ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS DOCTORAL

PROPUESTA DE UNA ARQUITECTURA SOFTWARE BASADA EN SERVICIOS PARA LA IMPLEMENTACIÓN DE REPOSITORIOS DE OBJETOS DE APRENDIZAJE DISTRIBUIDOS

Autor : Salvador Otón Tortosa
Director: Dr. José Ramón Hilera Gonzalez

Alcalá de Henares, Julio 2006

AGRADECIMIENTOS

El autor quiere agradecer, en primer lugar, a su director de tesis su trabajo y dedicación.

Al equipo de investigación sobre e-learning del Departamento de Ciencias de la Computación por sus aportaciones y colaboración.

A Antonio Ortiz por su increíble trabajo en el desarrollo de SROA y a Isaac Gutiérrez por sus ideas y aportaciones al trabajo.

Por último, quiero agradecer muy especialmente a mis padres su apoyo constante en la realización de mis estudios y a Maica por su cariño y comprensión.

RESUMEN

El objetivo de esta tesis es proponer una arquitectura software para la construcción de un sistema que permita la localización de objetos de aprendizaje de forma universal para, de esta forma, poder integrarlos en un sistema de teleformación o *e-learning*. Los sistemas de aprendizaje están alcanzando en la actualidad una gran proliferación, como demuestran las cifras que pueden encontrarse en los trabajos publicados sobre la materia. Sin embargo estos sistemas evolucionan constantemente a la par que los estándares a los que tratan de adaptarse.

Los sistemas de aprendizaje utilizan objetos de aprendizaje (LO: *Learning Object*) como base del contenido de sus cursos. Estos objetos residen en repositorios, que consisten en almacenes digitales de recursos educativos que son accesibles a través de una red de comunicaciones. El objetivo de un repositorio es facilitar la reutilización de dichos recursos educativos, facilitando el acceso a los mismos.

Para que un objeto de aprendizaje sea reutilizado debe ser desarrollado de forma que se ajuste, al menos, a algún estándar de etiquetado de metadatos asociados a los contenidos que contenga.

En el estado actual de desarrollo, los sistemas de teleformación son ya herramientas muy perfeccionadas, pero aún están basadas en una arquitectura monolítica y rígida; y aunque utilizan Internet, tan sólo lo hacen como medio de comunicación de los contenidos, sin hacer uso de sus extensas posibilidades. En estos sistemas se integran contenidos almacenados en un repositorio (generalmente propietario) con las herramientas que ayudan a la docencia, y suele ser habitual que no accedan a repositorios externos de objetos docentes.

En resumen, estamos ante una situación en la que las plataformas tienen mayor sofisticación que los contenidos, pero en la que la arquitectura también constituye un serio freno a la evolución en términos de desarrollo y accesibilidad.

Se necesita, pues, una arquitectura realmente distribuida, en la que cada elemento constituya un activo capaz de interactuar con los demás. Esta arquitectura se sustentará, según veremos, en los metadatos asociados a los objetos docentes que los convierten en piezas fundamentales del sistema, y en protocolos que traerán consigo una modificación de las herramientas actuales.

La solución que proponemos es la definición de un marco funcional y arquitectónico para la adaptación de un sistema basado en SOA, e implementado en base a servicios Web (que proporciona un mecanismo flexible de integración de distintas aplicaciones y de descubrimiento de recursos en Internet), que asegure la interoperabilidad de distintos repositorios de objetos de aprendizaje y que favorezca la reutilización de los mismos.

En esta tesis, en primer lugar se analiza el estado actual de los sistemas de teleformación ó *e-learning*, sus propuestas, sus avances y, sobre todo, sus limitaciones. Dentro de este documento se hará un especial hincapié en el estudio de los repositorios que los sustentan y los estándares que indican como construirlos. A partir de dichos estudios trataremos de señalar las limitaciones existentes y de definir nuestras propuestas encaminadas a superarlas.

Se propone una arquitectura en niveles o capas que ha de satisfacer una serie de requisitos que deberán observarse como normas fundamentales a considerar en cualquier sistema que se base en dicha arquitectura. También se definen los componentes necesarios de la arquitectura para asegurar la funcionalidad requerida, así como el flujo de información y las relaciones entre ellos.

Para validar la Arquitectura propuesta se describirá un prototipo real, desarrollado utilizando las técnicas presentadas, creado a partir de los principios arquitecturales propuestos.

El documento finaliza con la exposición de las conclusiones y trabajos futuros relacionados con los temas tratados.

Se ha incluido un apartado con las fuentes documentales (incluidos enlaces de Internet) en las que nos hemos inspirado, sin propósito de exhaustividad, dado que este trabajo se enmarca dentro de un contexto sometido a cambios intensos y continuos.

SUMMARY

The objective of this thesis is a software architecture suggestion for the construction of a system which allows locating learning objects in a universal way so it will be possible to integrate them in learning or e-Learning system. At the present time the learning systems are reaching a great proliferation as it is demonstrated by the numbers that can be found in the published works about the subject. Nevertheless these systems develop at the same time that the standards to which they try to adapt.

The learning systems use learning objects (LO) as content base of their courses. These objects consist in repositories, which are digital stores of educative resources is a collection of resources (objects and/or units of learning) that is accessible through a communication network. The aim of a repository is to facilitate the reusability of those educative resources, making easier the access to them.

A learning object can be reused if it is made up in such a way that it is supported at least to some labelled standard associated to the contents that support.

In the present state of development, e-learning systems are already perfect tools but they are still based on a monolithic and strict architecture, and although they use Internet only do it as contents mass media without making use of their long possibilities. In these systems the stored contents in a repository (usually the owner) are integrated with the tools that help in teaching, and it is usually that they do not accede to external repositories of educational objects.

In summary, we are in the presence of a situation in which the platforms have greater sophistication than the contents, but in which the architecture also constitutes a serious brake to the evolution in development and accessibility terms.

It is needed, then, a really distributed architecture, in which each element constitutes assets able to interact with the others. This architecture will be sustained; as we will see, in the associated metadata to the educational objects that turn them in basic pieces of the system, and in protocols that will bring with them a modification of the present tools.

The proposed solution is the definition of a functional and architectonic frame for the adaptation of a system based on SOA and implemented through Web services (what provides a flexible integration mechanism of different applications and resources discovery in Internet) that assures the interoperability of different learning objects repositories and which assure their reusability.

In this Thesis, in the first place it is analyzed the present state of e-learning systems (also of learning based on the Web or Internet), their proposals, their advances and mainly their restrictions. Within this document it will be made a special emphasis in the repositories study that sustain them and the standards that point out how to construct them. From these studies we will indicate the present restrictions and we will define our proposals to solve them.

It is proposed a layers architecture of the system that includes a series of requirements that will have to be noticed like basic norms necessary to consider in every system based on this architecture. Also it is defined the needed components to assure the required functionality, as well as the information flow and the relations among them.

To validate the proposed Architecture it will be described a real prototype, which has been developed using the displayed techniques and which has been created from the proposed architectural principles.

This doctoral document ends with the presentation of conclusions and future works related to the treated subjects.

It has been also included a last point with the documentary sources (including Internet links) in which we have been inspired, without any intention of thoroughly in a referential frame subdue to intense and continuous changes.

ÍNDICE

1 INTRODUCCIÓN

El presente documento es la memoria de la tesis doctoral “Propuesta de una arquitectura software basada en servicios para la implementación de repositorios de objetos de aprendizaje distribuidos”, presentada por Salvador Otón Tortosa dentro del programa de Doctorado “Información, Documentación y Conocimiento” del Departamento de Ciencias de la Computación de la Universidad de Alcalá.

Este primer capítulo de la memoria tiene como finalidad introducir los conceptos previos relevantes para la Propuesta y Aplicación, así como describir el propio documento para facilitar su lectura.

1.1 JUSTIFICACIÓN DE LA INVESTIGACIÓN

Actualmente vivimos en una era en la que la información y el conocimiento implican el desarrollo de una sociedad globalizada, rompiendo fronteras y límites geográficos: es lo que se suele definir como la “sociedad de la información”. El propio Consejo Europeo en el año 2000 afirmaba que “la Unión Europea se enfrenta a un enorme cambio fruto de la mundialización y de los desafíos que plantea una nueva economía basada en el conocimiento” [CCE, 2000], y estableció para la Unión un objetivo estratégico esencial: “convertirse en la economía basada en el conocimiento más competitiva y dinámica del mundo”.

La información es el nuevo y preciado recurso al cual hay que proporcionar la capacidad de acceso de forma universal. Esta cantidad de información que caracteriza la sociedad moderna impone más que nunca la necesidad de aprender; sin embargo, un volumen importante de información a aprender y la velocidad requerida para hacerlo pueden llegar a generar desmoralización. Como señala Pozo [2002], “nunca ha habido

una época en la que hubiera tantas personas aprendiendo tantas cosas distintas a la vez, y también tantas personas dedicadas a hacer que otras personas aprendan. Estamos en la sociedad del aprendizaje“.

Además, y según escribe Rosenberg [2000], “el caso es que, al mismo tiempo, muchos de los modelos de aprendizaje tradicionales se están demostrando obsoletos e inapropiados ante esta situación”.

Tradicionalmente, los libros de texto han sido el mayor soporte para los contenidos educativos, el uso de libros ha definido la forma de aprendizaje en las escuelas de la sociedad moderna hasta nuestros días. Tampoco ha cambiado mucho la forma en que los profesores imparten sus clases, ya que se siguen utilizando medios muy parecidos desde hace siglos, como son la pizarra, la tiza, los libros, etc.

También la tradición cultural y el conocimiento de una sociedad residía en las bibliotecas, es decir, lugares que generalmente almacenaban documentos de distintos tipos normalmente textuales, otros gráficos e incluso audiovisuales. Cuando un profesor necesitaba material para preparar sus clases acudía a la biblioteca de su centro de enseñanza para consultarla.

Con la llegada de los ordenadores la información pasó a tener un formato digital y cuando se empezaron a construir las primeras redes de ordenadores esta información empezó a ser compartida por distintas personas que utilizaban la misma red. Con el paso del tiempo la innovación tecnológica ha conseguido que la difusión mundial del conocimiento y de la información sea algo natural. Este medio de comunicación es Internet, una inmensa cantidad de ordenadores interconectados entre sí y con una capacidad prácticamente ilimitada de almacenamiento de información, sin importar donde reside físicamente esa información.

Si el libro en soporte de papel y la imprenta supusieron un decisivo paso para la reducción de costes y acceso a la cultura de la mayor parte de la población, los nuevos instrumentos que aportan las telecomunicaciones y la informática acentúan la interactividad del texto, redescubren la creatividad de todo tipo de ilustración gráfica sin costes adicionales de reproducción, facilita el inmediato acceso en cualquier parte del mundo y hace innecesaria u opcional la reprografía. Hoy en día, los materiales multimedia y la utilización de las redes de comunicaciones introducen un nuevo elemento: la interactividad, que supone la clave del material educativo en la sociedad de la información.

Las tecnologías de la información están cambiando el acceso al conocimiento, los procesos de aprendizaje y los diferentes procedimientos establecidos dentro del campo de la educación y el adiestramiento. Tal y como afirma Martignago [1998] “la habilidad

de los enseñantes, la responsabilidad creativa de los alumnos y los recursos de conocimiento distribuidos en red son las difíciles claves para la revolución didáctica".

Para que la Educación sea más eficiente debe adaptarse a los requerimientos actuales. De hecho, se pone en duda que la educación tradicional basada en la relación maestro-alumno siga siendo válida por sí sola: la sociedad, cada vez más especializada requiere trabajadores con nuevas habilidades [Nasseh, 1996].

La adopción de las nuevas tecnologías por parte de los centros de enseñanza no deben quedarse en la mera compra de ordenadores y su conexión a Internet [Levy, 1993], sino que debe ir acompañado de una estrategia pedagógica.

Resumiendo la situación explicada en los párrafos anteriores, podemos pensar en cómo adquirimos conocimientos, habilidades y destrezas y cómo podemos generar y utilizar recursos docentes que nos mantengan actualizados en esta nueva sociedad de la información y el conocimiento. Para enfrentarnos a esta situación tenemos actualmente una serie de tecnologías utilizadas en el ámbito del e-learning que podemos resumir como sigue:

Los **objetos o unidades de aprendizaje** son la pieza clave en la construcción del material docente de forma que los contenidos educativos se fragmentan en unidades modulares independientes que pueden ser reutilizados en distintos entornos y en diferentes aplicaciones. Por tanto la labor de un creador de contenidos didácticos será elaborar correctamente "unidades de aprendizaje", esto es, "unidades mínimas en las que se puede organizar el material de formación para facilitar la gestión del conocimiento: creación, indexación, almacenamiento, distribución, uso, reutilización, evaluación y mejora de la formación" [Moreno y Bailly-Baillière, 2002].

Por lo tanto, el empleo de objetos de aprendizaje permite reutilizar los contenidos creados de una determinada experiencia educativa en contextos de aprendizaje diferentes. El proceso de construcción y distribución a los usuarios del material docente implica por tanto la creación, descubrimiento y agregación de unidades de aprendizaje simples en recursos educativos más complejos.

