UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD # 10

"SOFTWARE LIBRE"

FRAMEWORK PARA LA GESTIÓN DE CONTENIDOS EDUCATIVOS.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

AUTOR:

Roxana Cañizares González.

TUTOR:

MSc. Daymy Tamayo Avila.

MSc. David Leyva Leyva.

Ciudad de La Habana, Cuba.

2007 – 2008

DECLARACIÓN DE AUTORÍA

Por este medio declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los 20 días del mes de junio del 2008.

Autores

_____

Roxana Cañizares González

Tutores

_____ _____

MSc. David Leyva Leyva MSc. Daymy Tamayo Avila

DATOS DE CONTACTO

MSc. David Leyva Leyva: Jefe de Departamento de Técnicas de Programación, Facultad 10, Universidad de Ciencias Informáticas. Teléfono: (53) (07) 837 2511. davidl@uci.cu Licenciado en Cibernética-Matemática, Universidad de Las Villas (UCLV), 1988. Máster en Computación Aplicada, Universidad de Las Villas (UCLV), 1995.

Profesor de la Universidad de Holguín desde 1990. Jefe de Departamento de Informática (2003-2005). Participó en un Proyecto de Educación a Distancia en la Universidad del 2001 hasta el 2005, año en el que comenzó a trabajar en la Universidad de Ciencias Informáticas. Ha participado en un gran número de eventos nacionales e internacionales. Le fueron concedidas sendas becas Intercampus en La Universidad de Castilla-La Mancha (1998) y La Universidad Autónoma de Madrid (2000). Trabajó como profesor invitado en la University of Belize (Belice, de 2001 a 2003) y de la Universidad Nacional de Ingeniería (Managua, Nicaragua, 2005). Actualmente líder del Polo Teleformación.

MSc. Daymy Tamayo Avila: Especialista de la Dirección de Teleformación, Profesora de Práctica Profesional V. Facultad 10, Universidad de Ciencias Informáticas. Carretera a San Antonio de los Baños Km. 2 ½, Torrens, Boyeros, Ciudad de La Habana. Cuba. Teléfono: (53) (07) 837 2453. daymy@uci.cu Ingeniera Informática, Universidad de Holguín, 2005. Máster en Informática Aplicada, UCI-CUJAE, 2007.

Culminó sus estudios con índice académico de 4.94 puntos, obteniendo Título de Oro. Fue Alumna de Alto Aprovechamiento Académico y Alumna Ayudante en la Disciplina de Técnicas de Programación de Computadoras desde el segundo año de su carrera. Participó en un Proyecto de Educación a Distancia en la Universidad del 2001 hasta el 2005, año en el que comenzó a trabajar en la Universidad de Ciencias Informáticas. Ha participado en un gran número de eventos nacionales e internacionales. Ha sido tutora de varios Trabajos de Curso y Diploma, y de Alumnos Ayudantes. Actualmente perteneciente al Polo Teleformación, desempeñando el rol de arquitecta principal.

PENSAMIENTO

..."Si un hombre es perseverante, aunque sea duro de entendimiento se hará inteligente y aunque sea débil se transformará en fuerte".

Leonardo Da Vinci

AGRADECIMIENTOS

En especial a mis padres, por todo el apoyo y la confianza que siempre han depositado en mí.

A mi compañero que siempre ha estado a mi lado, y me ha ayudado siempre Gracias Jesús.

A mis amigos, que de una forma u otra me ha ayudado en todos estos años.

A Javier por el apoyo incondicional que me ha brindado.

A todos los profesores; en especial a ti Daymy.

Roxana Cañizares González

Quisiera agradecer a todas las personas que han tenido que ver en todos estos años de estudio con mi formación, en especial a mis profesores; Israel y Milagro, gracias por encaminarme en mis primeros pasos, a Nury por toda tu ayuda, a todos los que tuvieron que ver con la realización de este trabajo, a mis compañeros; en especial a Javier, a Jesús por toda su dedicación y amor, a mí mis tutores; Daymy y David por toda su confianza.

¡Muchas Gracias!

DEDICATORIA

A quienes me enseñaron a nadar contra la corriente, mis amados padres, Manuel Valentín y Blanca Nieves, gracias por su apoyo, compresión y su inmenso amor.

A toda mi familia en particular a mi abuelita, que la quiero mucho.

A mis amigos especialmente a Yaité, Keisy e Ineldo, por todas las penas y alegrías vividas juntas, gracias por su amistad.

A mis compañeros del proyecto principalmente a Javier por su apoyo incondicional.

A mi compañero que me ha ayudado y apoyado en todo momento, gracias, Jesús.

A Nury , Milagro, Vude, Maricela, Estela, Adiaris, Eloisa que de una forma u otra me han ayudado en todos estos años.

En fin a todas las personas que han creído en mí.

¡Muchas Gracias!

RESUMEN

La evolución del uso de las Tecnologías de la Información y las Comunicaciones en la educación ha posibilitado el desarrollado del e-learning y con ello la construcción de herramientas capaces de gestionar contenidos educativos. Uno de los aspectos claves en este ámbito y que requiere de grandes esfuerzos, es la creación de contenidos con calidad. Lo idóneo sería poder reutilizar dichos contendidos y que además estos sean interoperables. Esto sólo es posible si dichos contenidos son estandarizados y diseñados como Objetos de Aprendizaje (OA). Del mismo modo, las herramientas que gestionan OAs deben ser interoperables entre sí. Varios proyectos en los últimos años han comenzado a utilizar estos estándares con el objetivo de lograr sistemas e-learning integrales.

En este trabajo se realiza una propuesta arquitectónica de un framework que garantiza la interoperabilidad entre herramientas para la gestión de contenidos educativos. La propuesta se basa en una Arquitectura Orientada a Servicios, usando los estándares de interoperabilidad IMS-DRI, SQI y OKI. Se identifican los posibles niveles de madurez que puede alcanzar el framework, proponiendo cómo alcanzar hasta el nivel tres mediante una primera iteración. Finalmente, se plantea realizar otras iteraciones hasta alcanzar todos los niveles de madurez identificados.

Palabras clave: e-learning, objetos de aprendizaje, interoperabilidad, SQI, OKI, SOA, servicios web.

INDICE

INDICE DE FIGURAS

INTRODUCCIÓN

Una de las ramas del quehacer humano que más cambios ha tenido en la historia debido al desarrollo tecnológico, ha sido la educación. Desde la invención de la escritura hasta la actualidad los educadores han puesto en práctica todas las herramientas tecnológicas que le han sido de ayuda para lograr un progreso en el aprendizaje.

La educación tradicional ha tenido como elemento principal al profesor y al alumno. Con el desarrollo de Internet y las Tecnologías de la Información y las Comunicaciones (TIC), han surgido nuevos modelos de enseñanza aprendizaje donde el alumno es el elemento principal y alrededor de él tiene el resto de los elementos encargados de formarlo (Vázquez, 2007). Un ejemplo de ello lo constituye el e-learning (electronic learning).

Existen varias formas de definir el e-learning, pero en este trabajo se tomará como el "*conjunto de tecnologías, aplicaciones y servicios orientados a facilitar la enseñanza y el aprendizaje a través de Internet/Intranet, que facilitan el acceso a la información y la comunicación con otros participantes*." (Red TTnet, 2005)

Innumerables herramientas de soporte al aprendizaje han surgido con el objetivo de llevar éste nuevo tipo de enseñanza a todo el mundo. Tal es el caso de los Learning Management Systems (LMS) o Sistemas para la Administración del Aprendizaje. Estos sistemas permiten la creación y gestión de cursos sin la necesidad de conocimientos profundos de programación o de diseño gráfico, ya que brindan una interfaz cómoda para su confección y accesible para cualquier persona, así como dar seguimiento a las actividades de aprendizaje.

Estas plataformas presentan una limitante respecto a los materiales docentes, puesto que actúan como plataformas de distribución, donde la mínima unidad didáctica es el curso. Sin embargo, la tendencia en la actualidad es granular los contenidos en pequeñas unidades llamadas objetos de aprendizaje (OA), originados bajo el paradigma orientado a

objetos, y que tienen la finalidad de maximizar el número de contextos en que puedan ser utilizados (Tamayo, 2007)

Existe una gran diversidad respecto a la definición de OA, en este trabajo se considerará como tal "*cualquier recurso con una intención formativa, compuesto de uno o varios elementos digitales, descrito con metadatos, que pueda ser utilizado y reutilizado dentro de un entorno e-learning".* (López, 2005)

Para su real utilidad los OA deben estar almacenados en Repositorios de Objetos de Aprendizaje (ROA), los cuales permiten realizar búsquedas eficientes a través de los metadatos (Tamayo, 2007). Para la creación de los OA es necesaria la utilización de Herramientas de Autor, las que a su vez posibilitan la reutilización de los mismos a través del uso de estándares.

Un *estándar* es un patrón, una tipificación o una norma de cómo realizar algo (AulaGlobal, 2007). Existen diversos grupos e instituciones que han trabajado en la estandarización del e-learning, entre ellos se destacan IEEE[1], AICC[2], IMS[3], ADL[4], los resultados principales de IMS están relacionados con los metadatos y su resultado más relevante es el estándar LOM (Learning Object Metadata), que establece cómo deben ser descritos los OA. ADL trabaja en coordinación con IEEE, IMS y AICC, y diseñó uno de los estándares más difundidos en la actualidad: el Modelo de Referencia de Objetos de Contenido Compartibles (SCORM) orientado a empaquetar OA descritos mediante (LOM)[5].

---

[1] IEEE The Institute of Electrical and Electronics Engineers (http://www.ieee.org).

[2] AICC Aviation Industry Computed Based-Training Comitee (http://www.aicc.org/).

[3] IMS Global Learning Consortium (http://www.imsproject.org/).

[4] ADL Advanced Distributed Learning (http://www.adlnet.org/.)

[5] Estos estándares serán explicados en el capítulo 1.

Uno de los aspectos del e-learning que en los últimos años ha evolucionado hacia la estandarización es en la *interoperabilidad***,** definida por la IEEE como "*la habilidad de dos o más sistemas o componentes para intercambiar información y para usar la información que ha sido intercambiada*". En este aspecto, los estándares más utilizados son Simple Query Interface (SQI), The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), Open Knowledge Initiative (OKI), Z39.50[6], IMS Digital Repositories Interoperability (IMS-DRI).

Con la aparición de los servicios web, y la Arquitectura Orientada a Servicio (SOA), se ha dando un paso gigantesco en cuanto a la interoperabilidad entre aplicaciones, ya que ésta supone el establecimiento de nuevos mecanismos de comunicación, haciendo posible que sistemas desarrollados en diferentes plataformas y diferentes lenguajes de programación, puedan interactuar.

Numerosas iniciativas (CAMPUS, 2007), (Santanach. F, Casamajó. J, Casado. P, et all, 2007), (JISC, DEST, CETIS), (Wilson .S, Olivier .B, Jeyes .S et all, 2003) entre otras, confirman la tendencia hacia la integración de sistemas e-learning, utilizando estos estándares y arquitectura a través de un *framework*[7]. Este término es muy utilizado en el campo de la informática, normalmente tratado como framework de desarrollo, que no es más que un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación, proporciona una estructura al código fuente, forzando al desarrollador a crear código legible y más fácil de mantener (Potencier. F & Zaninotto. F, 2008).

Por otro lado, específicamente en el ámbito del e-learning, algunas organizaciones (JISC, DEST, CETIS) lo definen como: "*Una colección de géneros de servicios cooperativos dentro de un dominio. Un vocabulario de componentes, usado para modelar conceptos y*

---

[6] Z39.50 también conocida como "Information Retrieval (Z39.50); Application Service Definition and Protocol Specification for Open Systems Interconnection"

[7] Traducido al español como Marco de Trabajo.

comportamientos dentro de un dominio o para crear un tipo particular de infraestructura (sistema o ambiente)". Donde "géneros de servicios" no es más que una capacidad genérica o abstracta expresada en términos de comportamiento.

A lo largo de este trabajo se referirá un framework representando esencialmente una arquitectura de software referida a un dominio, de modo que modele las relaciones entre las entidades de dicho ámbito.

En Cuba, varias universidades utilizan desde hace algunos años el e-learning en los procesos docentes. A partir del 2000 que se comenzó a utilizar una plataforma web para impartir cursos a distancia llamada Microcampus[8], y junto a ella surgieron otras herramientas creadas por Instituciones nacionales tales como AprenDist[9], del Instituto Superior Politécnico "José Antonio Echeverría" (CUJAE); SEPAD de la Universidad Central de Las Villas (UCLV); y EduDist[10] de la Universidad de Holguín. En los últimos años se ha generalizado el uso de esta modalidad, extendiéndose el uso del LMS Moodle, una de las plataformas de libre distribución más difundidas en la actualidad.

Algunas instituciones han integrado varias herramientas e-learning, como es el caso de la Universidad Virtual del Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA), como parte del Programa Ramal científico técnico del CITMA, denominado Red de la Ciencia. La misma está integrada por un aula, un laboratorio, una biblioteca, un museo, algunos eventos, todos estos virtuales y gestionados a través de una oficina. Esta plataforma utiliza CORBA[11], arquitectura que en los últimos años ha disminuido su uso debido a desventajas como complejidad y la incompatibilidad entre implementaciones,

---

[8] Una plataforma creada por un grupo de trabajo de la Universidad de Alicante, España.

[9] Basada en el proyecto Mundicampus.

[10] Plataforma desarrollada por el Proyecto de Educación a Distancia de la Facultad de Informática-Matemática de la Universidad de Holguín.

[11] CORBA: Common Object Request Broker Architecture *(http://www.corba.org/).*

debido a que las diferentes formas de implementar la gestión de la concurrencia, hacen difícil la portabilidad del código.

Otra institución que ha realizado alguna acción por la integración es la Universidad Virtual de Salud (UVS), actualmente está soportada utilizando software libre. Sus principales componentes son: plataforma para la gestión docente centralizada, plataforma educativa, repositorio de contenidos que permite sindicación y los metadatos utilizan el estándar LOM para la referenciación y búsqueda (Jardines, 2007). No obstante, a pesar de que estas aplicaciones utilizan estándares, no son completamente interoperables entre sí, los contenidos sólo son referenciados y no son diseñados como OA. (Tamayo, 2007)

Con el objetivo de utilizar las ventajas que proporcionan los LMS, en el año 2005 la Universidad de las Ciencias Informáticas (UCI) comienza a utilizar Moodle. Éste, aunque cuenta con las ventajas que brindan los LMS, carece de la posibilidad de gestionar OA.

Simultáneamente, en la Facultad #10 un grupo de trabajo perteneciente al Polo "Teleformación", comienza a desarrollar un repositorio de objetos de aprendizaje (*ROA)*[12] con el objetivo de gestionar OA basados en el estándar SCORM, al igual que una herramienta de autor *(ROXS)*[13] para crear los mismos.

En (Tamayo, 2007) se plantea que dichas herramientas "*pueden ser consideradas una primera iteración del desarrollo de un framework para el e-learning, en el que éstas están relativamente aisladas y que la forma de comunicación entre ellas es a través de ficheros (paquetes de OA); la segunda iteración, en la que se trabaja actualmente, garantizará la interoperabilidad entre los diferentes componentes; y la próxima iteración estará basada en servicios"*. La segunda iteración, a pesar de estar basada en el estándar SQI, no aprovecha las ventajas de una arquitectura orientada a servicios. Por otro lado, la

[12] ROA es el nombre del repositorio.

[13] Nombre de la herramienta.

tendencia es implementar esta arquitectura en los sistemas que se utilizan en los procesos de la universidad. Por todo ello se plantea como ***problema investigativo*** ¿Cómo garantizar la integración de herramientas para la gestión de contenidos educativos en la Universidad de las Ciencias Informáticas (UCI)?

Como **objeto de investigación** la integración de herramientas para la gestión de contenidos educativos y como ***campo de acción*** *los* estándares educativos para la interoperabilidad entre aplicaciones e-learning.

El ***objetivo*** es proponer un framework basado en estándares de interoperabilidad que garantice la integración de herramientas para la gestión de contenidos educativos en la Universidad de las Ciencias Informáticas (UCI).

Como ***idea a defender*** la interoperabilidad entre diferentes herramientas que posibilitan la gestión de los contenidos educativos puede alcanzarse a través de estándares.

Para dar cumplimiento al objetivo se plantean las siguientes **tareas**:

- Analizar los proyectos más importantes relacionados con la interoperabilidad entre herramientas e-learning y las tendencias actuales para el desarrollo de las mismas.
- Analizar los estándares educativos que permiten la interoperabilidad de OA entre las aplicaciones e-Learning.
- Proponer un modelo de un framework que garantice la interoperabilidad entre herramientas para la gestión de contenidos educativos, utilizando los estándares educativos correspondientes.

Los métodos que se utilizan para el desarrollo de la investigación son el **Análisis - Síntesis** para el análisis de la documentación existente relacionada con el tema, extrayendo los elementos más importantes y necesarios para dar solución al problema existente, y el método **Análisis Histórico – Lógico** para hacer un estudio de cómo ha

evolucionado el e-learning y las tendencias actuales de la interoperabilidad entre herramientas que lo soportan.

Este trabajo consta de introducción, tres capítulos y conclusiones generales.

En el Capítulo 1, “Fundamentos Teóricos”, se abordan conceptos relacionados con el e-learning, descripciones sobre diferentes estándares educativos, haciendo énfasis en los de interoperabilidad, así como algunas experiencias de proyectos que han logrado la integración entre varios sistemas.

En el Capítulo 2, “Descripción de la solución propuesta”, se hace una descripción detallada de la solución que se propone que posibilita una mejor compresión. Además de la propuesta de algunas iteraciones futuras.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

1. Introducción

En el capítulo se abordan aspectos relacionados con e-learning, las herramientas que dan soporte a las actividades formativas, los LMS y LCMS, los OA, etc.

