using an application library. This importation forces the native platform language support in the importing application.

Fort the sake of classification, API resources can be classified into four categories:

- *Category I*: Functions to manage the platform (e.g. visual configuration of user interface). These functionalities permit modify the platform, web user interface, general parameters, etc. These functions are not usually available as web services.
- *Category II*: Support classes (e.g. data structures used to manage a set of cites). These classes include special data structures, tools to manage dataset, etc. These functions are not usually available as web services but they can be used if the client application includes them.
- *Category III*: Functions used to access to persistent data (e.g. user registration). These classes give access to courses, announcements, etc. of the platform and manage persistent data. These functions are both available in APIs and as web services, because they enable interaction with the platform and its contents.

The category II and category III are related because some classes of the category II manages the data obtained with the functionality offered by the category III

TABLE III API AVAILABILITY FOR EXTERNAL APPLICATIONS

Programming Language	*e-learning platform*		
	Blackboard	***Moodle***	***Sakai***
Native	**Yes**	**Yes** (with dependencies between the API and the rest of classes of the Moodle engine)	**Yes**
No native	**Yes** (if the language supports java library)	**No**	**Yes** (if the language supports java library)

In addition, in order to use the classes of the category II is needed that the platform API can be used by external applications. Table III shows the API availability for external applications, in the native programming language and in other programming languages.

Next sections compare the functionality offered by APIs and web services in each analyzed platform.

1) Blackboard Learn's API and web services

Blackboard's API is characterized by offering a basic set of resources. This API does not have special classes (*category II*) associated with persistent data. In addition, the set of resources managed by this API is very basic. Moreover, this API has not support for advanced resources like chat, internal mail, etc.

Blackboard has also a set of web services for most the management of persistent data. In addition, its API provides a set of special classes used to manage data returned by web services. These classes are different of those offered by the API. However, Blackboard does not have classes to manage permanent dataset.

Additionally, this API is distributed in a JAR library (Java Archive), which can be imported by programming languages that support Java libraries.

Table IV shows a relation between the resources offer by the API and web service availability.

TABLE IV COMPARATIVE BETWEEN API AND WEB SERVICES IN BLACKBOARD

API Resources	*Category*	*Web service availability*
Announcement	II / III	Yes
Bookmark	II / III	No
Calendar	II / III	Yes
Category	I	n/a
Course	II / III	Yes
DataSource	I	n/a
DiscussionBoard	II / III	No
Portfolio	I	n/a
Filesystem	II / III	Partial (Inside content web service)
Monitor	I	n/a
GradeBook	II / III	Yes
Navigation	I	n/a
Role	II / III	Yes (Inside user web service)
User	II / III	Yes

2) Moodle's API and web services

Moodle's API is the complete API and provides the advance set of functionality. Table V shows the API elements and their classification according to the categories described in the beginning of the section.

Moodle includes a set of functions called API Module which includes additional resources: assignment, chat, choice, data, feedback, folder, forum, glossary, imscp, label, lesson, page, quiz, resource, scorm, survey, URL and wiki. These resources are modules that can be included in a course.

Analyzing Table V it is evident that web services offer less functionality than the one provided by the API. Therefore, the use of Moodle's services is unattractive. In addition, in order to use the API into an external application, it is needed to deploy the whole Moodle engine, because there are dependencies between the Moodle API and the rest of classes that make up the e-learning platform. Moreover, the application must be implemented in the same programming language (i.e. PHP).

TABLE V COMPARATIVE BETWEEN API AND WEB SERVICES IN MOODLE.

API Resources	*Category*	*Web service Availability*
Access	III	No
Activity	II	No
Advanced grading	II	No
Backup	III	No
Blog	II / III	No
Calendar	II / III	No

API Resources	*Category*	*Web service Availability*
Comment	II / III	No
Conditional activities	II	No
Course	II / III	Partial
Data definition	III	No
Data Manipulation	III	No
Events	II / III	No
File	II / III	Partial
Filter	IV	No
Form	II / III	No
Groups	II / III	Si
Grade	II / III	No
Logging	III	No
Message	II	No
Module	II / III	No
Navigation	II	No
Output	III	No
Page	II / III	No
Plagiarism	II / III	No
Preferences	II	No
Portfolio	III	No
Question	III	No
Rating	II / III	No
Repository	II	No
RSS	II / III	No
String	II / III	No
Tag	II	No
Time	II / III	No
Unit	III	No
User	II / III	Si

3) Sakai's API and web services

The functionality offered by the Sakai API is very complete. This API can manage all the platform's aspects: access to permanent data, web application configuration, etc. In addition, the Sakai's API has support classes that provide functionality to handle datasets. For example, the Citation classes offer functionalities to: handle a set of citations, configure the appearance of the citations, perform searches, share citations between users, use it in other platform modules, etc.