Para una completa reutilización de los objetos de aprendizaje es necesario ceñirse a los **estándares de e-learning**, en los que se han visto involucrados numerosas instituciones como IMS (*Instructional Management Systems Global Learning Consortium*), ADL (*Advanced Distributed Learning*) o IEEE. En estos estándares se hace especial hincapié en la necesidad de acompañar a los contenidos de los objetos de aprendizaje con un registro de metadatos que informe sobre el contenido del objeto y de su uso más apropiado [Sosteric y Hesemeier, 2002] y de su adecuación a unos determinados objetivos de aprendizaje.

Los llamados **Sistemas de Gestión del Aprendizaje** (*Learning Management System* – LMS) constituyen una categoría de software que automatiza la administración de acciones de formación: gestión de usuarios, gestión y control de cursos, gestión de los servicios de comunicación, etc. Estos sistemas gestionan los contenidos almacenados generalmente en repositorios. Aunque generalmente suelen ser meras adaptaciones de contenidos tradicionales, cumplen una labor muy productiva y beneficiosa en múltiples ámbitos; si bien presentan algunas carencias (poca formalización y estructuración del conocimiento, poca adaptabilidad a los estándares, contenidos demasiado expositivos, etc.) que los configuran como herramientas con limitaciones en su interacción con los alumnos.

Un **repositorio** o almacén digital de recursos educativos es una colección de recursos (objetos o unidades de aprendizaje) que son accesibles a través de una red de comunicaciones. No se necesita un conocimiento previo de la estructura de la colección, la cual puede contener los propios recursos o únicamente los metadatos que los describen, junto con una referencia para su localización [IMS, 2003a]. Por lo tanto, el objetivo de un repositorio es facilitar la reutilización de recursos educativos, facilitando el acceso a los recursos almacenados en el mismo. Los servicios que un repositorio de este tipo debe ofrecer al exterior, están relacionados con la búsqueda de objetos de aprendizaje a partir de metadatos, con el acceso a los objetos de aprendizaje localizados, o con el almacenamiento de objetos de aprendizaje en el repositorio.

Estas tecnologías son las que utilizan los sistemas de teleformación para la adquisición del conocimiento por parte del alumno; sin embargo, falta una pieza clave para que un sistema de este tipo sea o no útil: que haga un uso adecuado de las posibilidades de la Red.

Generalmente la gran mayoría de los sistemas actuales utilizan la red como un mero medio de comunicación. Sin embargo, si el sistema hace un uso avanzado de la red podremos utilizar sus posibilidades para desarrollar al máximo la interoperatividad entre sistemas de aprendizaje, la flexibilidad, la disponibilidad y la ubicuidad de los mismos; de forma que los objetos de aprendizaje que gestionan sean reutilizables, accesibles y adaptables. Este enfoque no implica necesariamente mayor complejidad en el software a desarrollar, al contrario, un sistema basado en esta filosofía debe ser sencillo para ser universal.

Parece, por tanto, oportuno intentar una síntesis entre los sistemas de teleformación y las posibilidades de la red, encaminada a una solución convergente para la construcción de una nueva generación de sistemas de teleformación distribuidos y en red donde la interoperabilidad entre sistemas y la reutilización de objetos de aprendizaje sean las piezas clave de su construcción.

1.2 OBJETIVOS Y MÉTODO DE TRABAJO

Así pues, estamos en condiciones de determinar, como punto de partida que justifique el desarrollo de este trabajo, la siguiente proposición:

> ***La investigación objeto de esta tesis se justifica en la creación de una arquitectura software orientada a servicios, e implementada mediante servicios Web, capaz de cubrir la necesidad de reutilizar objetos de aprendizaje almacenados en diferentes repositorios, que permitirá localizarlos independientemente de su ubicación física y de su tecnología de almacenamiento a través de Internet de manera que haga que los sistemas de teleformación sean interoperables.***

Un sistema construido en base a la arquitectura que vamos a proponer, y que podríamos denominar **Sistema de Reutilización de Objetos de Aprendizaje (SROA)**, dotaría a los sistemas tradicionales de docencia y aprendizaje a través de Internet de la flexibilidad y capacidad de poder publicar los objetos de aprendizaje de sus repositorios, y obtener objetos de otros repositorios externos de forma que la reutilización de estos objetos sea máxima y transparente para el usuario.

Podemos esperar los siguientes beneficios:

- Se propone una arquitectura abierta y fácil de adaptar a los sistemas actuales de teleformación ya que utiliza protocolos y formatos estándar para permitir la interoperabilidad e integración de dichos sistemas.

- Se ofrece la posibilidad de publicar y descubrir cualquier objeto de aprendizaje independientemente de su localización y los metadatos utilizados para su descripción didáctica.

- Se presenta una forma uniforme y bien definida de acceso a los objetos de aprendizaje.

- Se fomenta que los objetos de aprendizaje sean reutilizables.

Conviene, sin embargo, precisar una serie de principios que creemos fundamentales a la hora de desarrollar una arquitectura de estas características:

- La tecnología que nos proporciona Internet es la llave de una profunda revolución para la enseñanza. Pero la tecnología, cualquier tecnología, es una herramienta, no una estrategia.

- “La mejor tecnología no es aquella que reemplaza la realidad o la inteligencia con formas artificiales, sino aquella que hace aumentar las propias de cada uno” [Hodgins, 2000].

- Las clases presenciales continuarán teniendo un papel fundamental en la educación. “Quienes piensen que la tecnología reemplazará totalmente a los profesores al frente de una clase están tan equivocados como los que creen que Internet es una fiebre pasajera” [Rosenberg, 2001].

Nos proponemos avanzar conforme a los siguientes objetivos concretos:

1. Definir una arquitectura orientada a servicios para la localización universal de objetos de aprendizaje.

2. Describir cada uno de los servicios presentes en la arquitectura y su organización en la misma.

3. Estudiar las características de los repositorios existentes en la actualidad y su adaptabilidad a las especificaciones y estándares.

4. Facilitar la interoperabilidad de sistemas de teleformación aunque utilicen distintos estándares de metadatos de los objetos docentes que intercambian.

5. Ofrecer una capa de abstracción superior con servicios comunes disponibles para todos los sistemas que cubran sus necesidades de comunicación.

6. Basar la propuesta en estándares reconocidos y sólidos.

7. Construir un prototipo basado en la arquitectura propuesta.

Estos objetivos que nos proponemos alcanzar están en sintonía con las actuales investigaciones y trabajos sobre la utilización de los ordenadores y los sistemas de teleformación como ayuda a la docencia y el aprendizaje. Nuestra propuesta contribuye a la definición de un marco avanzado de dicha utilización. Así mismo, incluimos, y por supuesto seguimos, las recomendaciones de los principales grupos y comités tanto de estandarización como de implantación y uso de las nuevas tecnologías en el ámbito del aprendizaje.

Las actividades y el método de trabajo seguidos para conseguir los anteriores objetivos han sido las siguientes:

- Examen y análisis de arquitecturas existentes en el ámbito de los Sistemas de Gestión del Aprendizaje. De forma que la arquitectura propuesta se integre de la mejor forma, desde el punto de vista funcional y físico, con los sistemas existentes.

- Búsqueda y estudio de los diversos estándares existentes relacionados con la teleformación. Se estudiarán las diferentes propuestas y relaciones de los estándares más importantes. Fundamental en el desarrollo futuro de cualquier sistema de teleformación a través de Internet.

- Examen y análisis de los repositorios utilizados por los Sistemas de Gestión del Aprendizaje. Aquí estudiaremos las principales características de los repositorios desde su construcción hasta su utilización y veremos que problemas tienen los repositorios en Internet más utilizados.

- Definición de la arquitectura propuesta. Donde haremos una descripción detallada de cada una de las capas que la integran y de cada uno de los servicios utilizados en cada capa.

- Estudio de viabilidad de la propuesta mediante la implementación y prueba de un prototipo de la arquitectura que nos llevará a decidir si es factible su implantación.

- Por último plantearemos una serie de conclusiones a partir de la evaluación de la arquitectura así como una propuesta de futuras líneas de investigación.

1.3 ESTRUCTURA DEL DOCUMENTO

El documento en el que se recoge el resultado de la investigación llevada a cabo consta de seis capítulos y un anexo, estructurándose tal y como se describe a continuación.

En el **Capítulo 1**, que concluimos en este apartado, se incluye una introducción con la motivación y justificación de la investigación que se ha realizado, así como los objetivos y las actividades realizadas durante el desarrollo del trabajo para alcanzar los fines enunciados.

En el **Capítulo 2** desarrollamos un estudio sobre “el estado del arte” en el ámbito de la teleformación y las arquitecturas orientadas a servicios. En el apartado dedicado a la teleformación comenzaremos con una introducción histórica de la evolución de los sistemas de aprendizaje que tienen como soporte los ordenadores hasta llegar a los actuales Sistemas de Gestión del Conocimiento (LMS – *Learning Management System*), en los que nos detenemos con especial atención, sobre todo, para tratar de comprender cuales son sus principales características y sus limitaciones. Estudiamos también los estándares actuales dentro de la educación basada en Internet, ya que entendemos que cualquier propuesta que se haga hoy en día tiene que contemplar la interoperabilidad entre sistemas y la reutilización de los recursos educativos. A continuación se estudiarán las diversas tecnologías que hacen posible la definición de la arquitectura y su posterior construcción entre ellas destacan las arquitecturas orientadas a servicios y los servicios Web.

Iniciamos después lo que podríamos considerar la segunda, y más importante, parte de la memoria compuesta por los **Capítulos 3 y 4** en los que se plantea la problemática a resolver y la arquitectura que se propone en la tesis.

El **Capítulo 3** plantea la problemática a resolver, partiendo de un estudio de la evolución de los sistemas de aprendizaje para determinar donde encaja la arquitectura propuesta. A continuación se hace un estudio de las deficiencias encontradas hoy en día, tanto en los Sistemas de Gestión del Conocimiento como en los Sistemas de Gestión de Contenidos para llevar a cabo la reutilización de los objetos de aprendizaje que gestionan. Para ello examinaremos los principales repositorios que utilizan Internet para publicar sus objetos de aprendizaje de forma que podamos contrastar sus limitaciones.

En el **Capítulo 4** se expone la arquitectura propuesta, presentando las capas que la integran y los servicios que la forman, así como las relaciones entre ellos y el flujo de información.

En el **Capítulo 5** se estudia un caso práctico y real de aplicación de la arquitectura propuesta en un prototipo real. Se explicará la metodología de su construcción y la validación de la misma.

El **Capítulo 6** contiene las conclusiones de la investigación llevada a cabo junto con algunas líneas de investigación relacionadas con el tema en las que podría continuar el trabajo comenzado en esta tesis.

El **Capítulo 7** contiene las referencias bibliográficas citadas a lo largo de la tesis.

2 ESTADO DEL ARTE

En este capítulo efectuamos un estudio sobre “el estado del arte” de los sistemas de enseñanza y aprendizaje en relación con las nuevas tecnologías de información y comunicaciones. Comenzaremos con una introducción a los sistemas de gestión del aprendizaje y los estándares que lo sustentan tratando de comprender cuales son sus limitaciones. A continuación explicaremos las tecnologías asociadas al desarrollo de la arquitectura propuesta, básicamente servicios Web y SOA (*Service Oriented Architecture*). Finalmente, se comentarán otras propuestas de utilización de estas tecnologías en el ámbito del e-learning.

2.1 LOS SISTEMAS DE APRENDIZAJE BASADOS EN INTERNET

2.1.1 Evolución histórica

La educación es un dominio en el que históricamente se han intentado aprovechar las capacidades introducidas por el desarrollo de la tecnología, se podría decir que los cursos por correspondencia postal iniciados por Sir Isaac Pitman en 1840 [Horton, 2001] fueron los primeros, ya que constituyeron el primer intento de utilizar las comunicaciones para extender la formación más allá del sonido de la voz humana.

Muchos de los conceptos que forman la base de la teleformación estaban ya presentes entonces: el alumno podía marcar su propio ritmo de aprendizaje, éste podía llevarse a cabo sin contacto directo cara a cara, y un elevado número de estudiantes podían seguir el curso con una programación diferente.

En la década de los años 60 del pasado siglo, el uso de los ordenadores como soporte al proceso educativo y/o pedagógico propicia el nacimiento de las primeras (y primitivas) aplicaciones de educación asistida por computador, conocidas por el acrónimo CAI, que en inglés significa *Computer Assisted Instruction*, en la universidad de Stanford. Eran aplicaciones para el aprendizaje de la lectura y las matemáticas, que permitían individualizar dicho aprendizaje liberando al alumno de la asistencia a clase en grupos tradicionales. Estas aplicaciones poseían mecanismos de retro-alimentación a través de preguntas y respuestas de los propios alumnos, que permitían al mismo participar activamente en su propia formación [Molnar, 1990]. Los costes, tanto económicos como de tiempo de desarrollo, eran el mayor obstáculo para la expansión de estos métodos. Inicialmente se trataba de ordenadores *mainframe* programados en ensamblador o incluso lenguaje máquina, lo cual hacia larga y tediosa su programación, con lo que el uso de estos medios fue reducido y más bien experimental.