Por otro lado, se explican los estándares para el manejo de los contenidos educativos más utilizados en la actualidad como es el caso de SCORM y LOM, además de los estándares de interoperabilidad como son SQI, OKI, IMS-DRI entre otros, así como una breve descripción de los principales componentes de la arquitectura orientada a servicios y los servicios web.

Por último se describen algunas iniciativas de proyectos que han desarrollado la integración entre varios sistemas e-learning, que por su alcance sirven de referencia para este trabajo.

1.1 ¿Qué es e-learning? Características fundamentales.

El e-learning es la forma de educación a distancia surgida con el desarrollo de las nuevas Tecnologías de la Información e Internet, ofrece mayor flexibilidad respecto al método convencional de la clase en el aula, una vez ofertado un curso, cualquier persona pueden recibirlo sin importar el horario. El estudiante puede fijar sus propios ritmos de aprendizaje, según el tiempo de que disponga, se le hace fácil su acceso aunque para ello generalmente deba tener un terminal con conexión (Márquez, 2007). El e-learning permite reducir los costos pues suele ser más barato que la enseñanza tradicional, además que le brinda la posibilidad al profesor de tener un control exhaustivo del proceso de aprendizaje del estudiante.

En los sistemas de e-learning, se identifican por (García, 2002) tres elementos fundamentales. En primer lugar, una *plataforma* que da soporte a todas las actividades formativas y de gestión que tienen lugar durante el aprendizaje. En segundo lugar, los *contenidos* para el estudio o courseware[14]. Y por último, *herramientas de comunicación*, tanto síncrona como asíncrona, que permiten en contacto entre los participantes del curso.

Con el desarrollo de la web han surgido otros elementos con la intención de apoyar este tipo de enseñanza, tal es el caso de las *herramientas de trabajo colaborativo* que tienen como objetivo compartir la información y el conocimiento, el *software social* que facilita la interacción y colaboración entre comunidades en red, se puede mencionar como ejemplo de ellos, los correos electrónicos, la mensajería instantánea, wikis, finalmente la *web 2.0* que incluye algunos de los elementos antes mencionados y además describe la web como una convergencia de servicios dentro de una arquitectura SOA.

Todos estos elementos se han ido integrando formando diferentes herramientas con un mismo objetivo; apoyar el proceso de enseñanza aprendizaje de los estudiantes. En algunos de los casos se ha logrado una combinación de la mayoría de ellos, formando potentes herramientas.

1.1.1 Herramientas para el e-learning.

Como se dijo anteriormente, uno de los elementos que forma parte del e-learning son las plataformas que sirven de apoyo a este tipo de enseñanza. Las mismas son herramientas que se utilizan para la creación, gestión y distribución de cursos a través de la Web. Son potentes instrumentos que permiten diseñar, y elaborar todos los recursos necesarios para cursar, gestionar, administrar y evaluar las actividades educativas. Entre las principales características se encuentra; el uso masivo de los medios electrónicos para

[14] Cualquier programa de software de tipo instruccional o educacional.

trasmitir conocimientos, permitiendo así capacitar a más en menos tiempo en el momento y lugar donde se necesite, siempre que estén creadas las condiciones, agilizando así el proceso de enseñanza-aprendizaje.

De ahí que la inmensa mayoría de las instituciones y universidades hayan comenzado a utilizar los LMS y LCMS no como sustituto de la formación presencial tradicional, sino como un complemento más. Aunque se puede mencionar las Universidades Virtuales como un ejemplo de virtualidad a tiempo completo, y que en su mayoría tiene como uno de sus componentes a un LMS o LCMS.

1.1.1.1 ¿Qué es un LMS?

Learning Management System (LMS) son aplicaciones que facilitan la creación de entornos de enseñanza-aprendizaje, integrando materiales didácticos y herramientas de comunicación, colaboración y gestión educativa. Ofrecen un ambiente de aprendizaje donde cada alumno puede acceder a través de una clave personal, permitiéndole al profesor tener un seguimiento del progreso del estudiante, siendo éste su principal objetivo.

Se puede decir los LMS, deben contar como mínimo con las siguientes características (Lozano, 2008):

- *Perfiles de acceso* para ello se definen varios roles (alumno, tutor, profesor, coordinador, administrador), cada uno con diferentes privilegios.
- Las *herramientas de comunicación* tanto síncronas como asíncronas aspecto fundamental para el aprendizaje y para las relaciones sociales.
- Los *servicios y áreas configurables* es importante que dependiendo del curso tanto los servicios como las áreas sean configurables.
- La *gestión académica y administrativa* que permite tener un sistema de gestión de expedientes administrativos, expedientes académicos, control de perfiles de usuarios, administración de cursos, etc.

- *Los sistemas de gestión de calificaciones* que permiten recoger tanto los resultados obtenidos en pruebas objetivas, como las notas insertadas por los profesores para calificar otras actividades evaluables.

Los LMS presentan carencia en cuanto a la gestión de los contenidos, puesto que sólo actúan como plataformas de distribución, como se dijo anteriormente. Son varios los LMS existentes, tal es el caso de ATutor, Claroline, Dokeos, Sakai, Moodle etc, siendo estos dos últimos los más difundidos a nivel mundial, pero no son la única herramienta de gestión educativa. Existen otras opciones, en las que lo importante no es tanto la gestión de los alumnos, sino los contenidos, tal es el caso de los LCMS.

1.1.1.2 ¿Qué es un LCMS?

Learning Content Management System (LCMS) tienen su origen en los sistemas de gestión de contenidos (CMS) cuyo objetivo es simplificar la creación y la administración de los contenidos en línea. Los LCMS siguen el concepto básico de los CMS, que es la administración de contenidos, pero enfocados al ámbito educativo, administrando y concentrándose únicamente en recursos educativos y no en todo tipo de información.

Por otro lado, una plataforma LCMS además de garantizar el control del proceso de aprendizaje, debe facilitar la creación, almacenamiento y reparto de los contenidos. Para ello a diferencia de los LMS, cuenta con un *repositorio de objetos de aprendizaje* que su función principal es coleccionar recursos digitales, a manera de bases de datos, tanto los contenidos digitales, objetos de información y aprendizaje que conforman las lecciones, como unidades didácticas y cursos generados, brindando la posibilidad que los puedan consultar y reutilizar los distintos usuarios, sin dañar la integridad de la información; y además una *herramienta de autor* la cual está enfocada a crear OA que serán a su vez, almacenados en el repositorio. Estas deberán de considerar los estándares como es el caso de SCORM.

En un LCMS la pieza de información autocontenida más pequeña es un OA, por lo que la reutilización es posible.

1.1.2 Objetos de Aprendizaje

Una búsqueda en Internet de referencias asociadas con “learning objects” puede tener por resultado más de 90 millones. En cambio, si se introduce “objetos de aprendizaje” el número de citas baja alrededor de 2 millones. Las diferencias numéricas, son enormes, pero no son las únicas. Cuando se consultan algunas de estas bibliografías aumentan considerablemente las discrepancias entre qué es un OA, puesto que no existe una definición universal del mismo. Como se dijo anteriormente este trabajo se apoyará en *“cualquier recurso con una intención formativa, compuesto de uno o varios elementos digitales, descrito con metadatos, que pueda ser utilizado y reutilizado dentro de un entorno e-Learning”* (López, 2005).

Dentro de las principales características que presentan los OA se encuentran (García, 2005):

- Reutilización, objeto con capacidad para ser usado en contextos y propósitos educativos diferentes y para adaptarse y combinarse dentro de nuevas secuencias formativas.
- Interoperabilidad, capacidad para poder integrarse en estructuras y sistemas (plataformas) diferentes.
- Accesibilidad, facilidad para ser identificados, buscados y encontrados gracias al correspondiente etiquetado a través de metadatos que permiten la catalogación y almacenamiento correspondiente en repositorios.
- Durabilidad, vigencia de la información de los objetos, sin necesidad de nuevos diseños.

Para su real utilidad se requiere que estos OA sean compatibles con diversos ambientes y sistemas de administración de aprendizajes, fáciles de migrar de una plataforma a otra,

fáciles de localizar, acceder, archivar y reutilizar. Es por ello que surgen los Repositorios de Objetos de Aprendizaje (ROA).

Los ROAs permiten almacenar, buscar, recuperar y consultar OA. Se identifican (Downes, 2001) dos tipos: “los que contienen OA y sus metadatos, en éstos los objetos y sus descriptores se encuentran dentro de un mismo sistema e incluso dentro de un mismo servidor; los que contienen sólo los metadatos, en este caso el repositorio contiene sólo los descriptores y se accede al objeto a través de una referencia a su ubicación física que se encuentra en otro sistema o repositorio de objetos”.

Regularmente los ROAs operan de forma independiente, aunque lo ideal sería que los LMS se complementen con algún repositorio. Además, como se dijo anteriormente, un repositorio es uno de los componentes que se identifican dentro de un LCMS. En los últimos años la tendencia es que los repositorios se puedan comunicar entre si y a la vez con otros sistemas e-learning permitiendo así búsquedas federadas y la reutilización de los contenidos.

Para que los OA sean estandarizados y de esa forma poder almacenarlos en los repositorios son necesarias las Herramientas de Autor, las mismas permiten empaquetar dichos OA rigiéndose por estándares como SCORM e IMS Content Package. Se puede decir que existen pocas herramientas que implementen este estándar, aunque se pueden mencionar Reload, Exelearning, y Lompad como las más utilizadas actualmente.

El principal objetivo de estandarizar los OA es que estos tengan un lenguaje común para que sean entendidos por diferentes herramientas, permitiendo así la interoperabilidad entre sistemas.

1.2 Estándares para el desarrollo del e-learning.

Se pueden encontrar dos tipo de estándares; estándares de jure, cuando provienen de una organización acreditada que certifica una especificación, y estándares de facto, cuando la especificación se adopta por un grupo mayoritario de individuos.

La estandarización se requiere a distintos niveles, primero, cuando los recursos son creados deben considerarse tecnologías, políticas y formatos compatibles, éstos pueden ser de una institución en específico; segundo, cuando esos recursos son incluidos en un repositorio, y deben ser descritos, se utilizarán esquemas que aseguren su fácil localización y compatibilidad con otros sistemas, siendo posible a través de los metadatos; tercero, cuando esos recursos sean utilizados y tengan que incorporarse a diferentes servicios, repositorios, plataformas y aplicaciones son empaquetados; y cuarto, cuando los sistemas involucrados en un entorno tengan que interoperar con otros para cumplir sus funciones o ampliar sus capacidades.

La creación de estándares es una tarea compleja, sin embargo, diferentes grupos e instituciones trabajan en el desarrollo de especificaciones y de estándares en los distintos niveles requeridos, para establecer entornos e-learning integrados. Las que tienen más propuestas son:

- AICC (Aviation Industry Computer-Based Training Comitee). Es una asociación de entrenamiento profesional basado en tecnología. Se reconoce como una de las precursoras de la estandarización de materiales para entrenamiento profesional.(http://www.aicc.org/)
- ADL (Advanced Distributed Learning). La misión de ADL es la de proveer acceso de la más alta calidad en educación y entrenamiento, en cualquier lugar y momento, dentro de sus propuestas se encuentra SCORM (Shareable Content Object Referente Model) como la más relevante. (http://www.adlnet.org/)
- ARIADNE (Alliance of Remote Instructional Authoring and Distribution Networks for Europe). Patrocinado por la Unión Europea, este proyecto está enfocado al

desarrollo de herramientas y metodologías para producir, administrar y reutilizar elementos pedagógicos basados en computadora. (http://www.ariadne-eu.org)

- W3C (World Wide Web Consortium). Se encarga del desarrollo de tecnologías interoperables (especificaciones, normas, software y herramientas) para aprovechar todo el potencial del web. Aunque no está directamente vinculado con el desarrollo del e-learning es importante mencionarlo, ya que de la interoperabilidad de la web dependen muchas de las funciones de la educación en línea. (http://www.w3.org)
- MIT (Instituto Tecnológico de Massachusetts) es una de las principales instituciones dedicadas a la docencia y a la investigación en Estados Unidos.
- IMS (IMS Global Learning Consortium) cuenta con miembros de organizaciones comerciales, educativas y gubernamentales dedicadas a definir y distribuir arquitecturas abiertas para actividades de educación en línea. Uno de sus resultados es lo que se conoce como el estándar IMS. http://www.imsproject.org/
- Prolearn es una red de excelencia financiado por IST (Information Society Technology) programa de la comisión europea que tiene como misión reunir a los más importantes grupos de investigación en el área de aprendizaje profesional y la formación, así como otras organizaciones clave y socios industriales.

De estos grupos de desarrollo, las propuestas más ampliamente adoptadas, en gran número de herramientas, han sido las de IMS Learning Consortium y la de ADL, que han tenido como resultado las especificaciones IMS y el modelo SCORM, respectivamente. Otro resultado significativo es el MIT con su propuesta del estándar OKI junto a las OSID que define el mismo para garantizar interoperabilidad entre diferentes sistemas, y prolearn con el estándar SQI.

1.2.1 Estándares para la interoperabilidad de contenidos

1.2.1.1 Shareable Content Object Reference Model (SCORM).

SCORM, define un modelo software que describe el modelo de agregación de contenidos, las interrelaciones establecidas entre las componentes de los cursos, los modelos de datos y los protocolos de comunicación, de manera que los “objetos” definidos en un LMS puedan compartirse entre diferentes LMS.

Está formado por un conjunto de tres libros técnicos que recogen los diversos aspectos de la especificación, más un cuarto libro que ofrece una visión general sobre el estándar. Éstos son: Overview, Content Aggregation Model (CAM), Run-Time Environment (RTE), Sequence Information & Navigation (SN). (Fernández. B, Moreno. B, Sierra. J, et all, 2007).

El Overview contiene una introducción a los elementos de SCORM y su terminología describe brevemente las áreas de los otros tres libros (CAM, RTE y SN), mostrando cómo se relacionan una con otras.

El Content Aggregation Model (CAM) contiene información sobre la confección, etiquetado y empaquetamiento de los contenidos de aprendizaje. Un paquete de contenidos agrupa una serie de ficheros físicos (o URL) con el contenido real del paquete, además de contar con una organización que se describe en un manifiesto nombrado “imsmanifest.xml". El manifiesto y los ficheros son agrupados en un archivo comprimido en formato .ZIP definido por SCORM como PIF (Package Interchange File), siendo este el paquete que transita por las diferentes plataformas de formación. El manifiesto cuenta con la información necesaria para describir el paquete, está formado por cuatro secciones, como se muestra en la Figura 1.

- Metadatos: información que describe el paquete como un todo, en esta sección utiliza los metadatos de LOM[15] definidos por la IEEE.
- Organización: representa la organización de contenidos y su descomposición en actividades (item). Cada actividad está enlazada con los recursos que utiliza, que se encuentran en la siguiente sección, a través de su identificador (identifier).
- Recursos: describen los recursos externos (a través de URL) y locales que utiliza el paquete. Los recursos locales se encontrarán comprimidos en el mismo PIF.
- SubManifiestos: Los recursos complejos suelen estar formados por una jerarquía de entidades, cada una de las cuales tiene su propio manifiesto (cursos, lecciones, módulos...).

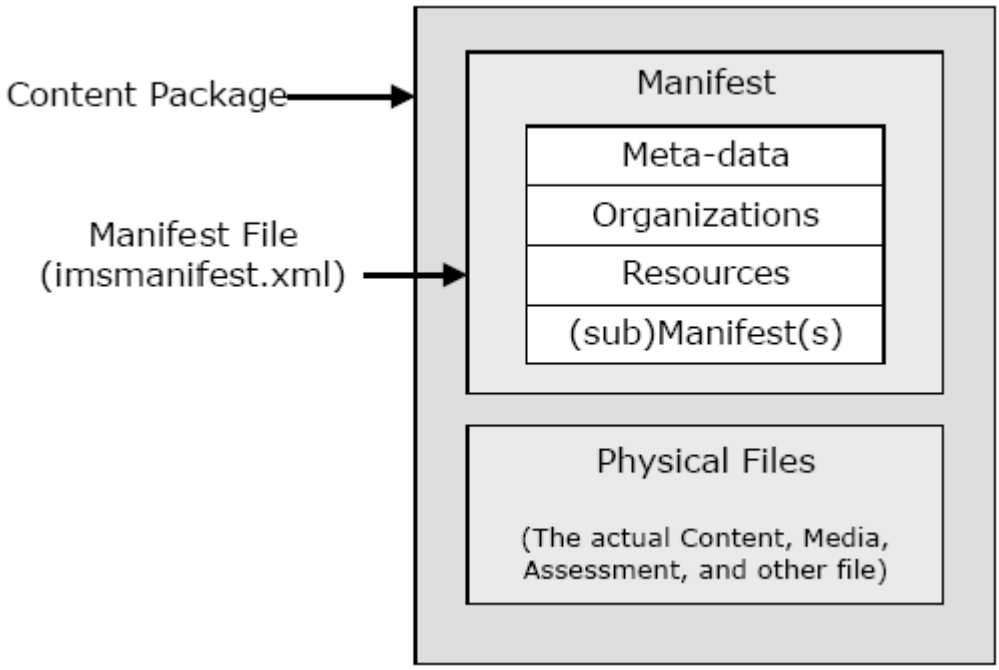


Figura 1. Modelo de empaquetado

El Run-Time Environment (RTE) requerimientos para la ejecución de los contenidos, comunicación, seguimiento, transferencia de datos y gestión de errores.

[15] Será explicado en la sección siguiente.

Este libro cubre tres aspectos:

- Gestión del Entorno de Ejecución: normas que indican el proceso del lanzamiento de los objetos de contenido y la gestión de la comunicación entre la plataforma y los SCO (Sharable Content Objects).
- API (Application Program Interface). Es el mecanismo para informar al LMS del estado del contenido (por ejemplo si está inicializado, finalizado o en error) y es usado para intercambiar datos entre el LMS y los SCO (por ejemplo datos de tiempo, de puntuación, etc.). La API es simplemente un conjunto de funciones predefinidas que se ponen a disposición de los SCO.
- Modelo de datos del RTE: Describe exhaustivamente todos los datos y su modelo de comportamiento para la construcción del RTE y para la gestión de los SCO por parte de la plataforma.