TABLE VI COMPARATIVE BETWEEN API AND WEB SERVICES IN SAKAI

API Resources	*Category*	*Web service availability*
Announcement	II / III	No
Calendar	II / III	Partial (Only support the copy the calendar between courses)
Chat	III	No
Chefttool	I	n/a
Citation	II	n/a
Courier	I	n/a
Login	III	Yes
Gradebook	II / III	No
Group	II / III	Yes
MailArchive	III	No
Message	III	No
News	II / III	No
Podcast	II / III	No
Portal	III	Yes
Postem	III	No
Presence	II	n/a
Resetpass	III	No
Rights	II / III	No
Role	II / III	Yes
Section	II / III	No
SiteAssociation	I	n/a
SiteManage	II / III	Yes
Taggable	I	n/a
User	II / III	Yes
Warehouse	I	n/a

The use of the Sakai's API as library in external applications is more complex than the use of Blackboard's API because Sakai's API is not distributed as JAR library. However, the source code can be downloaded from the website and includes in an application project.

The persistent data accessible using the web services is very limited compared with the persistent data accessible by this API, as Table VI depicts. This analysis is similar to the analysis performed in the previous section: Sakai's API supports much more functionality than Sakai's web services.

VII. CONCLUSION AND FUTURE WORK

At present, taking into account the requirements of e-learning projects, integration capabilities are needed for most e-learning platforms. Web services enable transparent integration of e-learning platforms in environments such as virtual campuses.

However, according to the analysis carried out in this paper, current implementation of web services does not fulfill the requirements of advanced users. Thus, the implemented functionality in terms of web services is less than half of the functionality offered by the web e-learning platforms.

On the contrary, APIs offer a good set of functionality, but they are intended to be used by applications written in the same language, binding the external application with the platform programming language. In addition, APIs do not have good libraries for facilitating resource manipulation, and therefore, resource manipulation becomes a complex task.

Blackboard is the most widely used e-learning platform and is also the most experienced. This is reflected in the implementation of its web services, Blackboard offers the greatest functionality implemented as web services, and most of persistent data stored in the application is accessible using these web services. These services have a classical implementation, similar to Sakai. However, Blackboard has increased modularity, and it also has a web service definition per resource or functionality. These features make Blackboard's services very understandable. Reagarding, Blackboard's API, it is distributed as a JAR library, which facilitates its use in external applications, whenever the application supports Java libraries.

Moodle has a characteristic implementation of web services. Unlike other platforms, the web services deployment architecture is dynamic and the end-user creates it. This architecture has a set of operations implemented. These operations can be added to web services and determine all the functionality that they can have. This architecture adds an important personalization feature to Moodle's web services. However, Moodle does not offer enough web services to support the needs of a normal user, although it implements several communication protocols. Perhaps fewer protocols and more web services would be a more balanced approach. In addition, Moodle's architecture for web services is more complex than its counterparts' architecture. Regarding, Moodle´s API, its use in external applications is complex because there are dependencies between the Moodle API and the rest of classes that make up the e-learning platform.

Sakai has a classic web services implementation architecture, classifying them according to their functionality. However Sakai's implementation of web services has a large drawback: all the functions are implemented in a single web service (i.e. a single "*Web Service Description Language (WSDL)*" [33] interface is provided). In addtion, only the basic persistent data is accessible using web services. Regarding Sakai's API, it has useful tools that can be used in external applications and, although the API is not distributed as a JAR library, the source code can be exported as a library because Sakai is an open platform.

Comparing web services' interfaces, they are very heterogeneous. Therefore, they are unsuitable to be directly used in an architecture such as the one promoted by the VCAA Project. Thus, Moodle deployment architecture is completely different and it does not have a stable set of web services. Blackboard and Sakai, despite having similar architectural philosophies, structure their web services in different interfaces. Therefore, their integration with each other is not trivial.

Future work includes the development of a common set of interfaces for e-learning functionalities and their implementation in Blackboard, Moodle and Sakai. The final goal is the development of a virtual campus isolated from its underlying e-learning platform, as the VCAA Project promotes.

ACKNOWLEDGMENTS

El Ministerio de Educación y Ciencia (TIN2009-14317-C03-01), *La Comunidad Autónoma de Madrid* (S2009/TIC-1650) and *La Universidad Complutense de Madrid* (Group 921340) have supported this work.