En los años 70 empieza a aparecer un conjunto de aplicaciones que hacen uso de técnicas avanzadas, como la Inteligencia Artificial o la Ciencia Cognitiva, y que suponen un gran adelanto en la forma en que se realizaba el aprendizaje asistido por computador [Brown et al., 1989]. El término más ampliamente aceptado para designar a estas aplicaciones es el de "sistemas de tutorización inteligentes" o "aplicaciones inteligentes de enseñanza asistida por computador" ("*Intelligent Computer Assisted Instruction*" -ICAI-). El objetivo era conseguir sistemas que literalmente "entendiesen" un dominio de conocimiento y fuesen capaces de orientar al alumno a la hora de articular sus propias ideas y estrategias de funcionamiento.

En la década de los 80, cuando los ordenadores comenzaron a ser habituales en los hogares y en las oficinas, se empezaron a implantar los primeros métodos de aprendizaje por medio de éstos de una forma amplia. Empresas como IBM ó Digital comenzaron a crear los primeros cursos para ser usados por medio de un ordenador. Estos cursos eran bastante sencillos, constaban fundamentalmente de texto y estaban distribuidos en forma de capítulos para ser seguidos de forma secuencial. Los cursos instruían sobre el manejo de aplicaciones de interés general, tales como procesadores de texto y hojas de cálculo; solían distribuirse junto con la aplicación relacionada y generalmente usando un disco magnético como soporte; con esto se conseguía que el futuro usuario de la aplicación pudiera aprender por sí mismo. Se trataba de cursos de autoestudio y prácticamente lo único que cambiaba con respecto a un libro era el soporte (de papel a soporte magnético). Esto fue el origen de lo que hoy se conoce como aprendizaje asistido por ordenador, *Computer Based Training* - CBT[1].

Seguidamente surgieron empresas particulares que se dedicaron de forma específica a crear materiales didácticos para ser utilizados por medio de ordenador. Generalmente

[1] Hay términos alternativos a éste que vienen a significar lo mismo, como son, el ya citado CAI o Computer Assisted Instruction, CBI o Computer Based Instruction y CAL o Computer Assisted Learning.

los temas tratados estaban relacionados con la propia Informática, como son los sistemas operativos, lenguajes de programación o uso de aplicaciones. Los soportes utilizados para la distribución fueron discos magnéticos y posteriormente discos ópticos como CD-ROMS, incluso llegando a hacer uso de redes de comunicación tanto locales como globales.

A lo largo de los años 90 aparecen y se extienden dos grupos de tecnologías que impactan profundamente en el mundo de la Informática y en los métodos de adiestramiento y enseñanza que usan el ordenador como punto central del proceso de aprendizaje:

- Las Tecnologías Multimedia.
- Las Redes de Comunicación.

2.1.1.1 Situación del e-learning hoy

Internet se nos ha presentado como el gran paradigma de los nuevos tiempos, su abrumadora y acelerada penetración en nuestras vidas ha decantado en lo que se conoce como la Sociedad de la Información y del Conocimiento. Los espacios educativos no han escapado a ello y han venido aplicando esta nueva tecnología de manera poco planificada y de una forma tal que muchos la ven como poco efectiva y deficiente.

Como nueva tecnología trata de ganar terreno al impulsar prácticas y procesos que le ayuden a superar un inicio poco claro. La combinación de la telecomunicación y la enseñanza ha permitido acuñar el término "*e-learning*" [Anido et al., 2002]. El e-learning, concepto adoptado para denominar el proceso enseñanza–aprendizaje que generalmente usa Internet en la educación, trata de formalizarse a través de métodos y herramientas de calidad. Los estándares e-learning están llamados a ser uno de los pilares fundamentales que ayudarán a gestionar con eficiencia uno de los activos más preciados de la denominada nueva economía: el conocimiento.

Este tipo de formación es realmente parte de lo que tradicionalmente se ha conocido como "Enseñanza Asistida por Ordenador" (EAO) si bien haciendo uso exhaustivo de las tecnologías de telecomunicación, por lo que el concepto de EAO está inevitablemente unido a la teleformación como así lo refleja la siguiente definición de Lanfranco [1993], "*la enseñanza asistida por computadora es la organización y combinación de los recursos educativos y tecnológicos para permitir el tele-aprendizaje por parte de los alumnos*".

El e-learning aporta a la educación la desaparición real de los problemas de espacio y de horarios. Los alumnos pueden realizar su aprendizaje desde cualquier sitio y a

cualquier hora. Los sistemas de enseñanza asistida por computador están disponibles 24 horas al día. Proporciona un canal de comunicación entre los propios alumnos, y entre éstos y los profesores. Hiltz y Norwood [1994] concluyen que la participación de los estudiantes puede llegar a ser superior en un entorno como éste que en un aula convencional. El canal de comunicación que se establece puede utilizarse con finalidades de seguimiento y tutorización de los alumnos por parte de los profesores. La información extraída de este seguimiento puede ser empleada posteriormente para labores de evaluación.

Dado que, habitualmente, el ámbito de actuación de estos sistemas es universal, los alumnos pueden elegir entre una gran diversidad de materias, cursos y especialidades. Éstos pueden ser preparados por los mejores especialistas en cada materia, para ser distribuidos a un conjunto amplio de estudiantes dispersos geográficamente. No sólo aquellos alumnos cercanos físicamente a ellos se benefician de éstos conocimientos.

El profundo cambio que ha supuesto Internet en el campo de las aplicaciones educativas no sólo afecta a aspectos puramente tecnológicos: también los paradigmas educativos se están alterando como consecuencia de las nuevas demandas sociales, en las que las redes globales de comunicación juegan un papel esencial. Actualmente se imponen los modelos educativos centrados en el alumno: ha crecido la demanda hacia una formación continua y “de por vida” en contraposición a los métodos educativos tradicionales en los que la formación se recibe en periodos determinados y centros específicos. Se trata de formar a alumnos con poco tiempo disponible y que necesitan obtener un elevado rendimiento.

Por otro lado, en éste análisis no podemos dejar de lado el componente principal de cualquier aplicación educativa: los contenidos educativos. Parece evidente que en última instancia, el éxito de una aplicación educativa radica en la calidad de sus contenidos. Por lo general, en la mayoría de las aplicaciones disponibles, los cursos se crean con el fin de cubrir una necesidad de aprendizaje concreta. Sin embargo, producir desde cero material educativo de alta calidad es una labor ardua que lleva mucho tiempo y requiere del conocimiento de diversos expertos en distintos campos.

La gestión del conocimiento se ha convertido en un tema recurrente en Informática. En este contexto, el conocimiento no sólo se identifica cómo un nuevo factor de producción de las sociedades industrializadas, sino cómo un producto en sí mismo. Para poder determinar su valor de cambio, el conocimiento debe ser creado, almacenado y gestionado, razón por la cual emergen las tecnologías de la información y comunicación cómo soporte a la gestión del conocimiento [Dodero, 2002].

En este sentido, como forma de compartir el conocimiento, una de las pretensiones en el ámbito del e-learning es la aplicación de **procedimientos que permitan la**

reutilización efectiva de material docente ya desarrollado, e idealmente, que faciliten dicha reutilización no sólo dentro del mismo entrono del e-learning para distintos cursos, sino entre aplicaciones diferentes con herramientas de creación de contenidos y plataformas distintas.

La tabla 2.1 (adaptada de Sigh [2001]) muestra una comparación entre los requisitos de los sistemas educativos tradicionales asistidos por ordenador (*Computer Based Training*) y los de las nuevas aplicaciones e-learning derivados de la utilización masiva de Internet como medio de distribución de información y comunicación, así como del cambio de mentalidad asociado a la adquisición del conocimiento y la formación en la sociedad actual.

ACTIVIDAD	ANTIGUOS SISTEMAS CBT	NUEVOS SISTEMAS E-LEARNING
Creación de contenidos	Realizado por el instructor. Los cursos se crean desde cero de principio a fin	Realizado por el instructor / diseñador del curso. Necesita conocimientos sobre la herramienta. Los cursos se crean recombinando material existente con nuevo material : incremento del valor del contenido
Distribución	Cara y complicada	Internet. Barato
Modelo educativo	Centrado en el instructor	Centrado en el alumno
Objetivo	Distribuir conocimiento	Distribuir y capturar conocimiento
Elemento o pieza de distribución / creación	Cursos completos	Módulos u Objetos Educativos (Learning Objects)
Actualizaciones	Reconstruir el curso y reenviarlo	Actualizar módulo
Velocidad	Depende del tamaño de la audiencia y de la extensión del contenido	Dependiente de la extensión el contenido y de la velocidad de la red de acceso
Tiempo típico	4 - 6 meses	4 - 6 semanas
Medidas de la efectividad	Observaciones del instructor	Sistemas de seguimiento y evaluación interactivos
Fuentes del conocimiento	Se crean todas	Buscar si existe material reutilizable y ensamblarlo (idealmente sin necesidad de adaptar cambios)

Tabla 2.1. Comparativa entre las aplicaciones tradicionales CBT y los sistemas e-learning

Teniendo en cuenta todo lo anterior, a partir de este punto del documento de tesis, consideramos lo siguiente:

> ***"Teleformación es la aplicación de Tecnologías de la Información y Comunicación para desarrollar, gestionar y distribuir cursos de formación."***

Cuando se habla de *"aplicación de Tecnologías de la Información y Comunicación"*, nos referimos en particular a las que se encuentren relacionadas con Internet. El término e-learning, no sólo cubre la distribución del curso, sino que también cubre el seguimiento, la programación, la gestión y otros aspectos del proceso de la enseñanza. Es decir, los sistemas de e-learning no sólo comprenden el contenido del curso, sino la plataforma tecnológica que lo distribuye y lo gestiona, y los servicios que soportan el mantenimiento. De hecho, las mayores compañías de e-learning no desarrollan el contenido, sino que se centran en las tecnologías (plataformas, gestión de desarrollo de contenidos, etc.) y servicios que permiten que el contenido sea eficazmente diseñado, distribuido y gestionado.

Se hacen necesarias, por tanto, nuevas tecnologías que faciliten y permitan llevar a cabo todos esos aspectos a través de Internet de una manera rápida y eficaz.

2.1.2 Sistemas de Gestión del Aprendizaje

Dentro del mundo del e-learning existe una inmensa cantidad de siglas, herramientas, arquitecturas, sistemas y tipos diferentes de aplicaciones que han aparecido en los últimos años; además, las diferencias y límites entre unas denominaciones y otras no están siempre claros. Antes de centrarnos en los Sistemas de Gestión del Aprendizaje propiamente dichos, vamos a intentar catalogarlos dentro de una clasificación que data del año 1998. Esta clasificación divide las nuevas herramientas para el desarrollo y distribución de materiales existentes en cinco categorías, que son las siguientes [Gram et al., 1998]:

1. **Herramientas de creación de contenidos**: Para la creación y edición de documentos HTML, PDF y otros formatos de texto, y la creación de gráficos (como Photoshop), animaciones, audio y video.
2. **Herramientas de creación de contenidos Web**: Aquellas ideadas para la edición basada en los formatos que se usan para ser publicados en Internet, como HTML. Es un subgrupo de la categoría anterior.
3. **Herramientas de comunicación**: Aplicaciones de *chats* de texto, video conferencia y otras de comunicación, tanto síncrona como asíncrona. Son ejemplo de este grupo programas como NetMeeting o Lotus Notes.
4. **Herramientas de autor para Internet**: Permiten la creación de contenidos y su publicación en la Web, incluso de forma simultanea. Se incluyen en este grupo herramientas como Authorware o Toolbook.
5. **Entornos educativos integrados distribuidos**: Enfocados a la distribución y gestión de materiales, a veces facilitan la administración de participantes, gestión de pruebas, etc. Son ejemplo de esta categoría entornos como WebCT o Learning Space.

Como ya hemos señalado este estudio fue realizado en el año 1998 y muchas de las herramientas consideradas han desaparecido o, si aún existen, han ampliado sus funcionalidades y características; además, han aparecido muchas nuevas. Sin embargo, esta aproximación es considerada todavía como válida a la hora de clasificar las herramientas existentes.

R.H. Jackson, de la Universidad de Tennessee, insiste en la necesidad de distinguir los entornos según sus funcionalidades, pero además incluye una nueva opción, clasificar dichos entornos según la forma en la que se realiza el aprendizaje. De esta manera Jackson ofrece dos clasificaciones de entornos de aprendizaje, según la forma y según la funcionalidad [Jackson, 2001]. Esta clasificación sólo tiene en cuenta entornos de aprendizaje diseñados para dicho fin, no incluye herramientas de creación de contenidos ni de comunicación.