Y por último Sequence Information & Navigation (SN) que permite la descripción de las reglas que marcan los secuencia de navegación entre distintos OA, este último libro surgió con la versión 2004 de SCORM.

SCORM propone básicamente cuatro cualidades necesarias para lograr OA reutilizables efectivamente. (Fernández. B, Moreno. B, Sierra. J, et all, 2007).

- Reutilización: el contenido debe ser independiente del contexto de aprendizaje, apto para su uso en diferentes situaciones, para diferentes públicos, sobre diferentes plataformas de entrega con diferentes herramientas de desarrollo.
- Interoperabilidad: el contenido debe funcionar en múltiples programas de aplicación, ambientes, hardware y software, independientemente de las herramientas usadas para crearlo o entregarlo.
- Durabilidad: el contenido debe continuar operando sin requerir modificaciones ante cambios o actualizaciones en el hardware o el software del sistema.
- Accesibilidad: el contenido debe ser identificable y ubicable cuando se lo necesite, para los requerimientos formativos necesarios. Debe poder conocerse su

adecuación a los objetivos sin necesidad de obtener el propio contenido o pagar derechos por él, mediante la provisión de información suficiente acerca de cada OA.

1.2.1.2 Learning Object Meta-Data (LOM).

LOM ha sido adoptado en la especificación de IMS Learning Resorce Metadata, su objetivo es la creación de descripciones estructuradas de recursos educativos. Su modelo de datos especifica qué aspectos de un OA deben ser descritos y qué vocabularios se pueden utilizar en dicha descripción. Cuenta con nueve categorías ellas son:

a) La categoría General agrupa la información general que describe un OA.
b) La categoría Ciclo de Vida agrupa las características relacionadas con la historia y el estado actual del OA, así como las que han afectado durante su evolución.
c) La categoría Meta-Metadatos agrupa la información sobre la propia instancia de Metadatos.
d) La categoría Técnica agrupa los requerimientos y características técnicas del OA.
e) La categoría Uso Educativo agrupa las características educativas y pedagógicas del OA.
f) La categoría Derechos agrupa los derechos de propiedad intelectual y las condiciones para el uso del OA.
g) La categoría Relación agrupa las características que definen la relación entre este OA y otros OA relacionados.
h) La categoría Anotación permite incluir comentarios sobre el uso educativo del OA e información sobre cuándo y por quién fueron creados dichos comentarios.
i) La categoría Clasificación describe este OA en relación a un determinado sistema de clasificación.

Dentro de los metadatos obligatorios se encuentra de la categoría general:

- Identifier (identificador): Identificador descriptivo del material educativo. Su valor debe identificar unívocamente el material en su contexto educativo.
- Title (título): Nombre descriptivo del material educativo.
- Description (descripción): Texto describiendo el contenido del material.
- Keyword (palabra clave): Colección de frases que representan palabras clave sobre el material.

De la categoría lifecycle (ciclo de vida):

- Version (versión): La edición o versión del material
- Status (El estado de producción del material): LOM propone el siguiente vocabulario para este metadato (aunque puede utilizarse cualquier otro): draft (borrador), final, revised (revisado), unavailable (no disponible).

De la Categoría metametadata (meta-metadatos):

- Metadata Scheme (esquema de metadatos): Esquema de metadatos utilizado (por ejemplo, LOMv1.0). Obsérvese que es posible haber usado más de un esquema (por ejemplo, en el caso de que se haya realizado una especialización o perfil de aplicación de uno ya existente).

De la Categoría technical (técnica):

- Format (formato): Formato del material. Dado que el material no tiene porque ser atómico, es posible que integre múltiples formatos (por ejemplo, una página web puede integrar un documento HTML con un conjunto de imágenes JPG), por lo que un mismo material puede exhibir múltiples metadatos format.
- Location(Localización): Una cadena utilizada para acceder a este objeto educativo. Puede ser un localizador (por ejemplo, un Localizador Universal de Recursos,

URL), o un mecanismo que finalmente permite acceder a una localización (por ejemplo, un Identificador Universal de Recursos, URI).

De la categoría Educational (Uso educativo):

- Tipo de recurso (learningresourcetype): El tipo específico de recurso educativo (ejercicio examen). El tipo predominante debe aparecer en primer lugar. Existe un vocabulario recomendado por el estándar.
- Interactivitylevel(nivel de Interectividad): El grado de interactividad que caracteriza a este objeto educativo. La interactividad en este contexto se refiere al grado en el que el aprendiz puede influir en el aspecto o comportamiento del objeto educativo. Existe un vocabulario recomendado por el estándar.

De la categoría rights (derechos):

- Cost (coste): Establece si el recurso es o no de pago. LOM propone como vocabulario controlado para este metadato el siguiente: *yes, no*.
- Copyright and other Restrictions (derechos de copia y otras restricciones): Establece si el recurso está o no sujeto a derechos de copia y otras restricciones. LOM propone como vocabulario controlado para este metadato, *de nuevo*, *yes* y *no.*

A pesar que estos estándares permiten la interoperabilidad de contenidos entre diferentes plataformas, no garantiza la comunicación entre ellas. Con este fin fueron creados algunos estándares de interoperabilidad.

1.2.2 Estándares para la Interoperabilidad entre aplicaciones.

Según (Aguirre .S, Salvachúa .J, Quemada J, et all, 2007) “la mayoría de los recursos educativos en formato digital que se encuentran almacenados dentro de sistemas e-learning o sistemas de gestión de contenidos educativos son cerrados y propietarios

dificultando el acceso a ello". Por otro lado, muchos de estos sistemas la arquitectura con que fueron diseñados no les permite comunicarse con otras herramientas, dificultando así el intercambio de información de unas a otras.

Esto ha propiciado que hayan surgido varias propuestas para integrar herramientas e-learning, con el objetivo de aprovechar al máximo los contenidos una vez que son creados, posibilitar la reutilización de los mismos y la interoperabilidad entre sistemas heterogéneos. Estas propuestas se han basado principalmente en utilizar diferentes estándares.

1.2.2.1 ANSI/NISO Z39.50.

Este estándar está diseñado para la búsqueda y recuperación de información en entornos abiertos. Conocido formalmente como ANSI/NISO Z39.50 permite comunicar sistemas que funcionan en distintos hardware y usan diferente software. Fue diseñado para solucionar los problemas asociados con las búsquedas a múltiples base de datos. Utilizado para bibliotecas digitales, ofrece a los usuarios una interfaz de acceso única, que les permite tanto la búsqueda en el sistema local como en cualquier sistema remoto que soporte el estándar. (López, 2007)

Z39.50 sigue las directrices del modelo cliente/servidor para el desarrollo de aplicaciones distribuidas. Donde la parte cliente se llama "origen" y la parte servidora "destino". Cuando un usuario hace una petición en su sistema local, con la interfaz que éste le ofrece, el módulo origen Z39.50 que reside en su sistema traduce esta petición a un formato estandarizado y la emite a los sistemas remotos que convenga, siempre que disponga el mismo de un servidor Z30.50. Este servidor realiza la búsqueda en el sistema remoto, traduciendo de nuevo la petición al lenguaje de comandos o procedimientos que use el servidor, y devuelve el resultado al módulo origen, como se muestra en la Figura 2.

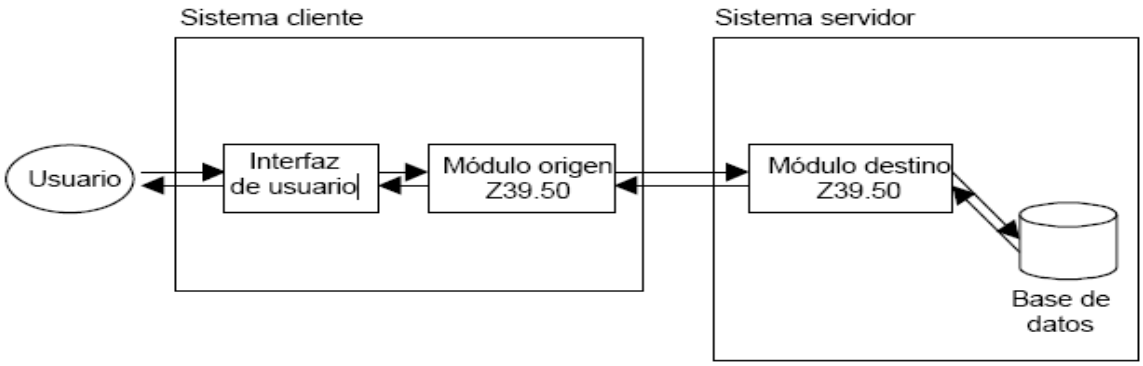


Figura 2 Diagrama de bloques de un sistema Z39.50.

Cuenta con tres tipos de servicios *confirmados* cuando tras una petición tiene una respuesta, *no confirmados* implica una petición sin respuesta, y *condicionalmente confirmado* cuando un servicio puede ser invocado tanto como confirmado como no confirmado. (López, 2007)

Para lograr el intercambio de mensajes entre cliente y servidor, se realizan 11 servicios que permiten la transferencia de información, estos son: initialization, search, retrieval, result-set-delete, access control, accounting/resource control, sort, browse, explain, extended services, termination.

Dicho estándar nació en las bibliotecas de los Estados Unidos, y es ahí y en Europa donde más se ha visto su empleo. Una de las grandes desventajas que tiene es su complejidad a la hora de su implementación, es por ello que muchas instituciones se miden al escogerlo, genera un flujo de intercambio de datos por las redes en ocasiones innecesarios produciendo demora y problema de tráfico.

1.2.2.2 The Open Archives Initiative Protocol for Metadata Harvesting OAI-PMH.

La Open Archives Initiative (OAI) se creó con la misión de desarrollar y promover estándares de interoperabilidad para facilitar la difusión eficiente de contenidos en Internet. Surgió como un esfuerzo para mejorar el acceso a archivos de publicaciones

electrónicas (eprints), en definitiva, para incrementar la disponibilidad de las publicaciones científicas. OAI no es solamente un proyecto centrado en publicaciones científicas, sino también en la comunicación de metadatos sobre cualquier material almacenado en soporte electrónico.

OAI-PMH utiliza transacciones HTTP para emitir preguntas y obtener respuestas entre un servidor o archivo y un cliente o servicio recolector de metadatos. El segundo puede pedir al primero que le envíe metadatos según determinados criterios como por ejemplo la fecha de creación de los datos. En respuesta, el primero devuelve un conjunto de registros en formato XML, incluyendo identificadores (URL por ejemplo) de los objetos descritos en cada registro.

Las peticiones se emiten utilizando los métodos GET o POST del protocolo HTTP y constan de una lista de opciones con la forma de pares del tipo: clave=valor. Existen seis peticiones que un cliente puede realizar a un servidor: *getrecord* es utilizado para recuperar un registro concreto, *Identify* para recuperar información sobre el servidor, *listIdentifiers* recupera los encabezamientos de los registros, en lugar de los registros completos, *listRecords* recupera los registros completos, *listSets* recupera un conjunto de registros. Estos conjuntos son creados opcionalmente por el servidor para facilitar una recuperación selectiva de los registros y por último *listMetadataFormats* el cual devuelve la lista de formatos bibliográficos que utiliza el servidor.

El protocolo soporta múltiples formatos para expresar los metadatos, no obstante requiere que todos los servidores ofrezcan los registros utilizando Dublin Core[16] no calificado, codificado en XML.

[16] Estándar de interoperabilidad de contenidos.

1.2.2.3 IMS Distributed Repository Interoperability (IMS-DRI).

IMS DRI su principal objetivo es facilitar el acceso a los contenidos en los repositorios para contextos educativos, con los LMS y los LCMS, pero también con otros sistemas como los portales de búsquedas. Debido a que el intercambio de información entre repositorios es muy poco probable que se realice de una misma forma, es muy difícil dar una solución exacta de cómo realizarlo, es por ello que este estándar sólo entrega algunas recomendaciones[17] para garantizar la interoperabilidad de las funciones más comunes de un repositorio, estas funciones son: buscar, exponer, colectar, enviar, almacenar, pedir, entregar y alertar. Las recomendaciones deben ser implementadas a través de servicios que establezcan una interfaz común entre ellos. (Ver tabla 1)

En un sentido amplio, la especificación define a los repositorios digitales como una colección de recursos que son accesibles vía red, sin ningún conocimiento de su estructura arquitectónica. Los repositorios pueden manejar recursos o metadatos que describen esos recursos, los cuales pueden o no estar en un mismo repositorio albergado.

Función	Descripción	Recomendación tecnológica
Buscar/Exponer (Search/Expose)	Ejecuta la búsqueda de metadatos asociados con los contenidos que el repositorio expone.	XQuery para colecciones con metadatos en XML y Z39.50 para búsquedas en sistemas bibliotecarios.
Colectar/Exponer (Gather/Expose)	Define la solicitud de metadatos que el repositorio expone, la agregación de los metadatos para	No hay recomendación específica, pero IMS DRI sugiere que OAI puede proveer una funcionalidad adecuada.

[17] En algunas bibliografías lo tratan como especificaciones.

	utilizarse en búsquedas subsecuentes y la agregación de metadatos para crear nuevos repositorios.	
Enviar/Almacenar (Submit/Store)	Se enfoca a la manera en la que un objeto se mueve a un repositorio desde un sitio accesible por red y cómo el objeto será representado en el repositorio para su acceso.	Se recomienda el uso de paquetes IMS a través de SOAP.
Pedir/Entregar (Solicitar/Entregar) (Request/Deliver)	Permite, que una vez que el usuario a localizado los metadatos en la función Buscar, pueda solicitar al repositorio el acceso al recurso.	Recomendación general para utilizar HTTP, y FTP, para diferentes tipos de recursos. También IMS CP.
Alertar/Exponer (Alert/Expose)	Función clave, en la que a través de correo electrónico se notifica a los usuarios sobre eventos en el repositorio, pero no está contemplada todavía en esta versión de la especificación.	No hay recomendación específica ya que esta función sale del alcance de esta especificación.

Tabla 1 Descripción y recomendaciones tecnológicas para las funciones de un repositorio

Este estándar será uno de los utilizados en la solución de este trabajo, debido a que define cuáles son los funciones necesarias para que los repositorios se comuniquen entre

sí, aunque no serán utilizados los estándares que propone para cada función, sino otras propuestas que han surgido tras esta investigación.

1.2.2.4 Simple Query Interface (SQI).

Es una especificación que pretende ser una capa que garantice la interoperabilidad entre redes o entornos educacionales heterogéneos. El objetivo es ser una parte del sistema que sea capaz de buscar en los distintos repositorios (heterogéneos) de OA existentes en las redes a pesar de que posean interfaces propietarias de búsqueda de cada fabricante.

Para permitir la interoperabilidad entre repositorios digitales heterogéneos, es necesario tener en cuenta ciertos aspectos. Por un lado, se necesita un modelo semántico común, el cual especifique el formato de las distintas propiedades de los objetos contenidos en los repositorios. Por otro lado, la interoperabilidad está basada en protocolos comunes, los cuales definen las interacciones posibles entre los repositorios.

SQI es neutral en términos de lenguaje de consultas, además es un Application Program Interfaces (API) para establecimiento de sesión y realización de consultas síncronas y asíncronas que define los servicios que un repositorio puede tener disponibles para recibir y responder consultas de otros repositorios, es decir, SQI es parte de la arquitectura para la interoperabilidad de repositorios educativos. Los servicios básicos son: servicios de autenticación, gestión de la sesión y servicios de aplicación como gestión de las consultas.

SQI define un conjunto de métodos agrupados por categorías como sigue a continuación[18]:

[18] Se mantendrán los nombres de las categorías y métodos en inglés para no alterar su significado.

- The query management: Métodos que permiten la configuración de los parámetros de consultas como son: setQueryLanguage(), setResultsFormat(), setMaxQueryResults(), setMaxDuration().
- In a synchronous query: Los resultados de las consultas son directamente retornados por el método (synchronousQuery). Métodos adicionales permiten la elección de los números de resultados retornados por una consulta (setResultsSetSize) y además permiten conocer el número total de resultados de una consulta (getTotalResultsCount).
- In an asynchronous query: Los resultados de consultas son enviados por el destino[19] hacia la fuente[20], este último debe tener implementado un listener[21] (queryResultsListener). Esto implica que la fuente ha de indicar la localización de su listener (set-SourceLocation) antes de enviar una consulta asíncrona.

En la Figura 3 se muestra un proceso a grandes rasgos, donde un repositorio A (fuente) envía una consulta a un repositorio B (destino).

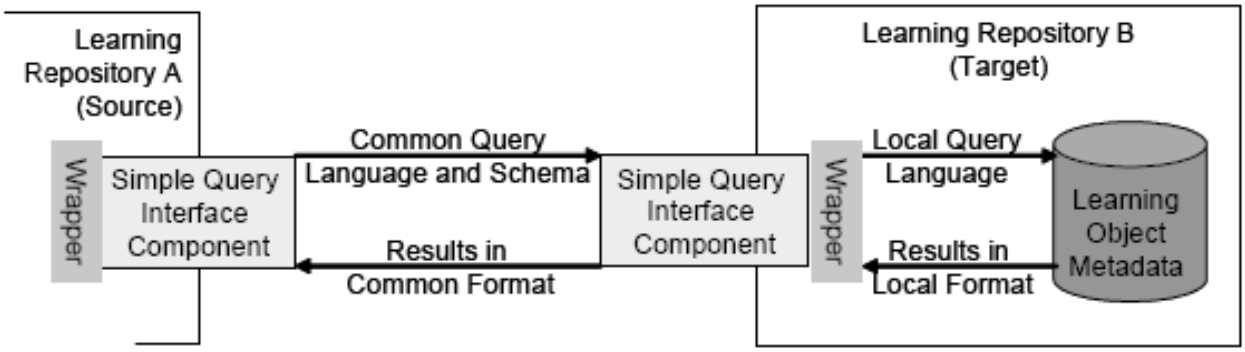


Figura 3 Conexión entre repositorios a través de SQI.