REFERENCES

[1] E. Kaplan-Leison E. "ASTD Learning Circuits. Glossary", 2001
http://www.learningcircuits.org/glossary

[2] G.C. Van Dusen G.C. "The Virtual Campus: Technology and Reform in Higher Education", ASHE-ERIC Higher Education Report, Vol. 25, No. 5, 2097.

[3] D.H. Allison, P.B. DeBlois and the EDUCAUSE Current Issues Committee (2008), "Current IT Issues Survey Report", EDUCAUSE Quarterly, Vol. 31, No. 2, 2008
http://www.educause.edu/ir/library/pdf/eqm0823.pdf

[4] R.M. Epper and M. Garn, "The Virtual University in America: Lessons from Research and Experience", EDUCAUSE Centre for Applied Research (ECAR) Research Bulletin, Vol. 2004, No. 2, , 2004
http://www.educause.edu/ir/library/pdf/ERB0402.pdf

[5] PLS RAMBOLL."Studies in the Context of the E-learning Initiative: Virtual Models of European Universities", 2004
http://www.elearningeuropa.info/extras/pdf/virtual_models.pdf

[6] K.C. Green. The 2005 National Survey of Information Technology in U.S. Higher Education: Growing Campus Concern about IT Security; Slow Progress on IT Disaster, 2005

[7] A. Navarro, J. Cristóbal, A. Fernández-Valmayor, C. Fernández, H. Hernanz, S. Guillomía, and F. Buendía, "Towards a New Generation of Virtual Campuses", AICT 2010, 2010

[8] CampusProyect. http://www.campusproject.org

[9] Huertas, F. and Navarro, A. Web Services Availability in e-learning Platforms. Seventh International Conference on Next Generation Web Services Practices (NWeSP'11), pp. 170-175, 2011.

[10] UCM Virtual Campus, https://www.ucm.es/campusvirtual

[11] A. Navarro and A. Fernández-Valmayor, "Conceptualization of Hybrid Websites", Internet Research vol. 17, 2007, pp. 207-228.

[12] T. Erl, "SOA Principles of Service Design". Prentice Hall, 2007

[13] D. Alur, J. Crupi and D. Malks, "Core J2EE Patterns: Best Practices and Design Strategies". 2nd edition. Prentice Hall/Sun Microsystems Press, 2003.

[14] F. Santanach, A. Bertran, C. Ors and M. Gener "Campus Project. Guide for Application Developers", Campus Project, February 2010.

[15] G. Collier and R. Robson; "What is the Open Knowledge Initiative?", Massachusetts Institute of Technology (MIT), September 2002.

[16] "Open Service Interface Definitions – Full Documentation V2.0", Massachusetts Institute of Technology, 2004

[17] Blackboard Learn http://www.blackboard.com/Platforms/Learn/Overview.aspx

[18] "Managing Online Education", The Campus Computing Project, November 2010.
http://www.campuscomputing.net/sites/www.campuscomputing.net/files/ManagingOnlineEd2010-ExecSummaryGraphics_1.pdf

[19] Greg Ritter "Blackboard, Building Blocks and Libraries" . 2003
http://library.blackboard.com/docs/BuildingBlocks/Bb_datasheet_Building_Blocks_Libraries.pdf

[20] N. Mitra and Y. Lafon "SOAP Version 1.2 Part 0: Primer"; W3C, 27 April 2007.

[21] T. Dierks and E. Rescorla "The Transport Layer Security (TLS) Protocol Version 1.2". RFC 5246, IETF, August 2008, Updated by RFCs 5746, 5878

[22] Campus Computing Project. Study of the role of eLearning. http://www.campuscomputing.net/

[23] PHP: Hypertext Processor. http://www.php.net

[24] D. Winer, "XML-RPC Specification", XML-RPC.com, Jun 2099.

[25] "AMF 3 Specification", Adobe Systems Inc, 2006.

[26] L. Richardson and S. Ruby, "RESTful Web Services", O'Reilly, May-2007

[27] S. Lonn "Sakai 2009 Multi-Institutional Survey Initiative (MISI)", Sakai project, 2009.

[28] "Web Services – Axis" Apache Web Service Project, Last Version 1.4 April 2006.

[29] A.M. Berg and M. Korcuska; "Sakai Web Services: Connecting to the Enterprise"; Open Source , June 2009.