Según la forma, Jackson clasifica los entornos en función del formato en que se realiza la enseñanza. Usando esta clasificación podemos determinar primero el formato que mejor se adecua a las necesidades del proceso y, después, seleccionar el entorno que más se ajuste a ellas.

Los entornos se clasifican en tres tipos, según el formato:

1. Estudio dirigido.
2. Eventos dirigidos por instructor.
3. Entornos colaborativos.

Un entorno de e-learning puede proveer más de uno de estos formatos y por tanto pertenecer a más de un grupo.

Estudio dirigido se refiere a aquel donde el alumno realiza su propio aprendizaje haciendo uso de tutoriales en CD-ROM, entorno *on-line* de aprendizaje, etc. Este estudio es complementado con interacción asíncrona con un instructor y con el resto de alumnos. El instructor guía o dirige al alumno en su aprendizaje y le resuelve dudas, aunque es el propio alumno quien se responsabiliza de aprender.

Eventos dirigidos por un instructor son aquellas actividades formativas síncronas donde existe una figura de profesor que imparte los conocimientos y dirige el aprendizaje. Permite un mejor ajuste del flujo de la enseñanza a las necesidades de los alumnos. Este método puede ser útil para complementar el estudio dirigido.

Finalmente, los entornos colaborativos son comunidades de aprendizaje. Proporcionan herramientas de comunicación síncronas y asíncronas para que pequeños

grupos de alumnos colaboren en sus estudios comunes, resolución de problemas, proyectos de investigación, etc. Puede existir un instructor o tutor que actué como soporte o guía.

Mientras que esta clasificación según la forma se centra en el aprendizaje, no tiene en cuenta las características técnicas; por eso Jackson propone una segunda clasificación basada en la funcionalidad de los entornos; bajo esta categoría se incluyen tres tipos:

1. ***Educational Delivery Systems*** (Sistemas de Distribución)
   Estos productos facilitan la publicación y distribución de contenidos *on-line*, no se centran en la creación de los mismos y no contienen mecanismos para medir el rendimiento o administrar recursos.
2. ***Learning Content Management Systems*** (LCMS) (Sistemas de Gestión de Contenidos)
   Estos entornos combinan la creación y distribución de materiales educacionales con mecanismos para medir el resultado y monitorizar el progreso de los estudiantes.
3. ***Learning Management Systems*** (LMS) (Sistemas de Gestión del Aprendizaje)
   Son similares a los LCMS pero dan a los estudiantes y a las organizaciones una visión integrada de todos los trabajos activos en múltiples cursos. Se centran más en la distribución de materiales y en el seguimiento y control de todos los elementos involucrados en el proceso educativo.

La diferencia entre los LCMS y los LMS es sutil. Los primeros permiten tanto la distribución como la creación de materiales, y pretenden que los educadores creen sus propios materiales *on-line*, mientras que los LMS se centran en la distribución y control de los materiales y no suelen incluir herramientas de creación de contenidos, o estas suelen ser muy simples; por tanto, los materiales deben ser creados externamente. Los LMS permiten siempre la creación de materiales o cursos completos exteriormente y después su integración al sistema. Con la implantación de los estándares, será incluso posible migrar los cursos de una plataforma a otra con todos sus contenidos.

2.1.2.1 Learning Management Systems

Podemos pues recapitular en una definición lo que entendemos por Sistema de Gestión del Aprendizaje (en inglés LMS "*Learning Management System*") como aquel sistema software que permite la distribución y gestión de conocimientos *on-line*. Un LMS permite realizar el aprendizaje habilitando métodos para ello, permitiendo la distribución de materiales docentes y contenidos. Además ofrece funcionalidades para gestionar y administrar todos los componentes que forman parte del proceso educativo.

La Web EduTools [EduTools, 2006] supervisada por WCET (*Western Cooperative for Educational Telecommunications*) se encarga de realizar una evaluación

independiente de los distintos proveedores de herramientas de e-learning del mercado, fijándose en sus principales funcionalidades, siendo algunas de ellas las siguientes:

- **Gestión e informe de los alumnos.** Permite a los administradores organizar a los participantes (otros administradores, educadores y estudiantes) en grupos lógicos, y monitorizar e informar sobre su progreso y actividades. Algunas utilidades de este tipo son:
  - Creación de registros de los alumnos y asignación de derechos de acceso.
  - Reparto de responsabilidades y funcionalidades entre varios administradores.
  - Creación de informes.
  - Mensajería.

- **Gestión de recursos y eventos de aprendizaje**. Permite a los administradores organizar los cursos y eventos en catálogos, al igual que gestionar todos los recursos que componen cada curso, incluyendo contenidos, instructores, 'aulas virtuales', etc. También permite comunicarse a estudiantes y administradores e informar sobre las actividades de los alumnos. Las principales funciones son:
  - Creación y gestión de recursos.
  - Distribución de derechos de acceso y del resto de permisos de los cursos.
  - Gestión de eventos, exámenes, foros de discusión, etc.

- **Distribución de cursos on-line**. Ofrece una infraestructura a los administradores para habilitar métodos de gestión del aprendizaje:
  - Medios síncronos, todos los participantes se reúnen a la misma hora y los eventos son dirigidos por un instructor.
  - Medios asíncronos, los materiales se encuentran a disposición de los alumnos y estos se responsabilizan de su propio aprendizaje.

  Se incluye también la creación del plan de estudios del curso y el calendario del mismo.

- **Creación y publicación de contenidos**. Los cursos pueden ser creados dentro del propio entorno o exteriormente al mismo:
  - Para cursos creados exteriormente, en un LMS:
    - Se permite la publicación de contenidos creados en formatos Web estándar.
    - Se exige ajustarse en lo posible a los estándares existentes.
  - En las herramientas de creación propias:

        - Se permite la integración de los sistemas de creación y publicación.
        - Se ofrece flexibilidad en las plantillas de creación de contenidos.

    Por otra parte se debe permitir importar y exportar cursos en un único paso.

- **Evaluación de conocimientos y habilidades.** Permite valorar las habilidades de los estudiantes, bien para determinar el curso o el plan de estudios que más se ajusta a sus necesidades o bien para saber el nivel de conocimientos adquiridos en el aprendizaje. Todo ello mediante:
    - La creación de tests y pruebas y su distribución y asignación temporal.
    - La generación de informes sobre la realización y resultados de las pruebas.

- **Gestión de bases de conocimiento.** En un LMS, debe integrarse un sistema específico de gestión de recursos accesible desde todos los cursos, que incluya contenidos de cualquier tipo en diversos formatos. También permite acceder a recursos externos como complemento a los cursos *on-line*.
  En todo caso, aunque muy publicitada últimamente, esta funcionalidad esta escasamente desarrollada y se detectan carencias de formalización, estructuración y conexión entre sus elementos.

- **Personalización del entorno centrada en el alumno.** Hace que el entorno reconozca al alumno y de esta manera le entregue sus mensajes y noticias, liste sus cursos, etc. También posibilita que los participantes personalicen el entorno en idioma, posición de los menús, etc. Esto en cuanto al alumno, pero además debe ser posible que la organización que instala un LMS personalice el entorno añadiendo su imagen corporativa, colores y demás elementos característicos; finalmente, debe existir un sistema de ayuda *on-line* sobre el funcionamiento del entorno.
  Estas personalizaciones se quedan en general en un nivel externo y de presentación, dándose muy raras veces el caso de que se profundice en funcionalidades de adaptación al alumno, no sólo en cuanto a contenidos, sino también a la manera de presentar estos contenidos.

Además de estos grupos de funcionalidades genéricamente implementadas, aunque en muy diverso grado, en los diferentes sistemas de gestión del aprendizaje existen otras características importantes que tratan aspectos menos funcionales, y sí más técnicos desde el punto de vista de la implementación:

- **Integración del LMS con otros sistemas.** A pesar de que los LMS ofrecen una amplia gama de posibilidades, puede resultar útil a algunas organizaciones la integración con otros sistemas, como otros LMS, bases de datos, entornos de creación de contenidos, etc. Estas integraciones suelen ser muy triviales en el sentido de que no van más allá del intercambio de datos poco estructurados, por ejemplo, los datos de los alumnos. En cualquier caso, a veces, proporcionan el medio a través del cual se pueden construir módulos que se integran con los LMS.

- **Seguridad del LMS.** El entorno debe mantener y guardar la seguridad de los participantes y los contenidos. Se debe asegurar que nadie sin permiso entre en el sistema, para ello se utilizarán identificadores de entrada y contraseñas.

- **Fiabilidad del LMS.** Si se pretende que los conocimientos estén siempre disponibles, el entorno debe de ser fiable, no debe fallar ni ante ataques externos ni ante errores internos.

Como señala Morrison [2004] un software LMS puede tener otras funcionalidades, como son:

- **Administración:** Debe permitir al administrador iniciar, dirigir, controlar y generar información de las clases, los estudiantes y los grupos. Un administrador puede configurar todas las opciones del sistema, controlar los costes de los cursos, establecer y modificar los campos definidos por los usuarios, especificar los requisitos de certificación y periodos de notificación (cuando el usuario deba ser avisado), gestionar los accesos, importar registros existentes y toda la información necesaria, añadir, borrar, modificar e importar toda la información de los cursos, controlar grupos de gestores, usar todas las herramientas de información, establecer las opciones de seguridad, etc.
- **Gestión "on-line":** Permitiendo a los profesores documentar, controlar e informar sobre las clases, los estudiantes y los grupos de estudiantes asignados a cada gestor "*on-line*", autorizado por el administrador. Un gestor "*on-line*" puede gestionar una serie de recursos asignados, como: cursos, estudiantes, grupos de estudiantes, control del uso de los cursos, ver e imprimir los informes de los estudiantes, monitorizar su progreso, asignarles cursos y currículos, añadir y modificar su información..
- **Gestión de estudiante "on-line":** Se trata de un usuario de una herramienta, a través de la cual, el estudiante registrado en el sistema puede acceder a los cursos, ver las tareas, y toda la información referente al curso. Un estudiante puede emplear su navegador Web para acceder a una interfaz, simple e intuitiva, donde ver el catálogo de cursos "*on-line*", ver información detallada

de los cursos, planificaciones, ejecutar informes personales, acceder a las lecciones de los cursos, ver su información personal, cancelar el registro, etc.

- **Conectividad e Interoperabilidad:** Son dos conceptos que han de estar muy presentes cuando un LMS puede ofertar, gestionar y medir las actividades de un curso: ejecución, control, puntuación, ejecutar cualquier tipo de recurso electrónico del curso (presentaciones, archivos, vídeos, etc.) desde el navegador, descargar cursos, etc. dado que todas estas actividades suelen necesitar que se intercambien ficheros.
- **Notificación vía e-mail automática:** Sirve para informar de los próximos cursos, sobre los cursos completados, sobre matrículas, clases y cursos superados, para el envío de documentos necesarios, etc.
- **Gestión de base de datos embebida:** En ocasiones se precisa de una base de datos para las necesidades de algunos clientes, esta es generalmente embebida, aunque cabe la posibilidad de emplear una base de datos externa al sistema.

De esta manera el LMS maneja todas las tareas administrativas del e-learning, lo cual resulta muy útil, puesto que dichas tareas pueden ser verdaderamente complicadas cuando se trabaja con centenares de cursos y miles de estudiantes.

Desde el punto de vista técnico, un LMS es un software basado en un sistema servidor, que controla el e-learning. Al acceder los usuarios normalmente vía navegador Web, el sistema es sencillo de manejar.

Existen tres tipos fundamentales de LMS [Piskurich, 2003]:

- **Propietarios:** Son sistemas muy herméticos, y generalmente con poca interoperabilidad con otros componentes e-learning.
- **Basados en estándares:** Aquellos sistemas que soportan los estándares e-learning, proporcionando interoperabilidad, pero que se ven limitados por las propias carencias de los estándares que soportan
- **Basados en sistemas de arquitecturas abiertas:** Proporcionan una interoperabilidad muy elevada gracias al conocimiento de su funcionamiento interno.

Como se aprecia en la figura 2.1 los principales componentes de un LMS típico según Henderson [2003] son:

- **Portal Web:** El sitio Web al que los usuarios acceden para aprender. Se puede considerar como la “puerta principal” del LMS, que conduce al catálogo de los cursos disponibles y a las demás partes del LMS. En ocasiones también incluye noticias, anuncios y otra información.

- **Catálogo de cursos:** Se trata de una descripción de cada uno de los cursos e-learning que están disponibles en el sistema. Los alumnos pueden seleccionar aquellos cursos que deseen cursar.
- **Gestión de los alumnos y registros:** Se trata de la parte administrativa del LMS y gestiona los registros de los alumnos, a la vez que proporciona informes a los administradores.
- **Evaluaciones (preguntas, test):** Muchos LMS incluyen la funcionalidad de realizar test sobre los cursos, algunos incluyen evaluaciones de temas concretos, en las que los estudiantes pueden realizar evaluaciones "*on-line*" de los conocimientos adquiridos y mediante las mismas, obtener recomendaciones de cursos que pueden hacer, para mejorar en aquellas áreas en las que sus resultados fueron bajos.