[19] Se considera al repositorio al cual se le hizo la consulta.

[20] Se considera al repositorio al cual hizo la consulta.

[21] Método implementado en la fuente para escuchar las respuestas que son emitidas por el destino e una consulta de modo asíncrono.

Como se describió en el acápite anterior, IMS DRI define algunas propuestas de estándares a utilizar como son el Z39.50 y OAI respectivamente, sin embargo este trabajo se basará en SQI ya que es el estándar más utilizado actualmente en la comunicación entre repositorios, debido a todas las ventajas que permite el mismo y que fueron explicadas en este acápite.

1.2.2.5 Open Knowledge Initiative (OKI).

Es una iniciativa que surge de la colaboración del MIT con otras universidades y organismos de estandarización y propone una arquitectura abierta y extensible que especifica cómo los componentes de un entorno de software educativo pueden comunicarse entre sí y con otras plataformas.

OKI[22] provee especificaciones detalladas para interfaces de los componentes que gestionan la administración de un entorno de aprendizaje y ejemplos de cómo estas interfaces trabajan. Proporciona una base estable y escalable que apoya la flexibilidad para que proyectos e-learning estén cada vez más integrados al proceso educativo.

Es un estándar de interoperabilidad que detalla cómo utilizar los servicios que define y con qué restricciones, es además una propuesta concreta de desarrollo, promueve especificaciones que describen cómo los componentes de un entorno de software se comuniquen entre sí y con otros sistemas. Al definir con claridad los puntos de interoperabilidad, la arquitectura permite que los componentes de entorno de aprendizaje sean desarrollados y actualizados de forma independientes unos de otros. Esto lleva a una serie de ventajas: (Thorne & Kahn, 2006)

- Las tecnologías de aprendizaje y los contenidos puede ser fácilmente compartida entre centros educacionales.

[22] www.okiproject.org.

- Hay un menor costo a largo plazo de la propiedad del software porque los componentes pueden ser reemplazados y actualizados sin necesidad de que todos los demás componentes sean modificados.
- La modularidad hace que las tecnologías de aprendizaje sean más estables, fiables, y capaces de crecer con el aumento de su uso y permite que los componentes sean actualizados sin desestabilizar otras partes del ambiente.
- La arquitectura ofrece una base estándar para el desarrollo de software de tecnologías de aprendizaje. Esto reduce el esfuerzo de desarrollo y alienta el desarrollo de componentes especializados que se integren en sistemas más grandes.

OKI proporciona un conjunto de interfaces para la definición de servicios denominados OSID's (Open Service Interface Definitions) y proporciona una implementación de referencia para cada uno de ellos.

1.2.2.5.1 Arquitectura Open Service Interface Definitions (OSID) de OKI.

Desde la perspectiva tecnológica la mayor parte de los sistemas de enseñanza virtual actuales se basa en una arquitectura constituida por un conjunto de módulos o herramientas construidos sobre un sistema central de servicios comunes que a su vez comunica con un soporte persistente o repositorio de datos, generalmente constituido por una base de datos con lenguaje SQL. La comunicación entre el sistema central y cada módulo se lleva a cabo únicamente mediante llamadas basadas en las especificaciones OKI OSIDs. Cada módulo es responsable de gestionar tanto la funcionalidad como la información específica de este.

Los módulos se apoyan sobre una capa OSID, (vea Figura 4) un conjunto de especificaciones de interfaces desarrolladas por el proyecto OKI que definen una serie de servicios de bajo y medio nivel presentes en la mayor parte de las aplicaciones que se utilizan en instituciones educativas y que constituyen un nivel de abstracción entre la infraestructura existente y las aplicaciones que se construyan sobre tales interfaces.

Aunque escritas pensando en su uso con aplicaciones Java, las OSID son neutras en cuanto al lenguaje de implementación por lo que se pueden trasladar a otros lenguajes o incluso sería factible implantarlas en forma de servicio web si el rendimiento lo permite. Hay, no obstante, otro proyecto de *framework* similar aunque orientado explícitamente a servicios web, el *JISC E-Learning Framework Programme*. (Alfonso, 2007)

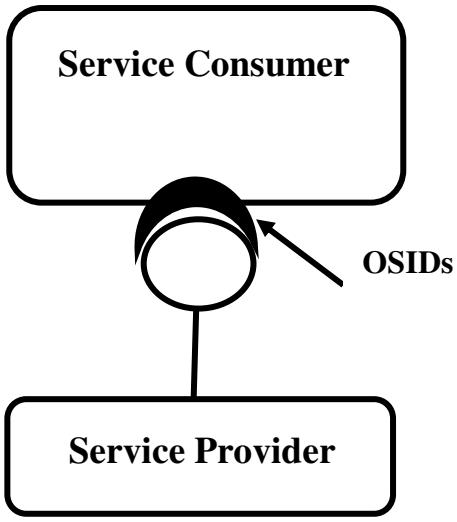


Figura 4. Especificaciones OSID de OKI.

Las OSID se agrupan en managers según el servicio al que atiendan; por ejemplo AuthenticationManager o AutorizationManager entre las de nivel medio, o SharedManager para los servicios compartidos de bajo nivel como Type, Iterator y otros.

Utilizar una interfaz similar tiene varias ventajas. En primer lugar, su modularidad permite mantener y resolver por separado cada conjunto de tareas sin entorpecer al resto de la aplicación. Además, las OSID no hacen asunciones acerca de los detalles de la implantación; tan sólo proporcionan un marco coherente en el que basar el desarrollo. No hay obligación alguna de tener implementaciones de todas las interfaces antes de empezar el desarrollo de la aplicación y tampoco las interfaces utilizadas tienen por qué incluir todos los métodos definidos, por lo que se puede aplicar una metodología de desarrollo incremental.

La principal ventaja de utilizar OSID radica en disponer de un modelo de referencia lo bastante preciso como para constituir un marco de desarrollo práctico y efectivo, y lo bastante abierto como para servir de base a las múltiples aplicaciones que se utilizan en las instituciones académicas. Las mismas detallan especificaciones para las interfaces de una infraestructura de servicios comunes. Provee una base estable y escalable que soporta la flexibilidad necesitada para incrementar los procesos de integración.

Todo sistema que va a tener como base el estándar OKI, debe implementar como mínimo las OSID de autenticación (osid_authentication) y autorización (osid_authorization). A pesar de que en muchos casos se confunde estos dos términos e incluso se llega a decir que se pueden ver juntos, tienen funcionalidades muy diferentes. La autenticación es el proceso de verificar la identidad de una persona, mientras la autorización es el proceso de verificación que una persona conocida tiene la autoridad para realizar una cierta operación. La autenticación, por lo tanto, debe preceder la autorización. ¿En qué consisten estas dos OSID?[23]

- osid_authentication: Soporta invocar procesos de autenticación. La implementación de estos servicios son responsables de garantizar cualquier información apropiada para llevar a cabo la autenticación. Los servicios que maneja esta OSID permiten probar si un usuario es autenticado, para ello devuelve el identificador que corresponde a cada usuario, los tipos de autenticación posible y además destruyen todas las autenticaciones o solo aquellas de un tipo dado. (Ver Tabla 2)

[23] En el capítulo 2 se hará referencia a todos los métodos de cada una de las OSID en mayor profundidad.

Valor de Retorno	Métodos
Void	authenticateUser(Type authenticationType)
Void	destroyAuthentication()
Void	destroyAuthenticationForType(Type authenticationType)
object &	getAuthenticationTypes()
object &	getUserId()
boolean	isUserAuthenticated(Type authenticationType)

Tabla 2 Métodos de la OSID authentication

- osid_authorization: Esta OSID provee una forma de definir quién, está haciendo qué y cuándo. Para ello se basa en los métodos que se recogen en la Tabla 3.

Valor de Retorno	Métodos
boolean	agentExists(object $Id)
object&	createAuthorization(object $Id)
object&	createDatedAuthorization(object $Id, integer $effectiveDate, integer $expirationDate)
object&	createFunction(object $Id, string $displayName, string $description, object $Type)
object&	createQualifier(object $Id, string $displayName, string $description, object $Type)
object&	createRootQualifier(object $Id, string $displayName, string $description, object $Type)
void	deleteAuthorization(object $Authorization)
void	deleteFunction(object $Id)
void)	deleteQualifier(object $Id)
object&	getAllAZs(object $Id, boolean $isActiveNowOnly)
object&	getAllAZsByFuncType(object $Id, object $Type, boolean

	$isActiveNowOnly)
object&	getAllUserAZs(object $Id, boolean $isActiveNowOnly)
object&	getAllUserAZsByFuncType(object $Type, object $Id, boolean $isActiveNowOnly)
object&	getExplicitAZs(object $Id, boolean $isActiveNowOnly)
object&	getExplicitAZsByFuncType(object $Id, object $Type, boolean $isActiveNowOnly)
object&	getExplicitUserAZs(object $Id, boolean $isActiveNowOnly)
object&	getExplicitUserAZsByFuncType(object $Type, object $Id, boolean $isActiveNowOnly)
object&	getExplicitUserAZsForImplicitAZ(object $Authorization)
object&	getFunction(object $Id)
object&	getFunctions(object $Type)
object&	getFunctionTypes()
object&	getQualifier(object $Id)
object&	getQualifierChildren(object $Id)
object&	getQualifierDescendants(object $Id)
object&	getQualifierHierarchies()
object&	getQualifierTypes()
object&	getRootQualifiers(object $Id)
object&	getWhoCanDo(object $Id)
boolean	isAuthorized(object $Id)
boolean	isUserAuthorized(object $Id)
boolean	supportsDesign()
boolean	supportsMaintenance()

Tabla 3 Métodos de la OSID authorization

Harmoni[24] es un framework que tiene definidas algunas de las OSID que propone OKI; como es el caso de Authentication, Authorization, Agent/Group Management, Hierarchy, Id Management y Repository. Está desarrollado sobre PHP. Además tiene implementado algunos otros servicios que no forman parte de las OSID pero que son usados por las mismas. Sin embargo, este framework posee poca documentación sobre su funcionamiento y la forma en que se implementan las OSIDs (sólo aparece de la OSIDs autenticación y autorización). Además definen algunas clases nuevas y constantes numéricas que no son parte de las OSID.

1.3 Arquitectura SOA

Esta arquitectura se distingue de las demás por el acoplamiento débil (loose coupling). Acoplamiento débil significa que el cliente de un servicio es esencialmente independiente del servicio. La manera en que un cliente (el cual puede ser otro servicio) se comunica con éste no es dependiente de la implementación del servicio, por lo que no le es necesario conocer el lenguaje en que se encuentra el mismo, ni en qué plataforma. Esto posibilita que en caso de que cambie la implementación del servicio, el cliente se puede seguir comunicando, siempre y cuando la interfaz permanezca igual. (Booth. D, Haas. H, McCabe. F et all, 2007)

Las arquitecturas anteriores por su parte funcionan bien en sistemas aislados, pero a la hora de integrar sistemas heterogéneos comienzan los problemas, debido a las dependencias entre los componentes. Una solución que se ha venido dando para que estos sistemas logren comunicarse sin tener que modificarlos es a través de los middleware o pasarelas que permiten acceder a los servicios, pero esta no es la mejor solución. Por su parte SOA se basa en la independencia de plataformas, de hardware, de sistemas operativos y de lenguajes de programación.

[24] http://harmoni.sourceforge.net/.

SOA tiene dos elementos principales, las funciones y la calidad de los servicios. Dentro de las funciones (Alvez. P, Foti. P, Scalone. M et all, 2007) identifican:

- **Transporte:** es el mecanismo utilizado para llevar las demandas de servicio desde un consumidor de servicio hacia un proveedor de servicio, y las respuestas desde el proveedor hacia el consumidor.
- **Protocolo de comunicación de servicios:** es un mecanismo acordado a través del cual un proveedor de servicios y un consumidor de servicios comunican qué está siendo solicitado y qué está siendo respondido.
- **Descripción de servicio:** es un esquema acordado para describir qué es el servicio, cómo debe invocarse, y qué datos requiere el servicio para invocarse con éxito.
- **Servicios:** describe un servicio actual que está disponible para utilizar.
- **Procesos de Negocio:** es una colección de servicios, invocados en una secuencia particular con un conjunto particular de reglas, para satisfacer un requerimiento de negocio.
- **Registro de Servicios:** es un repositorio de descripciones de servicios y datos que pueden utilizar proveedores de servicios para publicar sus servicios, así como consumidores de servicios para descubrir o hallar servicios disponibles.

Y dentro de Calidad de Servicio:

- **Política:** es un conjunto de condiciones o reglas bajo las cuales un proveedor de servicio hace el servicio disponible para consumidores.
- **Seguridad:** es un conjunto de reglas que pueden aplicarse para la identificación, autorización y control de acceso a consumidores de servicios.
- **Transacciones:** es el conjunto de atributos que podrían aplicarse a un grupo de servicios para entregar un resultado consistente.

  Administración: es el conjunto de atributos que podrían aplicarse para manejar los servicios proporcionados o consumidos.

Uno de los componentes de esta arquitectura son los Enterprise Service Bus (ESB). Componente que permite conectar los servicios con sus consumidores y que a la vez proporciona, soporte a la heterogeneidad de tecnologías y a la comunicación síncronas y asíncrona. Es un elemento de integración que combina mensajería, servicios web, transformación de datos, enrutación, políticas de seguridad, entre otras. Dentro de las características más comunes (TIBCO, 2007) identifica:

- Validación de la información, de los mensajes entrantes, sobre los que se pueden aplicar distintas condiciones y verificaciones que hagan cumplir los requisitos establecidos.
- Enriquecimiento de los mensajes, posibilidad de añadir información o metainformación adicional para cumplir las necesidades de integración en cada momento.
- Transformación de los mensajes, normalmente entre los diversos modelos de datos del conjunto de servicios y aplicaciones a integrar.
- Ruteo: la mayoría de los ESB permiten definir procesos, ofreciendo alguna (mínimas) lógica de flujos, como estructuras iterativas, condicionales, procesamiento en paralelo, invocación de otros servicios, entre otras.
- Seguridad Incorporada: normalmente cada ESB trae incorporada su propia seguridad referida a la autenticación y autorización de usuarios. También soportan protocolos de comunicación segura, mediante HTTPS.
- Interfaz de usuario: algunos ESB brindan la posibilidad de administrar el bus a través de una interfaz amigable.
- Orquestación de servicios: a través de lenguajes estándares como BPEL, permiten la coordinación y organización de la invocación de los distintos servicios. De esta manera es posible definir procesos de negocio dentro de los ESB.

Por otro lado, se encuentran los Business Process Management (BPM) que no son más que un conjunto de servicios y herramientas que facilitan la administración de procesos de negocio. Por administración de procesos se entiende: análisis, definición, ejecución, monitoreo, y control de los procesos. (Gil, 2007)

Los BPM permiten el ahorro de tiempo y costes de programación, los procesos se ejecutan, monitorizan y optimizan directamente por el personal de una empresa, y permite que los procesos puedan rehacerse indefinidamente sobre la marcha para adaptarse a nuevas necesidades o mejoras. (GONGO LABS, 2008)

Existen diferencias de criterios en cuanto a los niveles de madurez de SOA, este trabajo tomará el definido por (Martínez, 2007), que cuenta de 5 niveles (ver Anexo 1), y que tiene una relación directa con los 5 niveles de CMM[25], sólo que más enfocado a cuestiones técnicas de la propia arquitectura, los niveles son[26]:

1. Initial Services SOA: tiene como propósito introducir funcionalidades y nuevas tecnologías, llevándolo a cabo en proyectos pequeños y utilizando estándares como son (*SOAP, WSDL, XML)*[27], y obteniendo como producto los servicios web.
2. Architected Services SOA: su principal propósito es reducir los costos y lograr que múltiples aplicaciones sean integradas sin importar la hetereogenidad de las mismas. Para ello se utilizan diferentes estándares como son (*UDDI, WS-Security, XQuery)*[28]*.* Todos los servicios son publicados en un ESB producto que se obtiene en este nivel.
3. Business Services SOA: tiene como propósito aumentar la agilidad del negocio, entre diferentes empresas e incluso entre unidades con negocios distintos, para ello utiliza como estándares *WS-BPEL*, y obtiene como producto herramientas y motores *BPM*.

---

[25] Modelo de Capacidad y Madurez, para más información consultar http://es.wikipedia.org/wiki/Modelo_de_Capacidad_y_Madurez.

[26] Se mantendrán los nombres de los niveles de madurez en inglés para no alterar su significado.

[27] Estos términos serán explicados en la próxima sección.

[28] Términos que serán aclarados en la próxima sección.

4. Mesuared Business Services: el propósito principal es el monitoreo de procesos. Se pueden obtener métricas de los procesos de negocio basados en servicio.
5. Optimized Business Services: tiene como propósito la optimización de los procesos de negocio, y el mejoramiento continuo.

En ocasiones se confunde SOA con servicios web, es cierto que los servicios web han impulsado SOA, pero una cosa es un protocolo de comunicaciones y otra es una visión de arquitectura; están a distinto nivel (Martínez, 2004).

A continuación se aclaran las cuestiones generales relacionadas con los servicios web.

1.3.1 ¿Qué son los Servicios Web?

Un servicio web es básicamente una función o procedimiento que puede ser accedida vía web por cualquier programa o aplicación sin importar en qué plataforma reside el servicio, o en qué lenguaje ha sido desarrollado, el término "web" implica que el acceso se hace mediante una conexión a internet habitualmente vía http, aunque otros protocolos de transporte pueden ser utilizados.