[30] J.A. Lewis "Sakai and uPortal integration", 3 July 2008
https://confluence.sakaiproject.org/download/attachments/39616558/Sakai+uPortal+Integration.pdf

[31] P. Le Hégaret, R. Whitmer and L. Wood, "Document Object Model (DOM)". W3C DOM IG, Jan 2005.

[32] M. Murata, S. St. Laurent and Kohn, "D.: XML Media Types"; RFC 3023, IETF, January 2001

[33] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana; "Web Services Description Language (WSDL) 1.1", W3C, March 2001

6 Bibliografía

[1] E. Kaplan-Leison E. "ASTD Learning Circuits. Glossary", 2001 http://www.learningcircuits.org/glossary

[2] Floyd J. Fowler “Survey research methods”, Sage Publications 2002

[3] A. Navarro, J. Cristóbal, A. Fernández-Valmayor, C. Fernández, H. Hernanz, S. Guillomía, and F. Buendía, "Towards a New Generation of Virtual Campuses", AICT 2010, 2010

[4] G. Collier and R. Robson; "What is the Open Knowledge Initiative?", Massachusetts Institute of Technology (MIT), Septiembre 2002.

[5] A. Chaudhary, J. Guimerá y G. Ruiz “Hacia una Nueva Generación de Campus Virtuales: Integración de Plataformas en el Campus Virtual.”, 2010

[6] F. Santanach, M. Gener y M. Almirall, “The Campus Project: e-learning tools and platforms integration” 2008

[7] F. Huertas, A. Navarro “Web Services Availability in e-learning Platforms”. Seventh International Conference on Next Generation Web Services Practices, NWeSP 2011. IEEE CS. pp. 170-175, 2011.

[8] F. Huertas, A. Navarro. “Integration Mechanisms in e-learning Platforms”. International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM). En prensa.

[9] D. Alur, J. Crupi and D. Malks, "Core J2EE Patterns: Best Practices and Design Strategies". 2nd edition. Prentice Hall/Sun Microsystems Press, 2003.

[10] T. Erl, "SOA Principles of Service Design". Prentice Hall, 2007.

[11] A. ARSANJANI, B. BORGES y K. HOLLEY (2004) "Service-oriented architecture", Web Service Journal, Vol. 4. No. 9, pp. 34-38.

[12] A. Kropp, C. Leue y R Thompson. “Web Services for Remote Portlets Specification”. OASIS, Agosto 2003.

[13] World Wide Web Consortium (W3C), http://www.w3.org/

[14] World Wide Web Consortium (W3C), “Uniform Resource Identifier (URI)”

[15] World Wide Web Consortium (W3C), “eXtensible Markup Language (XML)”]

[16] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: “Elements of Reusable Object-Oriented Software”. Addison-Wesley, 1994.

[17] D. Winer, "XML-RPC Specification", XML-RPC.com, Junio 2099.

[18] N. Mitra and Y. Lafon "SOAP Version 1.2 Part 0: Primer"; W3C, 27 Abril 2007.

[19] L. Richardson and S. Ruby, "RESTful Web Services", O'Reilly, Mayo-2007

[20] "Open Service Interface Definitions – Full Documentation V2.0", Massachusetts Institute of Technology, 2004

[21] Managing Online Education", The Campus Computing Project, Noviembre 2010.

[22] "Web Services – Axis" Apache Web Service Project, Last Version 1.4 Abril 2006.

[23] NUSOAP

[24] "Apache Axis 2" Apache Web Service Project, Last Version 1.6 Mayo 2011.

[25] F. Santanach, A Bertran, C. Ors y Marc Gener “Campus Project. Guide for Application Developers”, Febrero 2010.

[26] Bell M. SOA Modelling Patterns for Service Oriented Discovery and Analysis. Wiley, 2010.

[27] Browning P. Six MLEs – more similar than different. Ariadne 36, http://www.ariadne.ac.uk/issue36/browning, 2003.

[28] Buschamnn F., Henney K., Schmidt D.C. Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing. Wiley 2007.

[29] Cristóbal J., Merino J., Navarro A., Peralta M., Roldán Y., Silveira R. Software Engineering Infrastructure in a Large Virtual Campus. Interactive Technology and Smart Education 8 (3), pp. 172-185, 2011.

[30] Dahlén T., Fritzon, T. Advanced J2EE Platform Development: Applying Integration Tier Patterns. Prentice Hall, 2003.

[31] Daigneau R. Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Addison-Wesley Professional, 2011.