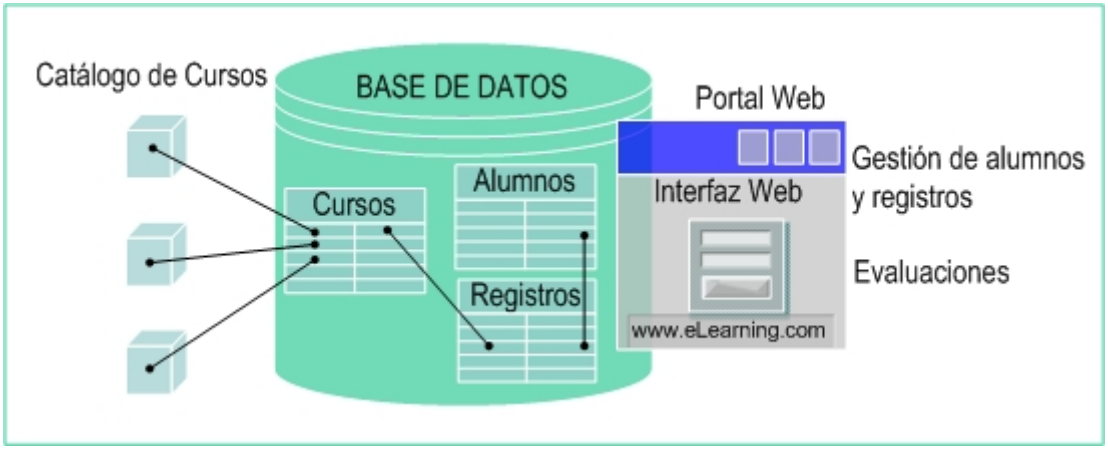

Figura 2.1. Esquema de componentes de un LMS

2.1.2.2 Learning Content Management Systems

La definición más sencilla de un sistema de gestión de contenidos (LCMS) es la que describe a éste como un sistema que permite la creación, almacenamiento, gestión y desarrollo de los contenidos e-learning bajo la forma de objetos de aprendizaje, para servir las necesidades de los individuos.

Los sistemas de gestión de contenidos, en adelante LCMS, aparecen por dos razones fundamentales: primero por la necesidad de algunos usuarios de LMS que precisaban crear y manipular contenidos, y en segundo lugar, por el rápido incremento de las prestaciones que se le exigían al sistema y para las cuales los LMS's no tenían la respuesta adecuada.

Los LMS y LCMS tienen una naturaleza complementaria, los segundos no fueron diseñados para reemplazar a los primeros, sino para proporcionar a los clientes que así lo requerían, la capacidad de gestionar también los contenidos de aprendizaje. Los

LCMS facilitan a los profesores la división de los cursos en unidades cada vez más pequeñas (granulación), y de esta manera fomentan la reutilización de contenidos y simplifican la creación de los cursos.

La siguiente tabla muestra cómo los resultados establecen las diferentes funcionalidades de los dos tipos de sistemas:

	LMS	LCMS
Usuarios principales	Gestores del aprendizaje, Profesores, Administradores.	Desarrolladores de contenidos y diseñadores de contenidos educativos
Provee herramientas para la gestión primaria de...	Estudiantes	Contenidos educativos
Realiza gestión de las clases, del aprendizaje...	Si. Pero no siempre	No
Realización de informes del aprendizaje...	Es una de las tareas principales.	Es una tarea secundaria.
Colaboración con el estudiante	Si	Si
Guarda datos de los perfiles de los estudiantes	Si	No
Comparte los datos de los usuarios con otros sistemas	Si	No
Planificación de eventos	Si	No
Control de competencias, análisis de las habilidades	Si	Sí, en algunos casos
Capacidades de creación de contenidos	No	Si
Organización de contenidos reutilizables	No	Si
Creación de preguntas de evaluación y gestión de test	Si, un 73% de los LMS tienen ésta capacidad.	Si, un 92% de los LCMS tienen ésta capacidad
Pre-evaluación dinámica y enseñanza adaptable	No	Si
Herramientas de gestión de aprendizaje para el proceso de desarrollo de contenidos	No	Si
Entrega de los contenidos, proporcionando controles de navegación e interfaces para el alumno	No	Si

Tabla 2.2. Comparación de las funcionalidades de un LMS y un LCMS

Habitualmente los LCMS proporcionan módulos para la creación de contenidos, la creación de evaluaciones, publicación, administración, repositorio de datos, y un módulo para la personalización del entorno de trabajo.

El LCMS simplifica y acelera el proceso de creación de contenidos facilitando a expertos en cualquier materia, pero no necesariamente expertos en el desarrollo de software, las funciones para diseñar, crear, distribuir y controlar la calidad del proceso de aprendizaje de una forma sencilla, rápida y eficiente. Es debido a esta funcionalidad por lo que se puede pensar que los LCMS disponen de potencial suficiente para emplearse como herramienta autónoma capaz de compartir y gestionar los conocimientos, reemplazando así, en cierto modo, al LMS. Pero los LMS y LCMS no son excluyentes, sino complementarios, han de trabajar conjuntamente para así intercambiar entre ambos la información, resultando un aprendizaje más enriquecedor para el alumno y una herramienta más completa para el administrador del e-learning. El LMS gestiona los grupos de usuarios, permitiendo a cada uno de ellos el acceso a los objetos apropiados almacenados y gestionados por el LCMS. A su vez, el LCMS también guarda los progresos individuales de cada alumno y sus notas en las evaluaciones, pasándole los datos al LMS para realizar informes.

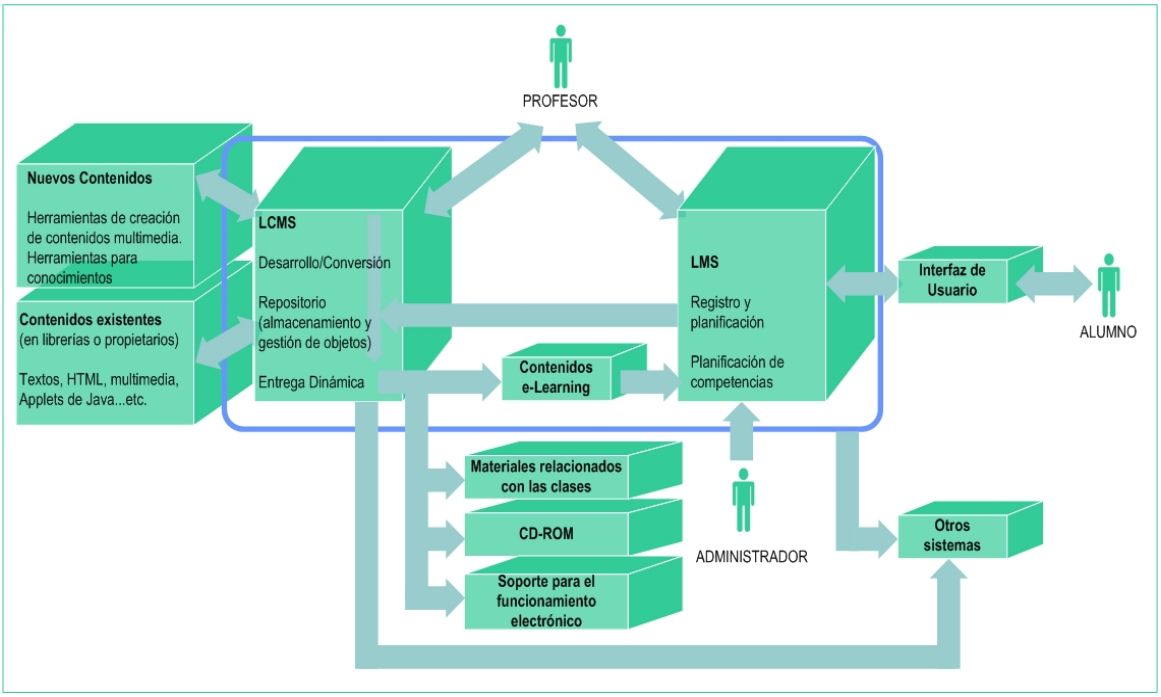

Figura 2.2. Integración entre LMS y LCMS

Los LCMS se construyen en torno a una base de datos central en la que se almacenan los elementos que constituyen los contenidos de los cursos; dichos elementos se presentan en formatos portables (reutilizables), y generalmente incluyen herramientas de creación y publicación de contenidos. Estos sistemas, que funcionan en un entorno Web, pueden crearse con una estructura y apariencia diferentes unos de otros; además, sus contenidos pueden ser fácilmente actualizados por usuarios sin conocimientos especiales en aspectos técnicos y de mantenimiento de entornos Web.

Dichos contenidos pueden ser usados por más de un sitio Web, de manera que se reduzca el esfuerzo del desarrollo, al poder reutilizar los mismos contenidos para diferentes cursos. Son estas posibilidades las que han popularizado el uso de las plataformas e-learning entre los profesores que han necesitado poner material de los cursos a disposición de los estudiantes (a distancia, por el método tradicional, o ambos tipos) de una manera rápida y fácil.

El siguiente diagrama de la figura 2.3 muestra algunos de los componentes de un LCMS y como se relacionan unos con otros.

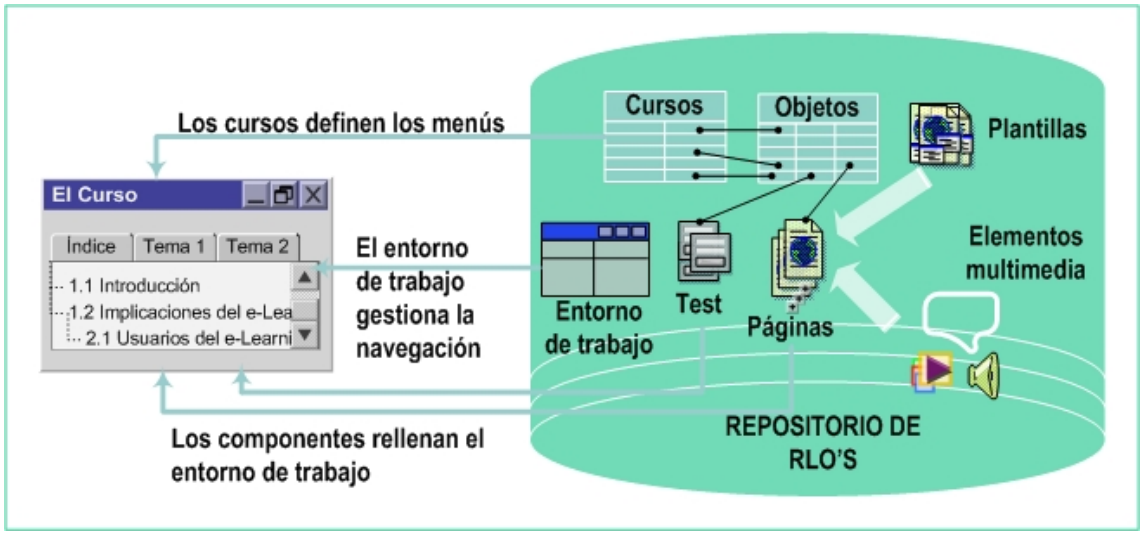

Figura 2.3. Componentes de un LCMS

Los creadores de contenidos crean los elementos multimedia, las evaluaciones y todos los componentes necesarios para la creación de un curso y los almacenan en el repositorio.

Sirviéndose de plantillas ya existentes, o comenzando desde cero, se agrupan los componentes almacenados que se precisen para responder a un requerimiento de un usuario, esos grupos de componentes se etiquetan en términos de los objetivos buscados y de los contenidos que se necesitan para alcanzar dichos objetivos, constituyendo así la materia de los cursos. Las lecciones y los cursos se definen en términos de los objetos de aprendizaje que contienen.

Los creadores de contenidos deben también definir el entorno de trabajo, para controlar la navegación y proporcionar la interfaz de usuario del curso. Cuando se necesita un curso se realiza una copia del entorno de trabajo; mediante la definición del curso, se genera el menú correspondiente y los estudiantes seleccionan de dicho menú las páginas y cualesquiera otros componentes del curso, disponibles en el entorno de trabajo.