La mayoría de las organizaciones protegen sus redes mediante firewalls[29] que bloquean el tráfico de información, para ello cierran la mayoría de los puertos Transmission Control Protocol (TCP) menos el puerto 80 debido a que es el utilizado por los navegadores, esto es aprovechado por los servicios web para enrutarse por este medio, además aportan una gran independencia entre la aplicaciones que usan los servicios web y el propio servicio, por lo que un cambio en uno no debe afectar al otro. Permiten además interoperabilidad entre aplicaciones sin importar las propiedades, y la organización por la que fue desarrollada.

[29] Es un sistema o grupo de sistemas que impone una política de seguridad entre la organización de red privada y el Internet.

El enfoque de servicios web está basado en un conjunto de estándares que están siendo ampliamente aceptados y utilizados. Esto posibilita que clientes y servicios se comuniquen y se entiendan entre sí. A continuación se describe brevemente los protocolos y las tecnologías que forman parte de estos estándares.

1.3.1.1 eXtensible Markup Language (XML).

XML se ha convertido en uno de los estándares de facto más utilizados, como su nombre lo indica, es un lenguaje para el análisis formal de documentos web. Comprende el uso de etiquetas (tags) que identifican los contenidos de un documento, y al hacerlo, los describen. Una etiqueta XML identifica información dentro de un documento, y la estructura de dicha información.

Los documentos XML poseen una estructura bien formada, por ejemplo, cada etiqueta XML debe tener una de cierre, y cualquier etiqueta que comienza dentro de otra debe terminar antes de la culminación de la misma, estando completamente anidada dentro de la otra etiqueta. A un documento bajo este lenguaje generalmente está asociado un schema que especifica sus "reglas gramaticales". En otras palabras, el schema especifica qué etiquetas están permitidas dentro de un documento, la estructura de las mismas, y otras reglas relacionadas con ellas, tales como el tipo de dato que se espera (o la ausencia de datos si es una etiqueta vacía). Esto ha posibilitado que se haya adoptado como el lenguaje de datos para los servicios web. (W3C, 2005)

1.3.1.2 Simple Object Access Protocol (SOAP).

SOAP es un protocolo que establece un formato común de mensajes para el intercambio de datos entre clientes y servicios. El ítem básico de la transmisión es un mensaje SOAP, consta de un sobre *(envelope)* obligatorio, una cabecera *(header)* opcional, y un cuerpo *(body)* obligatorio. (W3C, 2005)

El *sobre* especifica dos cosas: un XML *namespace* y un *encoding style*. El XML *namespace* especifica los nombres que pueden ser utilizados dentro del mensaje SOAP; están destinados a evitar conflictos de nombres –el mismo nombre se puede utilizar para diferentes ítems, siempre que los nombres estén en diferentes namespaces. El encoding style identifica los tipos de datos reconocidos por el mensaje SOAP. Si se provee una cabecera, la misma extiende el mensaje SOAP de forma modular. Este mensaje viaja desde un cliente hacia un servicio, el mismo puede pasar a través de un conjunto de nodos intermediarios, que cada uno es una aplicación que puede recibir y enviar mensajes SOAP. (W3C, 2005)

Un nodo intermedio puede proveer servicios adicionales tales como una transformación de datos, contenidos en el mensaje o ejecutar operaciones relacionadas con la seguridad. La cabecera SOAP se puede utilizar para indicar algún tipo de procesamiento adicional en un nodo intermedio. Generalmente, la cabecera se utiliza para comunicar información relacionada con la seguridad que ha de ser procesada en tiempo de ejecución y el cuerpo contiene la parte principal del mensaje.

1.3.1.3 Web Services Description Language (WSDL).

Una vez que el cliente va a realizar la solicitud a un servicio debe tener un formato para realizarla, tanto el cliente como el que brinda el servicio debe tener un mecanismo para saber qué significa esa solicitud. Esta información se encuentra dentro del XML, en un documento denominado WSDL, la misma cuenta con una descripción de la interfaz del servicio web. Este lenguaje define un XML schema para la descripción de un servicio web. (W3C, 2005)

Para saber la descripción correspondiente a un servicio web, el cliente necesita encontrar el documento WSDL del servicio. Para ello le es necesario encontrar en el XML un apuntador hacia el documento WSDL, la que puede estar en un registro UDDI o en un registro/repositorio XML.

Un documento WSDL describe a un servicio web como un conjunto de ítems abstractos denominados “puertos” o “puntos extremos”. También define las acciones ejecutadas por un servicio web y los datos transmitidos a estas acciones, de una manera abstracta. Las acciones están representadas por “operaciones”, y los datos están representados por “mensajes”. Un conjunto de operaciones relacionadas se conocen como un “tipo-de-puerto”.

Un tipo-de-puerto constituye el conjunto de acciones ofrecidas por un servicio web. Lo que convierte a una descripción WSDL de abstracta a concreta es un “binding”. Un binding especifica el protocolo de red y las especificaciones del formato del mensaje para un tipo-de-puerto particular. Un Puerto queda definido mediante la asociación de una dirección de red con un binding. Si un cliente localiza un documento WSDL y encuentra el binding y la dirección de red correspondiente a cada puerto, puede llamar a las operaciones del servicio de acuerdo al protocolo y al formato de mensaje especificados. (W3C, 2005)

1.3.1.4 Universal Description, Discovery and Integration (UDDI).

UDDI es un directorio de servicios web donde se puedan publicar los servicios ofrecidos, dar características del tipo de servicio, y realizar búsquedas. Es un elemento central del grupo de estándares involucrados en la tecnología de servicios web. Es el mediador a través del cual se conocen los posibles clientes con los proveedores de los servicios. (W3C, 2005)

Es un modelo de directorio para servicios web en el que se pueden realizar búsquedas, y a la vez publicar nuevos servicios desarrollados. Para describir las interfaces de los servicios web utiliza WSDL.

1.3.1.5 Estándar para garantizar la seguridad de los mensajes (WS-Security).

XML, SOAP, UDDI y WSDL son estándares que atienden los aspectos básicos de la interoperabilidad de servicios. Ellos aseguran que un cliente puede encontrar a un servicio requerido y realizar una solicitud que tanto el cliente como el servicio pueden comprender, independientemente de dónde residan el cliente y el servicio o en qué lenguaje han sido codificados, pero queda fuera de esto las áreas de seguridad. Varias organizaciones de estandarización, tales como la World Wide Web Consortium (W3C) y Organization for the Advancement of Structured Information Standards (OASIS) han generado borradores de propuestas a estándares en esta área.

WS-Security es un estándar liberado por OASIS en marzo de 2004. El mismo describe mejoras relacionadas con la seguridad aplicadas al SOAP messaging[30] que atienden a la integridad y la confidencialidad del mensaje. Integridad significa que un mensaje SOAP no es manipulado en su viaje desde un cliente hasta su destino final. Confidencialidad significa que un mensaje SOAP sólo es visualizado por los receptores a los que está destinado.

Cuando un emisor es autenticado combina un token[31] de seguridad con una firma digital, esta última se utiliza con el objetivo de comprobar que el emisor está asociado efectivamente con el token de seguridad y éste se asocia con un mensaje. Este estándar puede utilizar varias tecnologías de encriptación, tal como la Public-Key Infraestructure-PKI- y Kerberos, y el procolo Secure Socket Layer / Transport Layer Security (SSL/TLS) que provee lo básico para dotar de seguridad extremo a extremo a las comunicaciones sobre la Internet.

---

[30] Mensajes SOAP.

[31] Conjunto de declaraciones realizadas por el emisor de un mensaje SOAP.

1.3.2 Principios básicos para publicar servicios web.

No existe una definición estándar de cuáles son los Principios de la Orientación a Servicios, por lo tanto, lo único que se puede proporcionar es un conjunto de Principios que estén muy asociados con la Orientación a Servicios. Estos Principios según (Erl, 2007) son:

- Los Servicios deben ser reusables: Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.
- Los Servicios deben proporcionar un contrato formal: Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionales que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. De esta manera, todo consumidor del servicio, accederá a este mediante el contrato, logrando así la independencia entre el consumidor y la implementación del propio servicio. En el caso de los Servicios Web, esto se logrará mediante la definición de interfaces con WSDL.
- Los Servicios deben tener bajo acoplamiento: Es decir, que los servicios tienen que ser independientes los unos de los otros. Para lograr ese bajo acoplamiento, lo que se hará es que cada vez que se vaya a ejecutar un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama. Si conseguimos este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.
- Los Servicios deben permitir la composición: Todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel. En el caso de los Servicios Web, esto se logrará mediante el uso de los protocolos para orquestación (WS-BPEL) y coreografía (WS-CDL).
- Los Servicios deben de ser autónomos: Todo Servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y

nos podemos asegurar que así podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.

- Los Servicios no deben tener estado: Un servicio no debe guardar ningún tipo de información. Esto es así porque una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución, es que un servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema de información sea del tipo que sea.
- Los Servicios deben poder ser descubiertos: Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los Servicios Web, el descubrimiento se logrará publicando los interfaces de los servicios en registros UDDI.

1.4 Tendencia actual de interoperabilidad entre herramientas e-learning.

Diferentes iniciativas encaminadas a la interoperabilidad entre las herramientas e-learning se han venido desarrollando hace algunos años, una de las más importantes es el establecimiento de estándares, como los explicados en secciones anteriores, que han propiciado un entendimiento común en el ámbito de los desarrolladores de plataformas e-learning.

Son varios los proyectos que existen y que han puesto en práctica muchos de estos estándares, tal es el caso de SUMA, ARIADNE, ELENA, Agrega, REDOUAA, CAMPUS entre otros. A continuación se explicarán algunos de los proyectos más maduros y de mayor envergadura y que son de referencia obligada para este trabajo.

1.4.1 Agrega.

Agrega[32] no es más que un modelo de repositorios interoperables, utilizando estándares y protocolos Open Source de amplia aceptación por la comunidad educativa tal es el caso de IMS (IMS-DRI), que proporciona la funcionalidad básica para explotar los objetos digitales presentes en el repositorio (enviar/almacenar, y solicitar/entregar). Se ha implementado usando servicios web, y SOA como arquitectura. En la definición de los servicios no incluidos en IMS DRI se han incorporado las OSIDs de OKI.

Referente a las búsquedas de contenidos entre repositorios se ha implementado usando la especificación SQI. Respecto al mecanismo interno de búsqueda en un nodo[33], se basa en la indexación de un subconjunto de los metadatos que están contenidos en el manifiesto de un objeto. Estos metadatos son indexados a través de una herramienta denominada Lucene, que es la que se usa posteriormente para llevar a cabo las búsquedas. Para el Almacenamiento de contenidos utiliza el estándar SCORM 2004. La información de catalogación se almacena conforme al estándar LOM v.1.0. (Canabal. J & Sarasa. A, 2007)

1.4.2 REDOUAA.

La Universidad Autónoma de Aguascalientes (UAA) es otro de los ejemplos que ha trabajado en interconectar su repositorio con otros, utilizando el protocolo estándar SQI. También dicha universidad cuenta con un LMS (Moodle) por lo que fue necesario asociarle a dicha plataforma un cliente SQI para hacer factible solicitar el servicio y

---

[32] http://redes.agrega.indra.es

[33] Se considera un repositorio

consulta de OA a los repositorios de la Federación Latinoamericana de Repositorios de Objetos de Aprendizaje (FLO)[34] los cuales cuentan con una interfaz SQI cliente/servidor.

A pesar de que se logra la comunicación tanto entre repositorio como con el LMS, no se aprovecha todas las ventajas que trae consigo SOA y OKI.

1.4.3 CAMPUS.

Este proyecto fue promovido por la Secretaría de Telecomunicaciones y Sociedad de la Información (STSI) de la Generalitat de Catalunya. Nace del acuerdo suscrito por la mayoría de universidades catalanas para disponer de un campus virtual basado en código abierto y que permita impartir la enseñanza superior tanto en línea como en semipresencial. Es uno de los proyectos de mayores logros en cuento a la interoperabilidad en la actualidad.

Utiliza seis servicios básicos para la interconexión, ellos son: autenticación, autorización, seguimiento, internacionalización, configuración e instalación.

- El *servicio de autenticación* permite registrar un nuevo usuario en el sistema o saber si un usuario está conectado a él. Es un servicio indispensable en cualquier programa informático con registro de usuarios.

- El *servicio de autorización* permite saber si un usuario está autorizado para acceder a una funcionalidad o recurso. Es indispensable en cualquier sistema donde los usuarios desempeñan distintos roles.

[34] Federación que permitirá a las instituciones de América Latina compartir sus recursos a través de repositorios.

- El *servicio de seguimiento* permite almacenar datos de la actividad que realizan los programas. Es de gran utilidad para saber qué está pasando dentro de un sistema y cómo está funcionado.

- El *servicio de internacionalización* permite cambiar el idioma de un programa o añadir nuevos idiomas.

- El *servicio de configuración* permite crear o cambiar los parámetros de configuración de una aplicación informática.

- El *servicio de instalación* permite añadir una nueva herramienta al sistema.

CAMPUS[35] logra que las herramientas utilicen estos servicios y a la vez interactúen con las plataformas base (Moodle y Sakai) (Ver Figura 5)

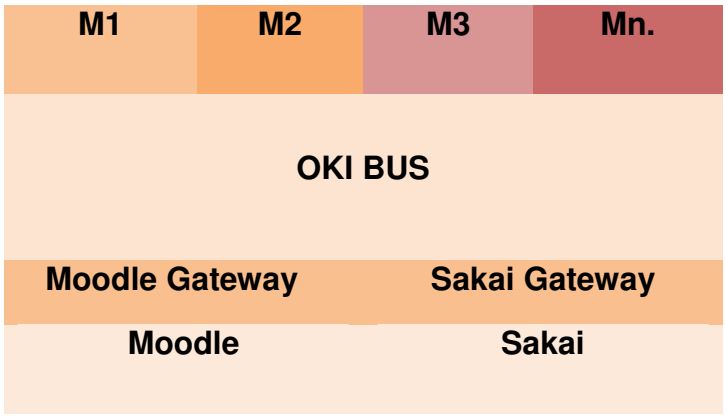


Figura 5 Arquitectura del proyecto CAMPUS

A su vez, una plataforma de aprendizaje que quiera usar los módulos debe disponer de un OKI Gateway, que no es más que una pieza de software que traduce las peticiones de

[35] http://www.campusproject.org.

los servicios de base que utilizan los módulos a llamadas a las API de la propia plataforma. Cada plataforma tiene el suyo. En la Fig. 5 se aprecia el OKI moodle gasteway (para la plataforma Moodle) y el OKI sakai gateway (para la plataforma Sakai). Para integrar una nueva plataforma debe implementarse el correspondiente OKI gateway.

También cuenta con una capa OKIBus. Este componente es una capa intermedia que resuelve toda la problemática relacionada con la comunicación entre aplicaciones donde se publican los servicios necesarios para integrar herramientas nuevas y que a la vez se comuniquen con los LMS bases. (CAMPUS, 2007), (Santanach. F, Casamajó. J, Casado. P, et all, 2007).
CAMPUS no propone la arquitectura para el desarrollo de herramientas que aprovechen las ventajas de una arquitectura SOA.

1.4.4 ROA, ROXS y Moodle.

Como se mencionó anteriormente, en la UCI se han desarrollado varias herramientas para la gestión de contenidos educativos como es el caso de ROA, ROXS y algunas personalizaciones y extensiones de Moodle. A continuación se describen las características fundamentales de estas herramientas.

1.4.4.1 ROA.

Este repositorio gestiona los contenidos a través del estándar SCORM, cuenta con varios roles (invitado, usuario, autor, revisor, administrador), permitiendo tener un mejor control de los paquetes subidos al sistema, además posibilita búsquedas tanto general[36] como avanzada, ésta última posibilita la búsqueda hasta por tres metadatos simultáneamente, cuenta con una mensajería interna, un panel de administración donde puede gestionar a

[36] Busca por varios metadatos.

los usuarios, categorías, configurar el sistema y una sección dedicada a la seguridad para tener un control de las actividades realizadas en el mismo por cualquier usuario.

Además permite el trabajo metodológico en equipos con los contenidos a través del manejo de roles, como se dijo anteriormente gestiona los OA empaquetados por el estándar SCOM, aspectos que lo distinguen de los más destacados en el mercado, que a pesar de ser llamados repositorios tienen las características de bibliotecas digitales pues almacenan cualquier tipo de recurso.

En su segunda etapa de desarrollo se le incorporó una nueva funcionalidad que permite la integración con otros repositorios e incluso con Moodle. Para ello se implementó el estándar SQI con sus 9 métodos, posibilitando búsquedas tanto síncronas como asíncronas y todas las ventajas que brinda dicho estándar para garantizar la interoperabilidad. Posibilitando así la reutilización de los contenidos.

1.4.4.2 ROXS.

ROXS, nombre con el cual se identifica la herramienta de autor desarrollada en la UCI, permite describir y empaquetar los OA guiándose por el estándar SCORM, tiene un solo rol el usuario, permitiéndole crear/modificar/eliminar un paquete, así como generar una vista previa. Actualmente se está trabajando en una nueva versión la cual cuenta con nuevas funcionalidades.

Dicha herramienta aún no se integra con ROA y Moodle, una vez que se genera el paquete este no puede pasar automáticamente al ROA. En próxima versión se garantizará dicha comunicación, en la cual los paquetes que sean creados puedan ser almacenados en el repositorio o utilizados en Moodle. Por otro lado, una vez que se vaya a diseñar un paquete se podrán realizar búsquedas en el repositorio posibilitando así la reutilización de los contenidos.

1.4.4.3 Moodle.

Moodle es un sistema que posee las características de un LMS. Su objetivo es la creación de cursos basados en Internet. Es un proyecto en desarrollo constante, diseñado para dar soporte a un marco[37]. Se distribuye gratuitamente como Software Libre bajo la Licencia Pública GNU (GPL) gracias a lo cual se ha convertido en una de las plataformas de aprendizaje más extendidas y usadas, con una amplia comunidad de usuarios y desarrolladores, lo que ha posibilitado realizar personalizaciones a la misma. Posibilita el apoyo a la docencia presencial, semipresencial y a distancia.