[32] Dewey, B.I., DeBlois, P.B. and the EDUCAUSE Current Issues Committee (2006) Current IT Issues Survey Report, 2006. EDUCAUSE Quarterly, 29 (2) http://www.educase.edu/apps/eq/eqm06/eqm0622.asp, 2006.

[33] Epper, R. M., Garn, M. The Virtual University in America: Lessons from Research and Experience. EDUCAUSE Centre for Applied Research (ECAR)

Research Bulletin, 2004(2) http://www.educause.edu/LibraryDetailPage/666?ID=ERB0402, 2004.

[34] Erl, T. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, 2005.

[35] Erl, T. SOA Design Patterns. Prentice Hall, 2009. Etzion O., Niblett P. Event Processing in Action. Manning Publications, 2010.

[36] Eyre J. De Monfor University Manage Learning Environment Architecture. http://www.jiscinfonet.ac.uk/Resources/external-resources/dmu_MLE_Architecture_v1_000.doc/view, 2006.

[37] Fernández-Valmayor A., Fernández-Pampillón A., Fernández C., Navarro A., Cristóbal J. Implantación de un Campus Virtual de Grandes Dimensiones: el Campus Virtual de la UCM. IEEE RITA 6(4), pp. 167-174, 2011.

[38] Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002.

[39] Green, K.C. The 2005 National Survey of Information Technology in U.S. Higher Education: Growing Campus Concern about IT Security; Slow Progress on IT Disaster Planning, The Campus Computing Project, http://www.campuscomputing.net, 2005.

[40] Harris, M., Lowndahl, J.M., Zastrocky, M. Magic Quadrant for Higher Education Administrative Suites. Gartner RAS Core Research Note, 10 October, 2008

[41] IMS Learning Information Services Specification. Version 2.0. http://www.imsglobal.org/lis/lisv2p0pd/LISspecificationv2p0pd.html, 2010.

[42] IMS Learning Tools Interoperability Specification Version 1.1. http://www.imsglobal.org/lti/index.html, 2011.

[43] JISC infoNET. Creating a Managed Learning Environment (MLE), http://www.jiscinfonet.ac.uk/InfoKits/creating-an-mle, 2006.

[44] Juric M.B., Basha S.J., Leander R., Nagappan R. Professional J2EE EAI. Wrox Press, 2001.

[45] Luckham D. The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Professional, 2002.

[46] Navarro A., Fernández-Valmayor A., Fernández B., Sierra J.L. Conceptualization Prototyping and Process of Hypermedia Applications. International Journal of Software Engineering and Knowledge Engineering 14 (6), pp. 565-602, 2004.

[47] Navarro A., Sierra J.L., Fernández-Valmayor A., Hernanz H. From Chasqui to Chasqui II: an Evolution in the Conceptualization of Virtual Objects. Journal of Universal Computer Science 11 (9), pp. 1518-1529, 2005.

[48] Navarro A., Fernández-Valmayor A. Conceptualization of Hybrid Websites. Internet Research, 17 (2), pp. 207-228, 2007.

[49] Navarro A., Fernández-Valmayor A., Fernández B., Sierra J.L. Characterizing Navigation Maps for Web Applications with the NMM Approach. Science of Computer Programming, 71 (1), pp. 1-16, 2008.

[50] Navarro A. A SWEBOK-based Viewpoint of the Web Engineering Discipline. Journal of Universal Computer Science 15 (17), pp. 3169-3200, 2009.

[51] Navarro A., Cristóbal J., Fernández C., Fernández-Valmayor A. Architecture of a Multiplatform Virtual Campus. Software: Practice and Experience, DOI: 10.1002/spe.1130, 2012.

[52] Navarro, A., Rodríguez-Artacho, M.A., Huertas, F., Cigarrán, J., Buendía F. Soa SOA Integration of a Gathering tool for Retrieving Open Learning Resources in a Modular Virtual Campus. IADIS International Conference e-Learning 2012 (EL 2012). ISBN: 978-972-8939-71-7 © 2012 IADIS

[53] Santacruz L.P., Navarro A., Delgado C., Aedo I. ELO-Tool: Taking Action in the Challenge of Assembling Learning Objects. Journal of Educational Technology and Society, 11, pp. 102-117, 2008.

[54] Steel C., Nagappan R, Lai R. Core Security Patterns: Best Practices and Strategies for J2EE, Web Services , and Identity Management. Prentice Hall, 2005.

[55] Van Dusen, Gerald C. The Virtual Campus: Technology and Reform in Higher Education. ASHE-ERIC Higher Education Report Volume 25, No. 5. Washington, D.C.: The George Washington University, Graduate School of Education and Human Development, 1997.