Por lo tanto, una plataforma LCMS, además de garantizar el control del proceso de aprendizaje, debe facilitar la creación, almacenamiento y reparto de los contenidos, ateniéndose a las siguientes características:

- Proporcionar herramientas sencillas que faciliten la creación de contenidos, en forma de aplicaciones o software de autor embebidas en el sistema, incluyendo editores para la creación de contenido Web, con el fin de eliminar la necesidad de manejar editores HTML o XML.
- Emplear sistemas flexibles de diseño y distribución de los recursos, que permitan adaptarse a las necesidades de la organización y a los diferentes sistemas y ritmos de aprendizaje de los usuarios.
- Posibilitar la reutilización de los objetos de aprendizaje; de hecho, cada elemento de aprendizaje debiera ser tratado como un objeto de aprendizaje reutilizable y mantenido a disposición de los profesores, quienes pueden aprovecharse de esta facilidad para emplear el mismo objeto en cursos diferentes, de manera que se minimice el esfuerzo de desarrollo, o actualización.
- Proporcionar herramientas para la administración del sistema que permita las matriculaciones, el seguimiento del aprendizaje, controlar el uso de los tiempos, el seguimiento de los usuarios, la adecuación de contenidos, etc.
- Proporcionar herramientas para la evaluación, tanto inicial como de la evolución de los aprendizajes que se produzca a lo largo del curso, ya sea en lo que se refiere a los cursos en general, como a los objetos de aprendizaje en particular; el sistema debe proveer recursos suficientes para valorar los aprendizajes bajo distintos niveles de dificultad y diferentes modalidades de medición.
- Permitir la conectividad con otros LMS y la adecuación a los estándares más importantes.
- Disponer de herramientas para la comunicación y el aprendizaje en colaboración. Incluyendo recursos, tanto síncronos como asíncronos, que faciliten la comunicación sencilla entre iguales y con el profesorado; y, asimismo, recursos para el aprendizaje en colaboración que permita compartir conocimiento y realizar trabajos en grupo.
- Establecer mecanismos de seguridad y protección del conocimiento almacenado; dicha seguridad dependerá del uso de los privilegios de los diferentes usuarios y de las diversas funciones que los mismos desarrollan dentro de la organización, y afectará a las cargas y descargas de documentación, así como al acceso a la misma.
- Simplificar la migración de contenidos para facilitar la adaptación a las diferentes necesidades y escenarios de formación que se puedan presentar.

- Facilitar el proceso de instalación de manera que haga innecesarias las adaptaciones, localizaciones, personalización y demás tareas que encarecen el producto y retrasan ese proceso.

Viendo las características y funcionalidades de cada sistema, se puede decir que la principal diferencia de los LMS con respecto a los LCMS, radica en que para los primeros el elemento con el que trabajan es la totalidad de un curso ya existente, sin entrar en el detalle de los contenidos que constituyen dicho curso; mientras que para un LCMS su elemento de trabajo son dichos contenidos, vistos como componentes aislados de cursos, sea de cursos ya existentes o de cursos que podrán crearse en un futuro.

La necesidad de respetar esas características y proporcionar esas funcionalidades de forma satisfactoria en términos de eficiencia y seguridad, obligó a los desarrolladores a potenciar el uso generalizado de técnicas sin las cuales puede decirse que el e-learning no alcanza los objetivos para los que ha sido concebido.

Ya se ha hecho referencia, en páginas anteriores, a el cumplimiento de estándares universales como condición necesaria para garantizar el éxito de una aplicación e-learning; igualmente hay otros dos aspectos que es imprescindible tomar en consideración cuando se estudia una plataforma, se trata de la capacidad de reutilización de los Objetos de Aprendizaje y la conveniencia de su almacenamiento en repositorios públicos.

Satisfacer adecuadamente las necesidades de acceso, distribución y creación de contenidos de cursos e-learning, tal y como se viene planteando en esta tesis, supone el empleo de Objetos de Aprendizaje Reutilizables y de Repositorios. Dada la importancia que esos conceptos tienen en la actividad de desarrollo de plataformas e-learning, se exponen seguidamente algunas ideas básicas para facilitar su comprensión.

2.1.2.3 Objetos de Aprendizaje Reutilizables

El enfoque de Objeto de Aprendizaje Reutilizable ("*Reusable Learning Objects*" o RLO) vino impuesto por la práctica de los desarrolladores de cursos, quienes constataron que, frecuentemente, existía una considerable redundancia, o solapamiento, en el contenido de los cursos, pese a que éstos iban dirigidos a usuarios con perfiles diferentes y con diferentes objetivos de aprendizaje. Esa redundancia suponía una multiplicación del esfuerzo de desarrollo y encarecía considerablemente los productos finales, a la par que prolongaba los tiempos de desarrollo y dificultaba los objetivos de intercomunicación, migración, escalabilidad, actualización, etc.

Al definir los Objetos de Aprendizaje Reutilizables como los componentes elementales en que puede descomponerse el contenido "formacional" de un curso, se está hablando de la fragmentación de contenidos (granulación) en "pedazos autónomos de material de aprendizaje" [Patron, 2003]. El grado que se alcance en esa fragmentación o granulación de contenidos (respetando siempre la exigencia de autonomía aludida por Patron), indicará, a su vez, el grado de la posibilidad de reutilización de dichos objetos. Tanto mayor será ese grado de reutilización, tanto mayor será la eficacia del desarrollo y su idoneidad para satisfacer las necesidades exigidas al sistema.

La fragmentación de los contenidos trajo como consecuencia la multiplicación de los RLO's que era necesario manejar; a título de ejemplo de este hecho, se suele emplear el juego de construcción Lego, donde puede apreciarse la diferencia entre construir un castillo como un todo (prácticamente una sola pieza), y hacerlo mediante las piezas del Lego, en el que es necesario manejar un número considerablemente mayor de piezas elementales; en contrapartida, con las piezas elementales del Lego se pueden hacer castillos de diferentes formas, o incluso otro tipo de construcciones que nada tengan que ver con un castillo. Una postura razonable en el grado de fragmentación de los contenidos, ha de buscar el equilibrio entre las ventajas obtenidas por la granulación y las dificultades de manejar un número muy elevado de RLO's.

La dificultad de manejo de los RLO's, principalmente en lo que atañe a su acceso y ubicación, obligó a la creación, para cada RLO, de metadatos o metainformación, asociado indisolublemente al objeto, por lo que puede decirse que un RLO está compuesto de dos partes: por un lado el contenido del objeto consistente en la información didáctica que éste aportará al curso en que habrá de integrarse; y por otro lado la meta-información que proporciona datos referentes a lo que el objeto de aprendizaje encierra en sí mismo, y en ella figuran por ejemplo: título, palabras clave relativas al contenido, a los objetivos del mismo, así como nivel, prerrequisitos, evaluación, autor, fecha, lenguaje, versión etc. La meta-información se trata, a su vez, como un elemento independiente que puede ser ejecutado en una plataforma Web estándar o sistema operativo, no necesita de "*plug-ins*" o de la instalación de aplicaciones especiales. La razón de ser de la meta-información es permitir la localización de los objetos de aprendizaje para integrarlos en un curso determinado, eso se consigue gracias a la información, estructurada en una serie de campos, que se obtiene de la meta-información.

2.1.2.4 Repositorios digitales

Para muchos autores la idea de repositorio es intrínseca a los objetos de aprendizaje. No es posible pensar en objetos de aprendizaje si no se los concibe albergados en repositorios.

Los repositorios digitales, en el sentido más amplio de la definición, se emplean para almacenar cualquier tipo de material digital. No obstante, los repositorios digitales para objetos de aprendizaje son mucho más complejos en términos de qué es necesario almacenar y cómo se almacenará. El término “objeto de aprendizaje” se refiere a cualquier elemento digital que puede ser usado para permitir el aprendizaje o la enseñanza. Se puede tratar de cualquier tipo de objeto, como: gráficos, imágenes simples o videos, pasando por documentos, exámenes complejos o agrupaciones de objetos. Cada elemento ha de tener su propia identidad y ser localizable, por ello, los criterios de búsqueda de esos objetos han de considerar bastante más que títulos, autores o palabras clave.

De hecho, un examen de la granulación de los objetos de aprendizaje que se almacenan en un repositorio digital es un buen punto de inicio para determinar la complejidad del manejo de dicho repositorio.

Si los objetos son cursos completos, o módulos substanciales de los cursos, entonces el repositorio no es más que un portal, desde el cual acceder al material. Lo que hace de los repositorios digitales de objetos de aprendizaje algo más que un simple portal, es la capacidad de encontrar los objetos de aprendizaje y darles nuevos usos. El propósito de un repositorio digital de objetos de aprendizaje no es simplemente almacenar y distribuir dichos objetos, sino permitir que los mismos sean compartidos por distintos estudiantes y, sobre todo, facilitar su reutilización en diferentes actividades formativas. La ventaja de los repositorios desde el punto de vista de los usuarios es el hecho de tener acceso a los contenidos depositados en el repositorio/almacén; y para usar esos servicios ha de pagar el coste de someterse a unos procedimientos, normas, y estándares, cuya aplicación va dirigida a potenciar uno de los aspectos más interesantes y fructíferos del repositorio, como es la creación de objetos de aprendizaje reutilizables.

Se puede argumentar que un repositorio digital es una biblioteca digital en la que se realizan intercambios entre sus contenidos. Pero tal argumentación no es del todo correcta, ya que los gestores de una biblioteca digital tienen como principal responsabilidad establecer dónde colocar los objetos, para que el usuario pueda tener acceso a los mismos; mientras que los repositorios han de ser vistos, principalmente, como almacenes de objetos que son puestos a disposición de diferentes usuarios, en el momento y lugar que éstos los soliciten, permitiendo asimismo que esos usuarios no

sólo se sirvan de la información contenida en dichos objetos, integrándola en un curso determinado, sino que puedan actuar sobre ella modificándola, ampliándola, incorporándole mejoras; actividades todas ellas que, al estar sometidas a las normas y estándares a las que antes se aludió, permiten obtener e incorporar al repositorio objetos de aprendizaje con diferentes contenidos, pero de características homogéneas en cuanto a su composición y estructura, lo cual permite su reutilización cuantas veces sea preciso de forma cómoda y sencilla.

El empleo del término repositorio con preferencia al de biblioteca digital, tiene como objetivo enfatizar el hecho de que, en el primero, al permitir la aportación de mejoras por parte de los usuarios, se consigue que diferentes personas puedan compartir los objetos de aprendizaje para la creación de cursos que, desde el punto de vista de la formación proporcionada, tengan objetivos diferentes.

Existen dos tipos de repositorios digitales de objetos de aprendizaje, según Downes [2002]:

- Aquellos que contienen tanto los objetos de aprendizaje con su contenido de información, como los metadatos de dichos objetos de aprendizaje.
- Aquellos que contienen sólo los metadatos de los objetos de aprendizaje, mientras que los objetos de aprendizaje con su contenido de información se encuentran almacenados en otra ubicación en la que el repositorio puede localizarlos, a partir de la información contenida en los metadatos y mediante el empleo de una herramienta adecuada para ello.

La mayor parte de los repositorios suelen ser autónomos. Esto es, funcionan como portales a los que se accede mediante una interfaz basada en Web, proporcionando un mecanismo de búsqueda y una lista de categorías donde buscar, sin necesidad de emplear otro producto. Algunos otros repositorios digitales de objetos de aprendizaje funcionan como una base de datos conectada a otro producto (generalmente una plataforma).

Usos de repositorios digitales

Para definir los posibles usos de los repositorios digitales es útil considerar primero quien los usará y qué procesos educativos se beneficiarán de ellos. Los usuarios de un repositorio digital de objetos de aprendizaje son generalmente, aunque no siempre, los profesores, ya que normalmente son los encargados de producir los cursos. El repositorio digital no será afectado por el uso pedagógico del material y actuará simplemente como un almacén en el que no tiene influencia el dónde o quién emplea los recursos almacenados, o el propósito con que estos son empleados.

Aquellos que usan repositorios digitales no deben preocuparse de la arquitectura interna de los mismos. Es útil, sin embargo, seguir las especificaciones de "*IMS Digital Repository Interoperability Working Group*" (DRIWG) [IMS, 2003a] que permiten a los repositorios digitales interactuar entre sí ignorando su arquitectura interna. En dichas especificaciones se definen los repositorios digitales en términos de lo que pueden contener.