Permite distribuir materiales de aprendizaje, crear y gestionar debates temáticos y tablones de anuncios, cuestionarios a los estudiantes, evaluar tareas, visualizar recursos digitales, crear glosarios, gestionar el tiempo a través de un calendario global de distintas asignaturas, ofrece herramientas de comunicación entre los estudiantes, como la mensajería instantánea, permite la tutoría electrónica en privado o en grupo, gestiona las calificaciones, etcétera. (Molist, 2006).

Para llevar a cabo la integración con el ROA desarrollado en la UCI como se dijo anteriormente, se le incorporó una actividad, la cual permite realizar búsquedas en el repositorio y una vez localizado el paquete SCORM o un recurso en específico, subirlo a la plataforma, utilizando para ello el estándar SQI. Además, de almacenar en el repositorio los contenidos que se encuentran en Moodle, gracias a un módulo que se le incorpora (C2SCORM) el cual permite extraer los cursos en forma de un paquete SCORM.

A pesar de que esta solución permite la comunicación entre ROA y Moodle, no aprovecha todas las ventajas de una arquitectura SOA, puesto que cada aplicación expone sus interfaces y se realizan comunicaciones punto a punto entre ellas. Este modelo trae

[37] de educación social constructiva.

aparejados problemas como dependencias entre sistemas y dificultad de mantenimiento de las aplicaciones (debido al acoplamiento).

1.5 Conclusiones del capítulo.

En este capítulo se abordaron los conceptos fundamentales relacionados con el e-learning. Además se realizó un análisis de los estándares de interoperabilidad entre herramientas e-learning, de los principales componentes que conforman una arquitectura SOA y de los proyectos con mayores logros en este ámbito.

CAPÍTULO 2

DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

2. Introducción.

En este capítulo se define una arquitectura para el desarrollo de un framework, que garantice la interoperabilidad entre herramientas para la gestión de contenidos educativos. Para ello el capítulo se ha estructurado en tres partes fundamentales.

Una primera parte, con la descripción detallada de la arquitectura del framework, explicando cada una de las capas que conforman la solución.

Una segunda parte, con un conjunto de estándares que se utilizarán en la solución, basándose principalmente en una descripción de los métodos que los conforman, así como los niveles de madurez que puede alcanzar una aplicación al utilizar como arquitectura base dicho framework, teniendo en cuenta los niveles de madurez de SOA explicados en el Capítulo 1.

Y finalmente, se hace referencia a las proyecciones futuras para lograr que el framework alcance un nivel más alto de madurez, no sólo profundizando en los estándares sino también en los elementos que forman parte de una arquitectura SOA.

2.1 Arquitectura del framework.

Existen en la actualidad gran cantidad de herramientas para la gestión de los contenidos educativos. El gran potencial de estas tecnologías, reside en la distribución y reutilización de OA. Muchas de estas herramientas poseen arquitecturas diferentes y no incluyen mecanismos de interoperabilidad.

Como se pudo apreciar en el Capítulo 1, en los últimos años la arquitectura SOA ha tenido gran auge, así como los estándares que se han propuestos para garantizar la interoperabilidad entre sistemas e-learning. La arquitectura del framework que se describe a continuación, garantiza el desarrollo de herramientas interoperables para la gestión de contenidos educativos, así como la adición de otras cuya arquitectura no puede ser cambiada. Posee tres capas, como se muestra en la Figura 6, presentación, servicios, y acceso a datos.

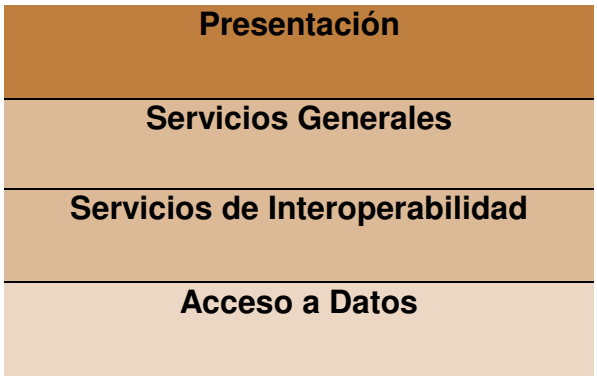


Figura 6 Arquitectura por capas del framework

Capa Presentación: es la responsable de interactuar con el usuario de la aplicación. Maneja el contexto del usuario y le permite interactuar con la capa de negocio (servicios) dónde esta implementada toda la lógica de la aplicación.

Capa de Servicios ó negocio: BUS de servicios que publica tanto los servicios necesarios para garantizar la interoperabilidad como otros servicios adicionales[38]. En esta capa se manejará el negocio del sistema. La responsabilidad de la misma es implementar lógica de negocio que será consumida por la interfaz del sistema y por otros sistemas a través de los servicios compartidos. Dichos servicios se basan en un conjunto de estándares de comunicación, como son XML para la representación de datos, SOAP

[38] Servicios de funcionalidades específicas de cada herramienta.

para el intercambio de datos y el lenguaje WSDL para describir las funcionalidades de un servicio web.

Capa de Acceso a Datos: El acceso a los datos se realiza a través del BUS de servicios. Permite conectar la capa de negocio con los orígenes de los datos, accediendo a los datos que la capa de negocio necesita para funcionar.

En la Figura 7 se puede apreciar la estructura de la capa servicios de interoperabilidad, y los diferentes estándares que se utilizan en la solución.

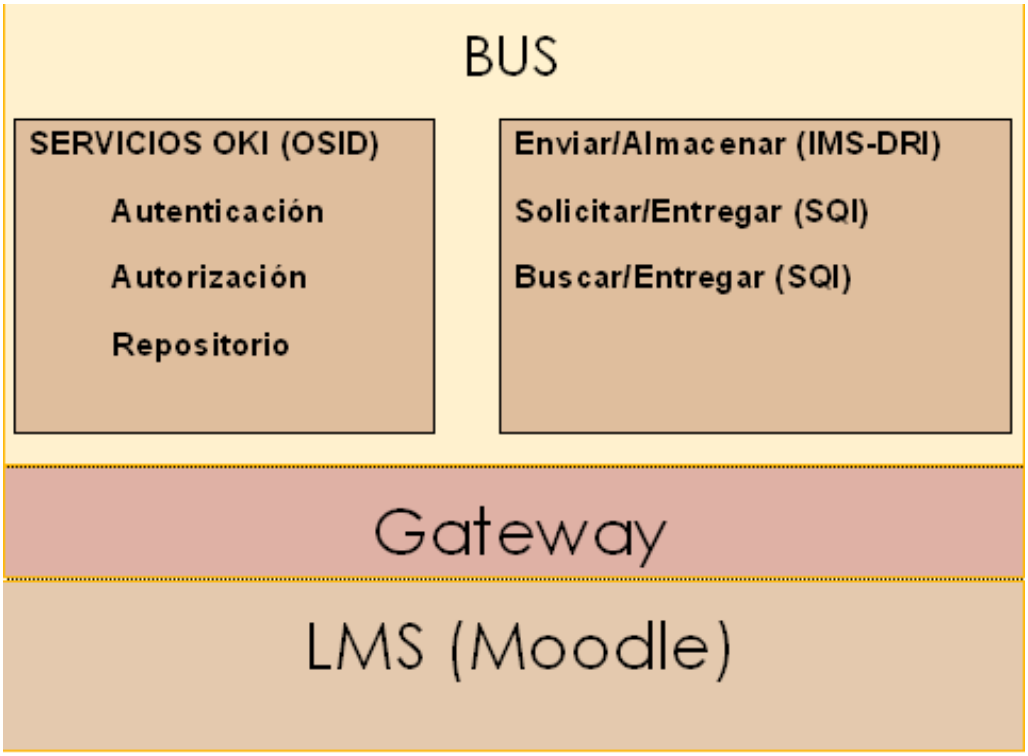


Figura 7 Arquitectura de la capa servicios de interoperabilidad

Dada la gran madurez y larga trayectoria de los entornos de e-learning existentes Ej. Moodle, Sakai, etc, se hace muy difícil cambiar la arquitectura de los mismos, como se dijo anteriormente. Es por ello que el Framewok incorpora una capa gateway o pasarela

entre estos sistemas y la capa de de servicios interoperables, con el objetivo de traducir las peticiones que se realizan.

Como se puede apreciar una de las aristas fundamentales en la solución, son los estándares de interoperabilidad entre aplicaciones. En el Capítulo 1 se hace referencia a los más utilizados en este ámbito, seleccionando IMS-DRI para establecer las funciones básicas en un repositorio, y para definir los métodos necesarios a la hora de establecer una comunicación entre sistemas e-learning SQI y OKI. A continuación se explican cada uno de ellos.

2.2 Estándares que conforman el framework.

2.2.1 IMS Distributed Repository Interoperability (IMS-DRI).

Como se explicó anteriormente, este estándar define las funciones básicas que se realizan en un repositorio con el objetivo de lograr la interoperabilidad, recomendando cuales estándares y especificaciones utilizar. Como bien expresa en su definición, IMS-DRI no obliga a utilizar estas propuestas.

En este framework sólo se utilizarán tres de las funciones básicas que define este estándar enviar/almacenar, solicitar/entregar y buscar/entregar, debido a que alertar/exponer y colectar/exponer, no son funciones necesarias para establecer una comunicación. Estas funciones serán implementadas usando servicios web, los que a su vez formarán parte de una arquitectura SOA.

Para el método de enviar/almacenar, se tendrá en cuenta el estándar que propone IMS-DRI, o sea mensajes SOAP para el envío de los contenidos. En el caso solicitar/entregar y buscar/entregar se utilizará el estándar SQI, debido a las ventajas que tiene dicho estándar con respecto a OAI que es el que propone IMS-DRI para el manejo de esquema de metadatos. OAI sólo es posible utilizarlo para repositorios que utilizan DublinCore como esquema de metadatos, principalmente en bibliotecas digitales, y SQI por su parte

no obliga un estándar específico para describir los OA. A continuación se explican los métodos de SQI que serán utilizados.

2.2.2 Simple Query Interface (SQI).

SQI propone 9 métodos como se expuso anteriormente, para establecer la comunicación con otros repositorios, posibilitando así búsquedas federadas de los mismos, y también con otros sistemas externos. A continuación se explican estos métodos, teniendo en cuenta el valor de retorno, una breve descripción y los diferentes parámetros que se le deben pasar (Simon. B, Massart. D, Duval. E, 2004).

- **SetQueryLanguaje ():** Permite a la fuente tener el control de la sintaxis usada para identificar el lenguaje de la consulta. Los valores para el parámetro queryLanguageID se distingue entre mayúsculas y minúsculas.

Tipo de retorno: Void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
queryLanguageID	String

Errores:

NO_SUCH_SESSION: en caso de TargetSessionID sea inválidos.

QUERY_LANGUAGE_NOT_SUPPORTED: si el lenguaje utilizado en la solicitud no es compatible con el de la fuente.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **SetMaxQueryResults ():** Este método define el número máximo de resultados, que puede producir una consulta. El máximo se fija en 100 por defecto, pero se puede controlar por medio de MaxQueryResults(), que debe ser cero (0) o mayor. Si se pone en cero (0) el número máximo de consultas, la fuente acepta todas los resultados producidos por la misma.

Tipo de retorno: Void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
maxQueryResults	Integer

Errores:

NO_SUCH_SESSION: en caso de TargetSessionID sea inválidos.

INVALID_MAX_QUERY_RESULTS: si el número previsto por maxQueryResults() es inválido.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **SetResultsFormat ():** Este método permite a la fuente tener el control del formato de resultado retornado por el destino, y que es especificado por el parámetro resultsFormat.

Tipo de retorno: Void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
resultsFormat	String

Errores:

NO_SUCH_SESSION: en caso de TargetSessionID sea inválidos.

RESULTS_FORMAT_NOT_SUPPORTED: cuando el formato proveído por el parámetro resultsFormat no es soportado por el destino.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **SetMaxDuration ():** este método permite a la fuente establecer un tiempo de consulta en caso de una interfaz de consulta asincrónica. Los valores de maxDuration deben ser cero (0) o mayor, y son interpretados en milisegundos, si no se define un valor, toma por defecto cero (0).

Tipo de retorno: Void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
maxDuration	Integer

Errores:

NO_SUCH_SESSION: en caso de TargetSessionID sea inválidos.

INVALID_MAX_DURATION: si el número previsto por maxDuration () es inválido.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **ResultsSetSize ():** Este método define el número máximo de resultados. El tamaño de los resultados está establecido por 25 registros por defecto, pero puede ser controlado por el método resultsSetSize que puede ser cero (0) o mayor.

Tipo de retorno: Void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
resultsSetSize	Intiger

Errores:

NO_SUCH_SESSION: en caso de TargetSessionID sea inválidos.

INVALID_RESULTS_SET_SIZE: si el número de resultsSetSize es inválido.

QUERY_MODE_NOT_SUPPORTED: en caso de que el destino no soporte una consulta síncrona.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **SynchronousQuery ():** Este método envía una consulta al destino conformado por varios parámetros, uno de ellos es queryStatement, el cual contiene el cuerpo de la consulta, y targetSessionID que identifica la sesión, el método devuelve los resultados dependiendo de la cantidad de coincidencias encontradas. El parámetro startResult identifica el comienzo de una serie de resultados, y que éste debe

estar entre el rango 1 y el número total de resultados que se define por el setResultsSetSize, y el máximo permitido está dado por setMaxQueryResults.

Tipo de retorno: String

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
queryStatement	String
startResult	Integer

Errores:

NO_SUCH_SESSION: en caso de *TargetSessionID* sea inválidos.

INVALID_ QUERY_STATEMENT: si la consulta no cumple con la sintaxis del lenguaje de consulta.

QUERY_MODE_NOT_SUPPORTED: en caso de que el destino no soporte consultas síncronas.

METHOD_FAILURE: Si la operación es fallida por otras razones.

INVALID_START_RESULT si es inválido el número que es propuesto por *startResul.*

NO_MORE_RESULTS: si *startResult* está con 0 y no hay más resultados disponibles.

- **GetTotalResultsCount ():** Este método retorna el número total de resultados disponibles por una consulta. Se le pasa como parámetro el targetSessionID para identificar la sesión y el cuerpo de la consulta a través del parámetro queryStatement.

Tipo de retorno: Intiger

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
queryStatement	String

Errores:

NO_SUCH_SESSION: en caso de *TargetSessionID* sea inválidos.

INVALID_ QUERY_STATEMENT: si la consulta no cumple con la sintaxis del lenguaje de consulta.

QUERY_MODE_NOT_SUPPORTED: en caso de que el destino no soporte consultas síncronas.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **SetSourceLocation ():** Este método es requerido antes de que se produzca una consulta asíncrona. El parámetro sourceLocation especifica la localización del listener[39] de la fuente para que el destino pueda ubicarlo, esto es a través de una URL.

Tipo de retorno: void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
sourceLocation	String

Errores:

NO_SUCH_SESSION: en caso de *TargetSessionID* sea inválidos.

QUERY_MODE_NOT_SUPPORTED: en caso de que el destino no soporte consultas síncronas.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **AsynchronousQuery ():** Este método permite a la fuente enviar una consulta al destino, mientras los resultados son retornados en un modo asincrónico. El cuerpo de la consulta se especifica por el parámetro queryStatement. Además se le pasa un identificador de consulta que es emitido por la fuente y se requiere para hacer un vínculo de los resultados de la consulta, a la consulta cuando éstos son más

[39] Un escuchador que espera por los resultados.

tarde retornados usando el results listener. Mediante el uso de un identificador de las consultas es posible enviar un número arbitrario de consultas por cada sesión activa. La localización del listener en la fuente se necesita y se debe proveer usando el método setSourceLocation. Debido a la naturaleza asincrónica de este método, pueden llegar resultados de consultas anteriores. La consulta se procesa y los resultados se envían en el plazo señalado en el método setMaxDuration.

Tipo de retorno: void

Parámetros:

Nombre:	*Tipo:*
targetSessionID	String
queryStatement	String
queryID	String

Errores:

NO_SUCH_SESSION: en caso de *TargetSessionID* sea inválidos.

QUERY_MODE_NOT_SUPPORTED: en caso de que el destino no soporte consultas síncronas.

NO_SOURCE_LOCATION: en caso de que la fuente no especifique su localización.

INVALID_QUERY_STATEMENT: si la consulta no cumple con la sintaxis del lenguaje de consulta.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **QueryResultsListener ():** Este método, iniciado por el destino, devuelve los resultados hacia la fuente. El parámetro queryID es usado para referenciar los resultados con la previa consulta enviada, cuando éstos son retornados más tarde usando el results listener[40]. El queryResults tiene un conjunto de resultados que consiste en una lista de registros de metadatos, los cuales concuerdan con el esquema especificado en la consulta.

[40] Oyente de resultados.

Tipo de retorno: void

Parámetros:

Nombre:	*Tipo:*
queryID	String
queryResults	String

Errores:

METHOD_FAILURE: Si la operación es fallida por otras razones.

INVALID_QUERY_RESULTS: en caso de que los resultados no sean interpretados por la fuente.

NO_SUCH_QUERY: en caso de que el parámetro queryID sea inválido.

- **CreateSession ():** Este método crea la sesión, para ello requiere los parámetros userID and password, retornando un Session ID.

Tipo de retorno: String

Parámetros:

Nombre:	*Tipo:*
userID	String
password	String

Errores:

WRONG_CREDENTIALS: en caso de que el *userID* and/or *password* sea inválido.

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **CreateAnonymousSession ():** Este método crea una sesión sin necesidad de una cuenta en el sistema, el mismo devuelve un ID de la sesión.

Tipo de retorno: String

Parámetros: _

Errores:

METHOD_FAILURE: Si la operación es fallida por otras razones.

- **DestroySession ():** Este método destruye la sesión, se identifica la misma a través del parámetro sessionID.

Tipo de retorno: void

Parámetros:

Nombre:	*Tipo:*
sessionID	Stirng

Errores:

NO_SUCH_SESSION: Si el sessionID es inválido.

METHOD_FAILURE: Si la operación es fallida por otras razones.

Para la comunicación entre sistemas e-learning heterogéneos, el estándar OKI recoge en un grupo de OSID, funcionalidades que permiten dicha interoperabilidad.

2.2.3 Las guías OSID de OKI.

Cada OSID se caracteriza por una serie de métodos que permiten un proceso en específico, como es el caso de la autenticación. Las OSID son neutras en cuanto al lenguaje de programación.

El framework tendrá tres de las OSID que propone OKI, tal es el caso de la authentication, authorization y repository. Como se dijo anteriormente, cada una de estas OSID cuenta con un conjunto de métodos, los cuales serán explicados destacando los parámetros, tipo de retorno y en qué consiste el método[41].

2.2.3.1 Métodos de la OSID Authentication.

- **GetAuthenticationTypes ():** Este método devuelve los tipos de autenticación que son soportados por la implementación.

---

[41] En algunos de los casos se mantendrán palabras en inglés para no alterar su significado.

Tipo de retorno: object TypeIterator
Parámetros: _

- **authenticateUser ():** Invoca el proceso de autenticación del tipo especificado para identificar al usuario. Puede ser necesario llamar a isUserAuthenticated para comprobar el estado de autenticación. Este estándar limita el tiempo de autenticación de los usuarios; para ello requiere consultas explícitas del estado de autenticación. El control del estado de autenticación se producirá con más frecuencia que invocar el mecanismo para autenticar al usuario.
Tipo de retorno: void
Parámetros: authenticationType

- **destroyAuthentication ():** Destruye la autenticación para todos los tipos de autenticación.
Tipo de retorno: void
Parámetros: _

- **destroyAuthenticationForType ():** Destruye la autenticación para los tipos de autenticaciones especificadas.
Tipo de retorno: void
Parámetros: authenticationType

- **getUserId ():** Retorna un único ID del agente que representa al usuario para un tipo de autenticación específica. Estos agentes son administrados en la Shared OSID[42].
Tipo de retorno: object Id
Parámetros: authenticationType

[42] Otra OSID que puede ser consultada en http://www.okiproject.org/.

• **isUserAuthenticated ():** Comprueba el estado actual de autenticación del usuario. Si retorna verdadero, el usuario fue autenticado y si retorna falso no logró dicha autenticación. Esto último puede indicar que los usuarios no pueden ser autenticados o que se ha caducado su cuenta.
Tipo de retorno: boolean
Parámetros: authenticationType

2.2.3.2 Métodos de la OSID Authorization.

• **agentExists ():** Retorna verdadero si el agentId existe en el servicio de autorización y falso en caso contrario.
Tipo de retorno: boolean
Parámetros: object $Id

• **createAuthorization ():** Crea una nueva autorización para que un agente[43] realice funciones con un calificador [44]. Usa la fecha y el tiempo actual como la effectiveDate y no modifica la fecha de expiración.
Tipo de retorno: object $Id
Parámetros: object Authorization

• **createDatedAuthorization ():** Crea una nueva autorización para que un agente realice una función con un calificador.
Tipo de retorno: object Authorization
Parámetros: Id - $agentId, effectiveDate , expirationDate –

• **createFunction ():** Los IDs en Autorización son definidos en el exterior y su singularidad es forzada por la aplicación.

---

[43] Una representación de un director o actor conocido a un sistema.

[44] Un identificador para el objeto de una acción de una Function. Un calificador tiene uno y sólo un QualifierType.

Parámetros: Id - $functionId, displayName , description , Type - $functionType
Retorna: object Function

- **createQualifier ():** Los IDs en autorización son definidos en el exterior y su singularidad es forzada por la aplicación. Crea un nuevo calificador en el servicio de autorización. Esto es diferente a hacer una nueva instancia de un calificador local ya que el calificador será insertado en el servicio de autorización.

Parámetros: Id - $qualifierId, displayName , description , Type - $qualifierType
Retorna: object Qualifier

- **createRootQualifier ():** Crea un nuevo calificador en el servicio de autorización que no tiene padres. Esto es diferente de la realización de una nueva instancia de un objeto clasificatorio a nivel local ya que este será insertado en los servicios de autorización

Parámetros: Id - $qualifierId, displayName, description, Type - $qualifierType
Retorna: object Qualifier

- **deleteAuthorization ():** Elimina una autorización existente.

Parámetros: Authorization - $authorization
Retorna: _

- **deleteFunction ():** Elimina una function[45] por ID.

Parámetros: Id - $functionId
Retorna: _

- **deleteQualifier ():** Elimina un calificador por Id. El calificador raíz no puede ser eliminado.

Parámetros: Id - $qualifierId
Retorna: _

[45] Una acción o rol que uno puede ser autorizado a tomar. Una función tiene una y solo una FunctionType.

• **getAllAZs ():** Dada una functionId y un calificador devuelve todas las autorizaciones que permitirán a los agentes hacer la function con el calificador. Este método difiere de getAuthorizations en que busca cualquier autorización que permite hacer la function con el calificador, incluso si el calificador de autorización pasa a ser un padre de este argumento calificatorio.
Parámetros: Id - $agentId, isActiveNowOnly
Retorna: object AuthorizationIterator

• **getAllAZsByFuncType ():** Dada una FunctionType y un calificador devuelve todas las autorizaciones que permitan a los agentes hacer función en el FunctionType con el calificador. Este método difiere de getAuthorizations en que este método busca cualquier autorización que permite hacer la function con el calificador, incluso si el calificador de la autorización pasa a ser un padre del argumento calificatorio.
Parámetros: Id - $agentId, Type - $functionType, isActiveNowOnly
Retorna: object AuthorizationIterator

• **getAllUserAZs ():** Dada una functionId y un calificador devuelve todas las autorizaciones que permiten al usuario hacer la función con el calificador. Este método difiere de la simple forma de getAuthorizations en que este método busca cualquier autorización que permite hacer la función con el calificador, incluso si el calificador de la autorización pasa a ser un padre de este argumento calificador.
Parámetros: Id - $functionId, isActiveNowOnly
Retorna: object AuthorizationIterator

• **getAllUserAZsByFuncType ():** Dada una FunctionType y un calificador devuelve todas las autorizaciones que permiten al usuario realizar las functions en el FunctionType con el calificador. Este método difiere de getAuthorizations en que este método busca cualquier autorización que permite hacer la función con el

calificador, incluso si el calificador de la autorización pasa a ser un padre del argumento calificador.
Parámetros: Type - $functionType, Id - $qualifierId, isActiveNowOnly
Retorna: object AuthorizationIterator

- **getExplicitAZs ():** Dado un agentId, un functionId, y un qualifierId (al menos uno de estos debe ser no nulo) devuelve las correspondientes autorizaciones.
Parámetros: Id - $agentId, isActiveNowOnly
Retorna: object AuthorizationIterator

- **getExplicitAZsByFuncType ():** Dado un agentId, un FunctionType, y un calificador (ya sea agentId o qualifierId debe ser no nulo) devuelve las correspondientes autorizaciones. Las autorizaciones deben ser para las funciones dentro de los FunctionType dado.
Parámetros: Id - $agentId, Type - $functionType, isActiveNowOnly
Retorna: object AuthorizationIterator

- **getExplicitUserAZs ():** Dado un functionId y calificador (uno no debe ser no nulo) devuelve las correspondientes autorizaciones de usuarios.
Parámetros: Id - $functionId, isActiveNowOnly
Retorna: object AuthorizationIterator

- **getExplicitUserAZsByFuncType ():** Dado un FunctionType y calificador devuelve las correspondientes autorizaciones de usuarios. Las autorizaciones deben ser para las funciones dentro de los FunctionType dado.
Parámetros: Type - $functionType, Id - $qualifierId, isActiveNowOnly
Retorna: object AuthorizationIterator

- **getExplicitUserAZsForImplicitAZ ():** Dado un implícito[46] devuelve las correspondientes autorizaciones explícitas del usuario. Autorizaciones explícitas pueden ser modificadas.
Parámetros: Authorization - $implicitAuthorization
Retorna: object AuthorizationIterator

- **getFunction ():** Devuelve todas las funciones.
Parámetros: Id - $functionId
Retorna: object Function

- **getFunctions ():** Devuelve todas las funciones dado un tipo específico.
Parámetros: Type - $functionType
Retorna: object FunctionIterator

- **getFunctionTypes ():** Obtener todos los FunctionTypes soportados por la implementación de autorización.
Parámetros: _
Retorna: object TypeIterator

- **getQualifier ():** Retorna los calificadores existentes
Parámetros: Id - $qualifierId
Retorna: object Qualifier

- **getQualifierChildren ():** Dado un qualifierId existente retorna los Ids de sus calificadores hijos.
Parámetros: Id - $qualifierId

---

[46] Una autorización que no fue creada utilizando AuthorizationManager y que tiene el agente, función, y calificador o calificador Descendiente (véase jerarquía calificadora) de una autorización explícita. La autorización implícita, a diferencia de la explícita, no se puede modificar directamente. Pueden ser modificadas sólo a través de una autorización explícita que contiene un ancestro calificador.

Retorna: object QualifierIterator

- **getQualifierDescendants ():** Dado un qualifierId existente retorna los Ids de todos los descendientes incluyendo los calificadores de su hijo.
Parámetros: Id - $qualifierId
Retorna: object QualifierIterator

- **getQualifierHierarchies ():** Devuelve las Qualifier Hierarchy [47] soportadas por la implementación de autorización. Las Qualifier Hierarchy son referenciadas por Id y pueden ser conocida y administradas a través de la Jerarquía OSID[48].
Parámetros: _
Retorna: object IdIterator

- **getQualifierTypes ():** Retorna todos los QualifierTypes soportados por la implementación de autorización.
Parámetros: _
Retorna: object TypeIterator

- **getRootQualifiers ():** Dado un hierarchyId, devuelve los calificadores de la raíz de la jerarquía especificada.
Parámetros: Id - $qualifierHierarchyId
Retorna: object QualifierIterator

[47] Una organización de los calificadores en los nodos de tal forma que todas las autorizaciones concedidas para un ancestro nodo se conceden a todos los nodos descendientes.

[48] Consultar www.okiproject.org

- **getWhoCanDo ():** Dada una functionId y un calificador devuelve el ID de todos los agentes autorizados a hacer la función con el calificador. Un calificador nulo es tratado como un comodín[49].
Parámetros: Id - $functionId
Retorna: object IdIterator

- **isAuthorized ():** Dado un agentId, functionId, y calificador devuelve verdadero si el agente está autorizado actualmente para realizar la función con el calificador.
Parámetros: Id - $agentId
Retorna: _

- **isUserAuthorized ():** Dada una functionId y calificador devuelve verdadero si el usuario está autorizado actualmente para realizar la función con el calificador.
Parámetros: Id - $functionId
Retorna: _

- **supportsDesign ():** Este método indica si esta implementación soporta los métodos AuthorizationManager: createFunction, deleteFunction. Métodos Función: updateDescription, updateDisplayName.
Parámetros: _
Retorna: _

- **supportsMaintenance ():** Este método indica si esta implementación soporta los métodos AuthorizationManager: createAuthorization, createDatedAuthorization, createQualifier, createRootQualifier, deleteAuthorization, deleteQualifier, getFunctionTypes, getQualifier, getQualifierChildren, getQualifierDescendents, getQualifierHierarchies, getQualifierTypes, getRootQualifiers, getWhoCanDo. Función métodos: getDescription, getDisplayName, getFunctionType, getId, getQualifierHierarchy. Calificador métodos: addParent, changeParent, getChildren,

[49] Puede ser cualquier calificador.

getDescription, getDisplayName, isParent, getId, getParents, getQualifierType, isChildOf, isDescendentOf, removeParent, updateDescription, updateDisplayName.
Parámetros: _
Retorna: _

2.2.3.3 Métodos de la OSID Repository.

- **copyAsset ():** Crear en un repositorio una copia de un Asset[50]. El Id, AssetType, y repositorio para el nuevo Asset se establece en función de la implementación.Todos los registros son copiados de manera similar.
Parámetros: Repository - $repository, Id - $assetId
Retorna: object Id

- **createRepository ():** Crea un nuevo repositorio del tipo especificado. La implementación de este método establece el Id para el nuevo objeto.
Parámetros: displayName, description, Type - $repositoryType
Retorna: object Repository

- **deleteRepository ():** Elimina el repositorio
Parámetros: Id - $repositoryId
Retorna: _

- **getAsset ():** Obtiene el Asset con el ID especificado.
Parámetros: Id - $assetId
Retorna: object Asset

- **getAssetByDate ():** Recibe el Asset con el Id especificado que es apropiado para la fecha especificada. La fecha especificada permite a una implementación de versiones soportar el versionado de Asset.

[50] Recurso que será almacenado en el repositorio.

Parámetros: Id - $assetId, date
Retorna: object Asset

- **getAssetDates ():** Obtener todas las fechas para el Asset con el Id especificado. Estas fechas permiten a una implementación de versiones soportar el versionado de Assets.
Parámetros: Id - $assetId
Retorna: object LongValueIterator

- **getAssetsBySearch ():** Realiza una búsqueda del tipo especificado y obtiene todos los Assets que satisfacen el SearchCriteria. La búsqueda se realiza para todos los repositorios especificados. Iteradores retornan un conjunto, uno a la vez.
Parámetros: Repository [] - $repositories, mixed - $searchCriteria (original type: java.io.Serializable), Type - $searchType, Properties - $searchProperties
Retorna: object AssetIterator

- **getRepositories ():** Obtiene los repositorios.
Parámetros: _
Retorna: object RepositoryIterator

- **getRepositoriesByType ():** Obtiene los repositorios del tipo especificado.
Parámetros: Type - $repositoryType
Retorna: object RepositoryIterator

- **getRepository ():** Obtiene el repositorio con el Id especificado.
Parámetros: Id - $repositoryId
Retorna: object Repository

- **getRepositoryTypes ():** Obtener todos los RepositoryTypes en este RepositoryManager. RepositoryTypes se utilizan para clasificar los repositorios. Iteradores retornan un conjunto, uno a la vez.

Parámetros: _
Retorna: object TypeIterator

En un entorno cambiante y diverso como el e-learning, las ventajas de un framework respecto a un producto son evidentes. Permite tener un patrón por el cual guiar todas las herramientas, logrando así una mayor homogeneidad de las aplicaciones y garantizar una mejor interoperabilidad entre las mismas. A continuación se describen los diferentes niveles de madurez que pudiera alcanzar el framework en su desarrollado.

2.3 Niveles de madurez del framework.

Para la definición de los niveles de madurez del framework que se propone, se tuvieron en cuenta los niveles de la arquitectura SOA explicados en el Capítulo 1.

2.3.1 Nivel 1 (Punto a Punto).

Para lograr un primer nivel, deben implementarse como servicios web las OSIDs authentication, authorization y repository de OKI, los 12 métodos del estándar SQI y la función básica enviar/almacenar IMS-DRI. Además, es necesario que los consumidores del servicio y quién lo publica acuerden un protocolo mediante el cual se defina cómo es invocado el mismo, cómo se pasan parámetros, cómo se reciben los resultados y cómo se manejan errores, entre otros. Para ello se propone XML y el protocolo SOAP.

XML será el lenguaje de etiquetado para describir los datos de los mensajes en formato de documento. SOAP se utilizará para el intercambio de mensajes basados en XML, utilizando http y https. Estos últimos son protocolos para solicitar o responder mensajes entre clientes y servicios, en el caso de https garantizando el envío seguro.

Por otro lado, los servicios deben permitir distribuir los detalles en un lenguaje neutro, proporcionando la información necesaria al cliente para que éste pueda interactuar con el servicio, para ello se utiliza WSDL, también basado en XML para describir las interfaces

públicas, protocolos y formato de mensajes, requeridos para interactuar con un servicio web. Todos estos aspectos son agrupados en un *schema*, por lo que para consumir un servicio lo primero que debe hacerse es localizar este schema. El documento WSDL está dividido en dos partes, una *concreta* que define el "cómo" y "dónde" y una *abstracta* que define qué hace el servicio a través de los mensajes que envía y recibe.

Una vez implementado el servicio, utilizando los estándares descritos, su publicación en este nivel se realiza localmente, por lo que al establecer una comunicación con una herramienta específica se debe conocer la dirección exacta de dónde se encuentra publicado cada uno de los servicios.

La mayoría de los sistemas que actualmente publican servicios se encuentran en este nivel, lo cual trae consigo algunas desventajas. Por ejemplo, en caso de cambiar la dirección de la aplicación que publica los servicios, las aplicaciones que estén consumiendo dichos servicios deberán ser actualizados, proceso que en ocasiones se hace engorroso, debido a que no siempre se tienen implementados los mecanismos de comunicación que garanticen la actualización continua de cambios. Esto puede provocar inconformidades entre los clientes que dependan de dichos servicios, además de no tener el control exacto del número de máquinas que se conectan a consumir los servicios.

2.3.2 Nivel 2 (UDDI/ESB).

El segundo nivel tiene como objetivo erradicar las deficiencias del primer nivel, logrando mayor madurez de interoperabilidad. Para ello se recomienda gestionar los servicios a través de ESB, aunque para llegar a este se puede desarrollar en dos etapas. En la primera los servicios se publican en una UDDI, y en la segunda se gestionan a través de un ESB.

Utilizar una UDDI permite descubrir el negocio (o servicio) adecuado, de los miles que pueden están registrados. Define cómo interactuar con el servicio escogido una vez

localizado. Describe los servicios y componentes o métodos de negocio de forma automática (o automatizable) en un entorno seguro, abierto y simple.