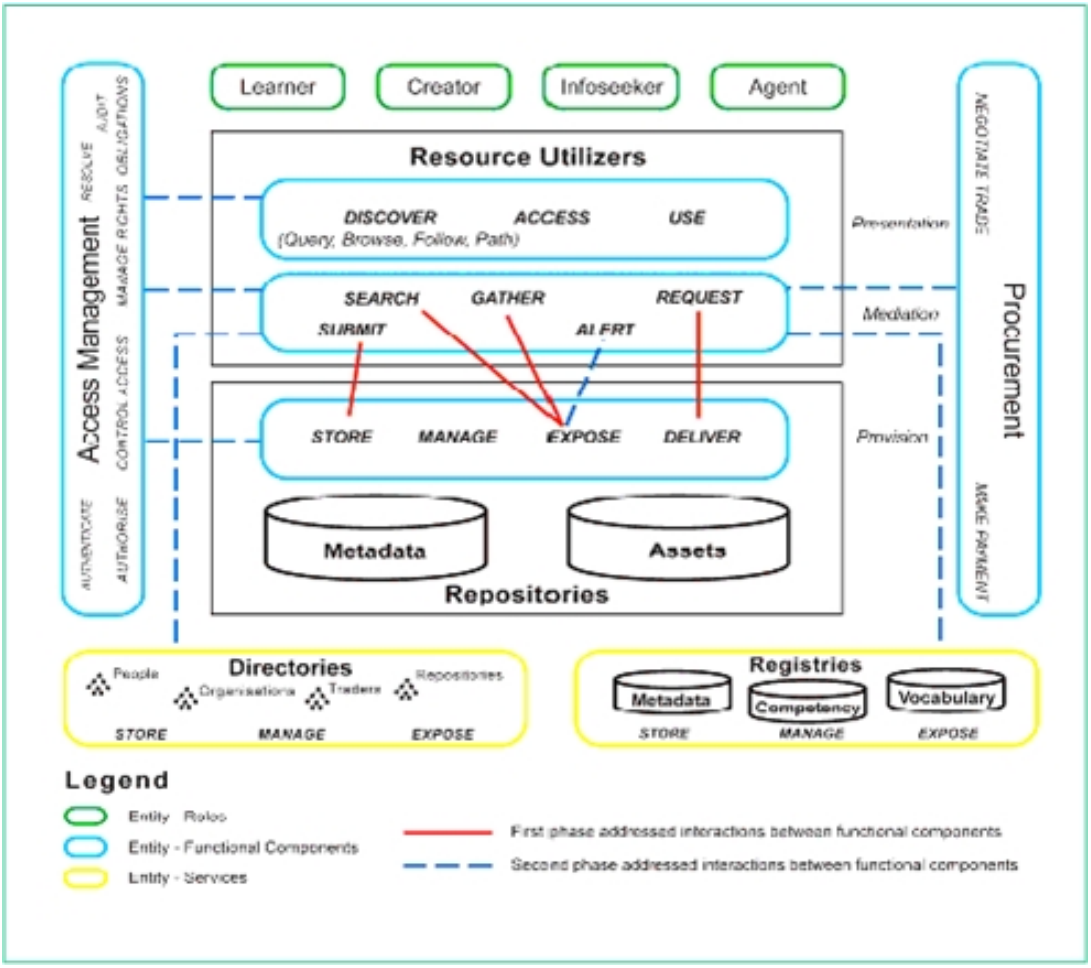

Figura 2.4. Contexto del IMS Digital Repository Interoperability

La figura 2.4 muestra la estructura interna propuesta por IMS; en la parte inferior se representan los repositorios digitales (de cualquier tipo) de objetos de aprendizaje, mientras que en la parte superior se encuentran los "usuarios de recursos" ("*resource utilizers*"), que representan cualquier persona o sistema que emplee esos recursos.

El DRIWG ha hecho corresponder las tareas más comúnmente empleadas por los usuarios con las respuestas que debe proporcionar el repositorio digital. Esto se detalla en la siguiente tabla, que además muestra tareas adicionales que los usuarios pueden demandar.

Tarea demandada por el usuario	Respuesta del repositorio digital
Buscar, reunir, alertar, hojear (discover, gather, alert, browse)	Presentar los datos requeridos por el usuario
Configurar el repositorio	Cambiar la Interfaz
Solicitar contenidos	Proporcionar los contenidos solicitados
Publicar	Almacenar
Entregar contenidos (desde el repositorio)	Almacenar (en otro repositorio)

Tabla 2.3. Respuestas de un repositorio digital

Roles en los repositorios digitales:

En el empleo de los repositorios digitales, los usuarios pueden tener distintos roles en distintos momentos. Los roles que los repositorios digitales deben soportar son:

- **Bibliotecario:** Es el responsable de mantener el sistema de clasificación del repositorio y asegurar la integridad de los metadatos. Un bibliotecario tiene los privilegios suficientes para editar y crear los metadatos, así como también para enlazar y desenlazar los objetos con los nodos del sistema de clasificación.
- **Contribuyente:** Es aquella persona que introduce los recursos (objetos de aprendizaje) en el repositorio; esta actividad forma parte de las prestaciones primordiales exigidas al repositorio, y tiene la ventaja de que quiénes crean los objetos son los mismos que crean los metadatos asociados, consiguiendo con ello que las descripciones sean más precisas.
- **Prestatario:** Aquel que toma prestados objetos de los repositorios de manera regular, y que suele personalizar la interfaz con la que interactúa con el repositorio; así como preservar y dejar constancia de las búsquedas realizadas en las sesiones con el repositorio.
- **Usuario casual:** En algunas circunstancias, los usuarios invitados pueden tener permisos para buscar, explorar o descargar objetos del repositorio pero sin tener su espacio personalizado propio. Generalmente los invitados no tienen que estar registrados como usuarios habituales.
- **Administrador:** Tiene la responsabilidad de gestionar los usuarios del repositorio, crear los usuarios nuevos, y borrar a los que ya no usen el repositorio. Es también el encargado de establecer las prerrogativas de acceso de que disponen los usuarios casuales.

- **Agente Software:** Es el nombre que reciben aquellos entornos visuales de aprendizaje u otros repositorios digitales, que pueden consultar el repositorio y descargar elementos.

En muchas ocasiones se da el caso de que el contribuyente y el prestatario es la misma persona, generalmente eso ocurre cuando el repositorio se emplea para compartir recursos entre los componentes de un grupo que generan recursos de aprendizaje y los usan para construir cursos e-learning.

Además de estos roles individuales también se pueden crear grupos de usuarios que trabajen conjuntamente, por ejemplo cuando un grupo asume la responsabilidad compartida de crear e impartir un curso.

2.1.3 Estándares

2.1.3.1 Concepto de Estándar

Los estándares son acuerdos internacionales documentados o normas establecidas por consenso mundial. Contienen las especificaciones técnicas y de calidad que deben reunir todos los productos y servicios para cumplir satisfactoriamente con las necesidades para las que han sido creados y para poder competir internacionalmente en condiciones de igualdad. El objetivo primordial al crear un estándar es impedir que en el mercado se impongan determinadas tecnologías ofrecidas por las empresas de un ámbito industrial concreto, evitando así que imperen intereses económicos privados.

Las diferentes organizaciones internacionales de estandarización ofrecen definiciones oficiales de lo que es un estándar. De acuerdo a la organización internacional de normalización (ISO) [2005], estándar se define como lo que "*contribuye para hacer la vida más fácil, y para incrementar la confiabilidad y efectividad de los bienes y servicios que utilizamos*". También, según la ISO, se trata de “acuerdos documentados que contienen especificaciones técnicas u otros criterios, para ser utilizados constantemente como reglas o definiciones de características, para asegurar qué materiales, productos, procesos y servicios son adecuados para sus propósitos”.

Por su parte, el BSI (*British Standard Institute*), describe un estándar como “*una especificación publicada que establece un lenguaje común, y contiene una técnica u otro criterio, que está diseñado para ser usado constantemente, como una regla o una definición*” [BSI, 2005].

Los estándares ofrecen:

- Una base de comparación.
- Una medida de la calidad, cantidad o nivel.
- Un consenso de opiniones entre individuos, grupos u organizaciones.

Los estándares son desarrollados por organizaciones oficiales con ánimo de evitar que intereses privados determinen normas y garantizar la comunicación entre los diferentes dispositivos y/o plataformas con los denominados estándares oficiales o de jure. En contraposición a éstos estándares se encuentran los designados de facto, los cuales sin estar impuestos por ninguna organización, y a veces sin estar emitidos por ninguna norma, hacen que se conviertan en estándares al usarse de manera repetitiva. El estado ideal de un estándar es cuando un estándar de jure es también de facto.

Existe cierta confusión sobre los términos estándares y especificación. La diferencia fundamental entre ambos es que las especificaciones son desarrolladas por comités no acreditados. Algunos de los más conocidos son: IETF (*Internet Engineering Task Force*), W3C (*World Wide Web Consortium*), OMG (*Object Management Group*), IMS (*Instructional Management Systems*). Por el contrario, un estándar es una especificación desarrollada y acreditada por comités de estandarización específicos. Ejemplo de este tipo de comités incluye a: IEEE (*Institute of Electrical and Electronics Engineers*), ISO, ANSI (*American National Standards Institute*), BSI, AENOR, etc. Incluso, en algunas industrias, el producto no puede ser vendido hasta que no recibe la certificación necesaria o es aprobada por el gobierno pertinente, como en el caso de los productos y servicios eléctricos o electrónicos, acreditados por IEEE.

Por lo tanto, una especificación es una descripción documentada de una tecnología. Las especificaciones son recomendaciones técnicas y de calidad que no se han adoptado de manera oficial en un ámbito tecnológico o de cualquier otra índole. Es decir, no es obligatorio su uso. La mayoría de los estándares tampoco son obligatorios, ya que solo lo son cuando los impone la ley.

2.1.3.2 El proceso de estandarización

El proceso de elaboración de un estándar es similar al de creación y aprobación de las leyes: una vez se ha realizado el grueso del trabajo, este debe ser ratificado por un organismo oficial. Puede parecer un proceso lento y poco efectivo, pero hay que tener en cuenta que el éxito de un estándar radica en su nivel de aceptación, por lo que un grupo de estandarización debe ser un organismo que se encargue de recopilar requisitos de múltiples fuentes y elabore con ellos una especificación consensuada [Sancho, 2002].

Algunas especificaciones acaban convirtiéndose en estándar, lo que significa que han recibido una acreditación oficial.

Para que una especificación evolucione hacia estándar debe pasar por un proceso, el cual tiene sus fases. Entre las principales fases en el diseño y desarrollo de estándares, y según Masie [2002], podemos mencionar las siguientes:

- **Fase 1- Requisitos**: El proceso comienza con grupos denominados consorcios y formados por gobiernos, empresas, individuos, académicos, etc. Se recogen requisitos e ideas y experiencias de la comunidad de usuarios, instituciones académicas, la industria y demás entes involucrados.
- **Fase 2- Especificaciones:** Expertos en realizar especificaciones preparan un borrador técnico del tema en cuestión, donde se presentan las principales ideas de la especificación que se desea desarrollar.
- **Fase 3- Prueba y uso**: Se desarrollan modelos de referencias y documentos específicos para ser usados y validados por grupos específicos de usuarios.
- **Fase 4- Estándar:** Si la especificación generada es válida y ampliamente aceptada después del período de prueba, puede ser enviada a cuerpos especializados para someterlos a su aprobación como estándar. Algunas oficinas de acreditación en e-learning son IEEE LTSC (*Learning Technology Standards Committee*), ANSI, o el CEN/ISSS (*European Committee for Standardization / Information Society Standardization System*). Una vez aprobada la especificación será reconocida como un estándar (por ejemplo, ISO) aprobado.

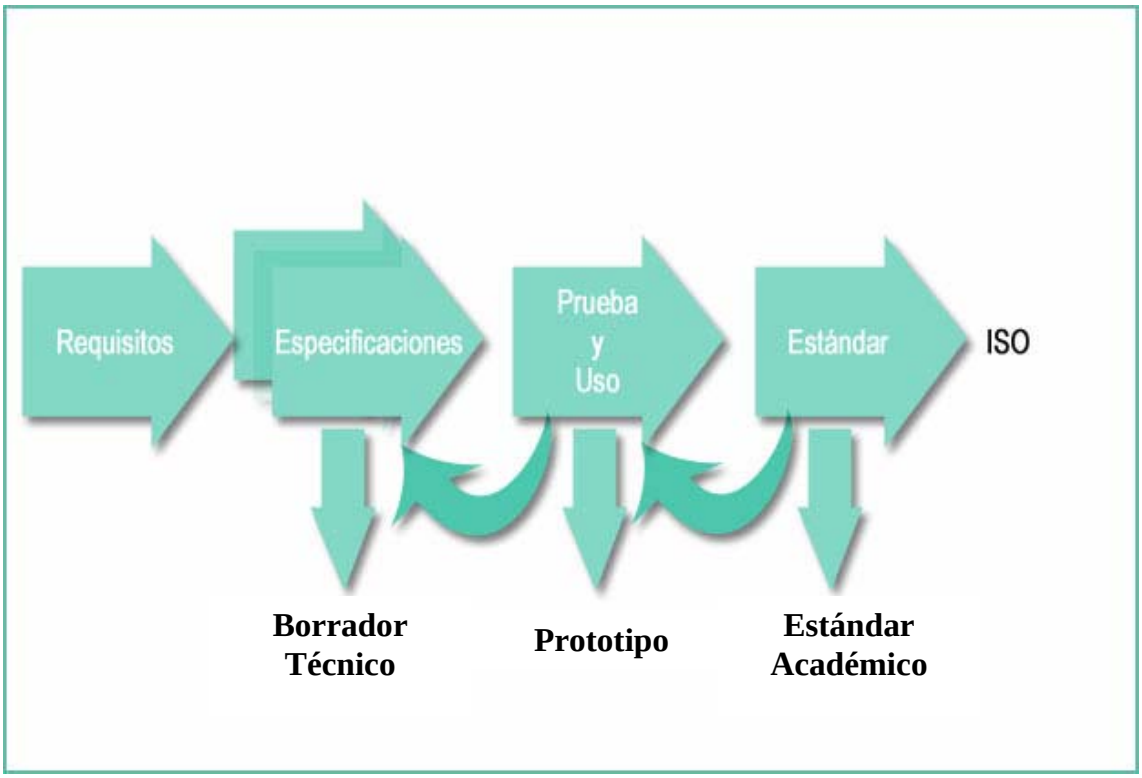

Figura 2.5. ***Proceso de creación de estándares***

2.1.3.3 Estándares de e-learning

Sin duda, el mayor problema que aborda la industria del e-learning en la actualidad es la ausencia de unas metodologías técnicas, documentales y psicopedagógicas comunes y aceptadas, que garanticen los objetivos de accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales curriculares encontrados en las redes.