Como se ha planteado, una UDDI es un repositorio donde se encuentran los servicios publicados por diferentes aplicaciones. Esto posibilita que, en caso de cambiar la dirección donde se encuentra un servicio, sólo sea necesario cambiarla en la UDDI. Sin embargo, no se tiene un control de las máquinas que se conectan para consumir un determinado servicio.

Un ESB, por su parte, erradica la desventaja de la UDDI antes planteada, cumpliendo todas las expectativas de lograr una interoperabilidad eficiente y segura. Las ventajas del uso de ESB fueron expuestas en el Capítulo 1.

2.3.3 Nivel 3 (BPM).

El tercer nivel tiene como objetivo principal la administración de los procesos a través de los BPM, ofreciendo así una solución completa, que abarca todo el ciclo de vida de un proceso de negocio: análisis, modelación, ejecución y monitoreo de los procesos. En el Capítulo 1, se explica en qué consiste un BPM.

2.4 Pautas para el desarrollo del framework.

Para el desarrollo del framework se propone seguir las siguientes pautas:

- Diseñar herramientas interoperables que gestionen contenidos educativos, de modo que no existan herramientas aisladas.
- Utilizar estándares y protocolos Open Source de amplia aceptación por la comunidad educativa.
- Utilizar una arquitectura SOA, teniendo en cuenta los niveles de madurez propuestos.

- Promover la interoperabilidad entre diferentes aplicaciones, utilizando estándares de interoperabilidad como SQI, IMS-DRI y OKI.
- Promover que las aplicaciones sean creadas desde un principio con dicho framework, para mejorar la integración futura con otras herramientas.

En el Capítulo 1 se hace referencia a los principios básicos a tener en cuenta al publicar un servicio. Estos principios deben ser tenidos en cuenta para evitar situaciones que pueden presentarse en el uso de los servicios publicados.

Cuando se usan múltiples servicios para implementar un sistema, es muy fácil que la comunicación entre estos se salga de control. Por ejemplo, se puede tener un servicio que llama a otros servicios, algunos de los cuales llaman a otros servicios, y de esta manera, muy fácilmente el sistema se vuelve inmanejable. De esta forma, un sistema grande puede terminar con múltiples dependencias. Detectar un problema de rendimiento o funcionalidad se puede volver muy complicado. Según lo expresado, si no se cuenta con una estrategia adecuada, se puede llegar a una implementación donde exista una explosión de dependencias entre los diferentes servicios.

Una solución a este problema es extraer los aspectos de procedimiento de varios servicios dentro de uno dedicado, llamado servicio de negocio. Así, un servicio de negocio centraliza la definición del proceso, disminuyendo las dependencias entre servicios y las aplicaciones clientes, ayudando a su vez a facilitar la administración del sistema.

2.5 Trabajo Futuro.

En la primera iteración del framework, se tienen en cuenta los estándares más utilizados en el campo de la interoperabilidad como son IMS-DRI, OKI y SQI. El próximo paso será desarrollar este framework, y aplicarlo a las herramientas ROA, ROXS y Moodle, desarrolladas en el Polo Productivo Teleformación perteneciente a la Facultad 10, garantizando así la interoperabilidad entre las mismas. En esta iteración sólo se tienen en

cuenta las funciones básicas que garantizan la interoperabilidad entre sistemas e-learning.

Se recomienda, por tanto, una segunda iteración donde se incorporen las restantes OSID de OKI, a las que no se hace alusión en esta propuesta, y que pueden contribuir a la comunicación con otros sistemas como son los gestores académicos.

Por otro lado, este framework está basado en SOA, que es una arquitectura compleja, difícil de abarcar completamente en una primera aproximación. En la segunda iteración se debe profundizar en la calidad de los servicios en cuanto a políticas, seguridad, transacciones y administración, elementos que fueron explicados en el Capítulo 1 (ver además Anexo 2).

Una tercera iteración estaría enfocada al alcance de los niveles 4 y 5 de madurez.

2.6 Conclusiones del capítulo.

En este capítulo se explicaron los métodos que conforman los estándares SQI y OKI definiendo los parámetros y resultado que retorno cada uno de los métodos. Por otro lado se presenta la arquitectura del framework, presentando cada una de las capas y en qué consiste cada una de ellas. Además se definen los niveles de madurez del framework teniendo en cuenta los niveles propuestos para una arquitectura SOA expuestos en el Capítulo 1. Por último, se realiza una breve descripción de las posibles iteraciones futuras que se pueden realizar para perfeccionar el framework.

CONCLUSIONES

Se cumplieron los objetivos del trabajo, obteniéndose resultados aplicables a un ámbito mayor a lo que se tenía propuesto inicialmente. La propuesta no sólo abarca la gestión de contenidos educativos, sino también las herramientas para el e-learning.
Además:

- Se propuso la arquitectura de un framewok para la construcción de herramientas e-learning completamente interoperables.
- Se identificaron los estándares de interoperabilidad IMS-DRI, SQI y OKI como los más utilizados en el campo del e-learning.
- El uso de una arquitectura SOA como base del desarrollo de aplicaciones para el e-learning garantiza la independencia entre las diferentes capas.

RECOMENDACIONES

Para dar continuidad al trabajo se recomienda:

- Implementar el framewok y aplicarlo en la UCI para la integración de las herramientas ROA, ROXS y el LMS Moodle.
- Realizar las restantes iteraciones propuestas del framework.
- Mantener la vigilancia tecnológica sobre la evolución de los estándares utilizados.

BIBLIOGRAFÍA

Referencias Bibliográficas

(AulaGlobal, 2005) Aula Global: Observatorio de e-Learning, 2005.

(Aguirre .S, Salvachúa .J, Quemada J, et all, 2007) Quemada Salvachua Aguierre, J. J. S., Mediadores e Interoperabilidad en Elearning, 2007.

(Alfonso, 2007) Alfonso, J. ÁGORA VIRTUAL: una propuesta de entorno colaborativo y de enseñanza sobre interfaces OSID, 2007

(Alvez. P, Foti. P, Scalone. M et all, 2007) Scalone Foti Alvez , M. P. P., Estado Del Arte, 2007

(Booth. D, Haas. H, McCabe. F et all, 2007) McCabe Haas Booth, F. H. D., Arquitectura Orientada a Servicio y Web Services: Conceptos, Tecnologías y Herramientas, 2007

(Canabal. J & Sarasa. A, 2007) Sarasa Canabal, A. J., Agrega- Plataforma de Objetos Digitales Educativos, 2007

(Downes, 2001) Downes, S., Learning Objects: Resources For Distance Education., 2001

(Fernández. B, Moreno. B, Sierra. J, et all, 2007) Sierra Moreno Fernández, J. B. B., 2007 Uso de estándares aplicados a Tic en educación

(Gil, 2007) Martín Gil, M. A. Arquitectura Orientada a Servicios y los BPM, 2007.

(GONGO LABS, 2008) Preguntas que se hace un empresario ante la implantación de un BPM, 2008.

(Jardines, 2007) Jardines Méndez, J. B., Educación en red: mucho más que Educación a Distancia. Experiencia de las universidades médicas cubanas, 2007.

(López, 2005) López Guzmán, C., Los Repositorios de Objetos de Aprendizaje como soporte a un entorno e-Learning, Tesina doctoral, Universidad de Salamanca, 2005.

(Martínez, 2007) Martínez, M., Modelo de Madurez en Arquitectura Orientada a Servicios (SOA) 2007.

(Molist, 2006) Molist M., Institutos y universidades apuestan por la plataforma libre de e-learning, 2006.

(Márquez, 2007) Márquez J., Estado del arte del eLearning. Ideas para la definición de una plataforma universal, 2007.

(Potencier. F & Zaninotto. F, 2008) Zaninotto Potencier, F. F., Synfony la guía definitiva, 2008

(Red TTnet, 2005) Red TTnet. La formación sin distancia. Estudio realizado por el Grupo de Trabajo de e-Learning de la red TTnet, 2005.

(Tamayo, 2007) Tamayo Avila, D., Herramientas para la reutilización de contenidos educativos, 2007.

(Simon. B, Massart. D, Duval. E, 2004). Duval Massart Simon, E. D. M., Simple Query Interface Specification, 2004..

(Santanach. F, Casamajó. J, Casado. P, et all, 2007) Casado Casamajó Santanach, P. J. F., Una plataforma de integración, 2007

(Thorne & Kahn, 2006) Thorne Kahn S. J., O.K.I. Architectural Concepts, 2006.

(TIBCO, 2007) El papel de un bus de servicios empresariales (ESB) en una SOA, 2007.

(W3C, 2005) Arquitectura Orientada a Servicio y Web Services: Conceptos, Tecnologías y Herramientas, 2005.

Bibliografía Consultada

Aguirre .S, Salvachúa .J, Quemada J, et all, Mediadores e Interoperabilidad en Elearning, 2007.

Aula Global: Observatorio de e-Learning, 2005.

Alfonso, J. ÁGORA VIRTUAL: una propuesta de entorno colaborativo y de enseñanza sobre interfaces OSID, 2007

Alvez. P, Foti. P, Scalone. M**,** Estado Del Arte, 2007

Booth. D, Haas. H, McCabe. F, Arquitectura Orientada a Servicio y Web Services: Conceptos, Tecnologías y Herramientas, 2007

Baéz, F., Somos su empresa de Soporte a Desarrollo Informático. 2007.

Barruecos, Protocolo para la transmisión de contenidos en Internet, 2007.

Barchino. R, Otón. S, Ortiz. A, Arquitectura para publicación y localización universal de Objetos de Aprendizaje mediante Servicios Web.

Canabal. J & Sarasa. A, Sarasa Canabal, A. J., Agrega- Plataforma de Objetos Digitales Educativos, 2007.

Guzmán, C. L., La web y los sistemas e-learning, 2008.

Jardines J. Méndez B., Educación en red: mucho más que Educación a Distancia. Experiencia de las universidades médicas cubanas, 2007

.

López Guzmán, C. Los Repositorios de Objetos de Aprendizaje como soporte a un entorno e-Learning, Tesina doctoral, Universidad de Salamanca, 2005.

Martínez, M. Modelo de Madurez en Arquitectura Orientada a Servicios (SOA) 2007.

Molist M. Institutos y universidades apuestan por la plataforma libre de e-learning, 2006.

Pablo P. Alvez F., Marco Scalone Estado del Arte, 2006.

Pedruelo, M. R., El estándar SCORM para EaD, 2004.

Red TTnet: La formación sin distancia. Estudio realizado por el Grupo de Trabajo de e-Learning de la red TTnet, 2005.

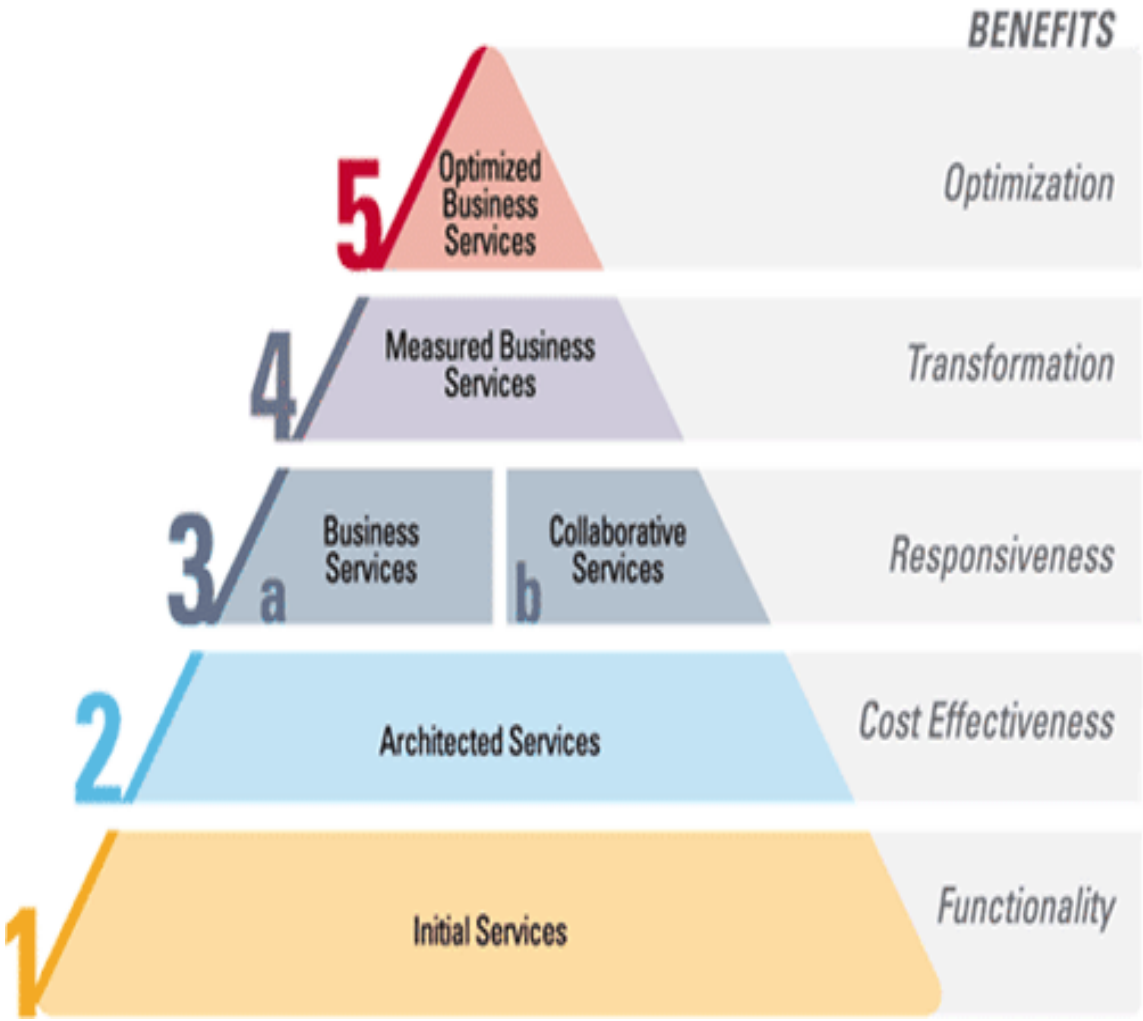
Simon. B, Massart. D, Duval. E**,** Simple Query Interface Specification, 2004.

Santanach. F, Casamajó. J, Casado. P, Una plataforma de integración, 2007.

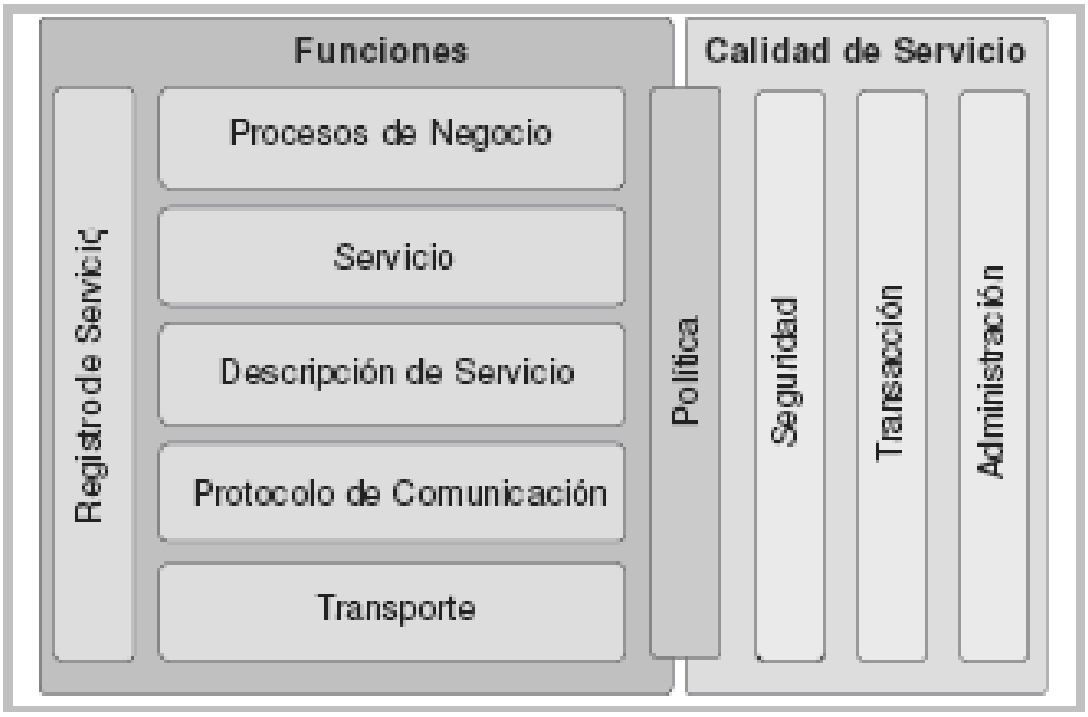
Torrija, A. L., Z39.50 en el siglo XXI: Estándar real o virtual, 2007

Thorne S. Kahn J., O.K.I. Architectural Concepts, 2006.

Anexo 1 Niveles de Madurez de SOA



Anexo 2 Elementos de una Arquitectura Orientada a Servicios



GLOSARIO DE TÉRMINO

E

E-learning: conjunto de tecnologías, aplicaciones y servicios orientados a facilitar la enseñanza y el aprendizaje a través de Internet/Intranet, que facilitan el acceso a la información y la comunicación con otros participantes. (Red TTnet, 2005)

Estándar: es un patrón, una tipificación o una norma de cómo realizar algo (AulaGlobal, 2007).

F

Framework: representando esencialmente una arquitectura de software referida a un dominio, de modo que modele las relaciones entre las entidades de dicho ámbito.

I

Interoperabilidad: la habilidad de dos o más sistemas o componentes para intercambiar información y para usar la información que ha sido intercambiada.(IEEE, 2007)

O

Objetos de aprendizaje: cualquier recurso con una intención formativa, compuesto de uno o varios elementos digitales, descrito con metadatos, que pueda ser utilizado y reutilizado dentro de un entorno e-Learning (López, 2005).