Hay estándares genéricos apoyados por diferentes organismos que, desde hace años, establecieron sus propias especificaciones. Actualmente se está produciendo una convergencia hacia estándares comunes e intercambiables que soportan la definición de recomendaciones y nuevos estándares para campos de actividad específicos como en el caso del e-learning.

Las principales razones que impulsan la creación de estándares en el área de e-learning son [Rehak, 2003]:

- Protección de la inversión ante quiebra de proveedores.
- Portabilidad de contenidos de cursos entre diferentes tecnologías de adiestramiento y/o plataformas e-learning.
- Integración de iniciativas e-learning con sistemas de Recursos Humanos Corporativos, Sistemas de Control o Administración de gestión académica.
- Integración de plataformas e-learning en la infraestructura tecnológica existente.
- En definitiva, la mejora del e-learning.

Los objetivos que persiguen cumplir los estándares e-learning son [Hodgins, 2001]:

- Durabilidad: Que la tecnología desarrollada con el estándar evite la obsolescencia de los cursos.
- Interoperabilidad: Que se pueda intercambiar información a través de una amplia variedad de LMS o LCMS.
- Accesibilidad: Que se permita un seguimiento del comportamiento de los alumnos.
- Reutilización: Que los distintos cursos y objetos de aprendizaje puedan ser reutilizados con diferentes herramientas y en distintas plataformas.
- Adaptabilidad: Los estándares se refieren al hecho de poder facilitar la adaptación o personalización del entorno de aprendizaje.
- Productividad: Si los proveedores de tecnología e-learning desarrollan sus productos siguiendo estándares comúnmente aceptados, la efectividad de e-learning se incrementa significativamente y el tiempo y costos se reducen.

2.1.3.4 Tipos de estándares e-learning

En el mercado existen tanto LMS como LCMS de muchos fabricantes distintos. Por ello se hace necesaria una normativa que compatibilice los distintos sistemas y cursos a fin de lograr dos objetivos:

- Que un curso de cualquier fabricante pueda ser cargado en cualquier LMS de otro fabricante.
- Que los resultados de la actividad de los usuarios en el curso puedan ser registrados por el LMS.

Como se puede ver en la siguiente figura, los distintos estándares que se desarrollan hoy en día para la industria del e-learning se pueden clasificar en los siguientes tipos:

- **Sobre el Contenido o Curso**: Estructuras de los contenidos, empaquetamiento de contenidos, seguimiento de los resultados.
- **Sobre el Alumno**: Almacenamiento e intercambio de información del alumno, competencias (habilidades) del alumno, privacidad y seguridad.
- **Sobre la Interoperabilidad:** Integración de componentes del LMS, interoperabilidad entre múltiples LMS.

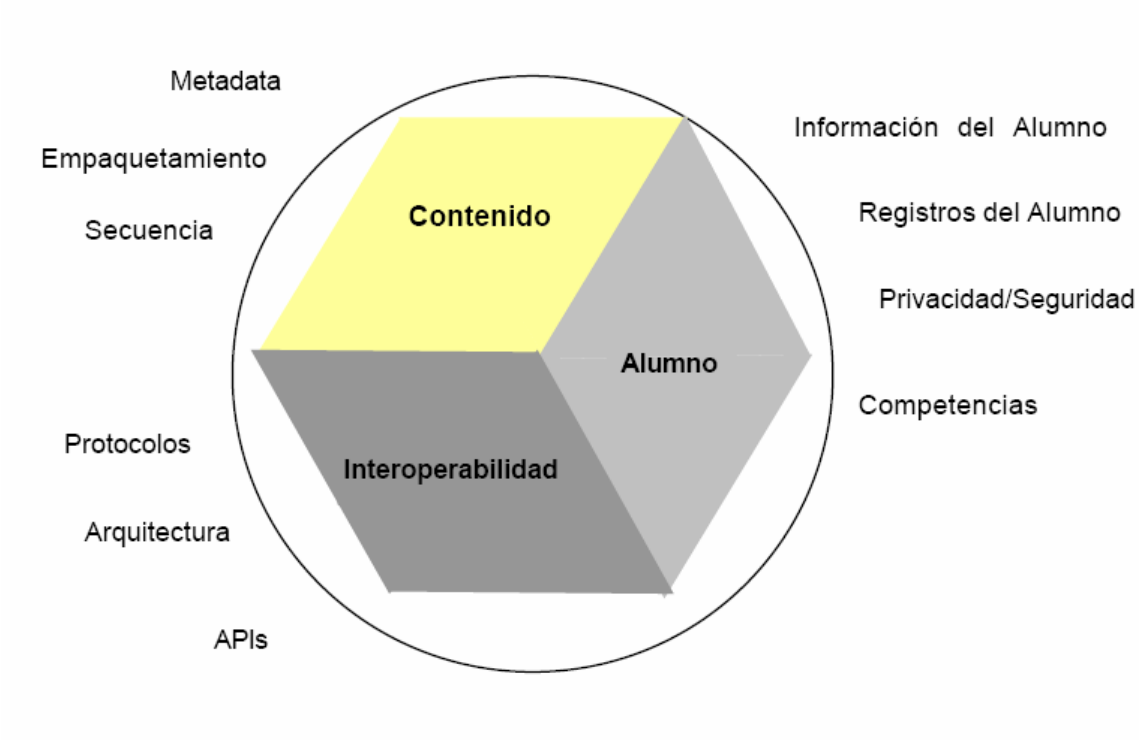

Figura 2.6. Áreas afectadas por los estándares de e-learning

Al hablar sobre un estándar e-learning, nos estamos refiriendo a un conjunto de reglas en común para las compañías dedicadas a la tecnología e-learning. Estas reglas especifican cómo los fabricantes pueden construir cursos *on-line* y las plataformas sobre las cuales son impartidos estos cursos de tal manera de que puedan interactuar unas con otras. Estas reglas proveen modelos comunes de información para cursos e-learning y plataformas LMS, que básicamente permiten a los sistemas y a los cursos compartir datos o comunicarse con otros. Esto también ofrece la posibilidad de incorporar contenidos de distintos proveedores en un solo programa de estudios.

Estas reglas, además, definen un modelo de empaquetamiento estándar para los contenidos. Los contenidos pueden ser empaquetados como objetos de aprendizaje (*learning objects* o LO), de tal forma que permita a los desarrolladores crear contenidos que puedan ser fácilmente reutilizados e integrados en distintos cursos.

Si se consigue la estandarización de esta tecnología, desarrolladores de cursos *on-line* y constructores de componentes y plataformas e-learning beneficiarían a toda la comunidad de usuarios, ya que además de facilitar la interoperatividad de componentes, preservaría las inversiones que se realizaran en este campo. La vida de los cursos *on-line* se vería incrementada al poder intercambiarse cursos virtuales entre diferentes plataformas, sin la necesidad de realizar costosas modificaciones.

Los estándares han iniciado el camino hacia una forma cómoda y viable de empaquetar los recursos y contenidos, tanto para los estudiantes que cambian de sistema, los docentes que utilizan en distintos contextos estos materiales y los desarrolladores que tienen que construir nuevas herramientas y mejorar las vigentes.

2.1.3.5 Beneficios de los estándares de e-learning

Los estándares en e-learning no son totalmente sólidos ni están definidos en estos momentos. La madurez de la industria presionará por estándares claros y bien definidos. Existe una gran confusión en la proliferación de consorcios, especificaciones e individuos que desean ganar terreno en la nueva industria que se vislumbra.

Salvaguardar las inversiones realizadas, garantizar la interoperatividad de los componentes tecnológicos que se adquieran, y el acceso a la información desde diferentes puntos de vista, es una tarea prioritaria para las personas involucradas en el e-learning.

Las instituciones educativas deben estar atentas a las iniciativas de estandarización de su tecnología, ya que sería muy costoso quedar con contenido aislado en un mundo cada vez más interconectado y que clama por la colaboración institucional como mecanismo para garantizar una educación de calidad.

Una industria que nace y carece de estándares es atractiva para incorporar nuevas maneras y formas de trabajar sistematizadas y organizadas. Por ello, educarse y fomentar la cultura de los estándares es imperativo, la experimentación debe dar paso a la organización y consolidación de una industria que nace y se fortalece cada día.

No solo el uso y aplicación de estándares es propicio para el desarrollo de contenidos y materiales educacionales. Es de igual importancia, la utilización de estándares en los llamados portales educacionales. Un portal, la puerta virtual de acceso a la institución, debe reflejar consistencia y organización coherentes que guíen a los visitantes a las distintas instancias de información a las cuales deseen acceder. Como consecuencia, una imagen institucional sería de mayor beneficio para los visitantes del portal, que múltiples imágenes esparcidas a lo largo del mismo.

Los estándares e-learning proporcionan beneficios multifacéticos, esto incluye instituciones académicas, corporaciones, individuos y a la industria en general. He aquí algunos casos:

- La industria e-learning como un todo. La interoperatividad entre diferentes componentes tecnológicos de e-learning elimina temores de inversión en la tecnología, al mismo tiempo que incentiva la adopción más generalizada del e-learning, lo cual facilita el desarrollo de la industria como un todo.
- Proveedores de tecnología. Con un sistema de estándares, los proveedores pueden ver expandidos sus mercados. Los contenidos y las plataformas basadas en estándares son más sostenibles a largo plazo. Los proveedores de contenido podrán fácilmente reutilizar contenidos entre diferentes programas. De igual manera, las herramientas estándares facilitan el desarrollo de contenidos y de nuevas herramientas.
- Instituciones Académicas. Compartir contenidos de cursos será mucho más fácil para profesores. Teniendo como estándar un navegador de Internet, los estudiantes y los profesores podrán fácilmente intercambiar información. Los estándares e-learning ayudan a preservar el capital invertido en tecnología y desarrollo de profesores. Transferir contenidos y evaluaciones entre instituciones será mucho más sencillo.
- Corporaciones. El poder de adquirir una gran gama de contenidos y que puedan funcionar correctamente en cualquier plataforma expande las potencialidades de formación de las empresas. La rapidez de puesta en marcha de cursos y programas enriquece los programas de formación corporativos. Todo esto trae consigo una mejor rentabilidad de la inversión realizada en e-learning.

- Individuos. Personas independientes tendrán acceso a mucho más conocimiento en diferentes formatos y lenguajes, esto conlleva a una reducción en costes de formación.

2.1.3.6 Futuro de los estándares de e-learning

En los próximos años, el trabajo de las distintas organizaciones que están trabajando en las especificaciones para estándares e-learning estará centrado en los siguientes temas:

- Repositorio de Contenidos: Las organizaciones están orientando su esfuerzo en desarrollar estándares de contenidos e-learning. **El principal objetivo es tener repositorios de objetos de aprendizaje reutilizables, de tal manera que puedan ser agrupados en objetos de aprendizaje adaptables y expedidos por cualquier plataforma e-learning.** Sin embargo, **uno de los mayores problemas al que se enfrenta hoy en día la industria del e-learning es la interoperabilidad entre distintos sistemas de aprendizaje que quieran compartir sus contenidos.**
- Internacionalización y Localización: Los distintos grupos que están desarrollando especificaciones para e-learning participan de forma activa en todo el mundo, y cada día existe una mayor colaboración entre ellos. Esto genera dos desafíos: la creación de estándares "culturalmente" neutrales (internacionalización), y la adaptación de los estándares a las necesidades locales (localización).
- Programas de certificación: Existe un creciente énfasis en crear test de compatibilidad y programas de certificación. ADL está trabajando en un programa de certificación. Actualmente sólo existen los programas de certificación de AICC.
- Arquitectura: La industria del e-learning ha estado creciendo sin tener una clara visión de los componentes de un sistema de e-learning y de la forma en que interactúan. **La necesidad de definir una arquitectura global es crítica para la evolución del desarrollo de estándares.**

2.1.3.7 Organizaciones de estandarización en e-learning

La implantación y difusión de estándares ha sido el medio de generalización de las aplicaciones en Internet y de la extensión de la propia red. No se podría entender la generalización del Web sin la definición del protocolo estandarizado HTTP (*HyperText Transfer Protocol*), del lenguaje HTML (*HyperText Markup Language*), o la de la propia Internet sin el protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*). El desarrollo de recomendaciones y estándares por parte de la World Wide Web Consortium, como XML (*eXentesible Mark up Language*), derivado de SGML

(*Standard Generalized Markup Language*), meta lenguaje de marca con el cual podemos crear lenguajes específicos, supone un nuevo empuje para la creación de contenidos en Internet.

Estos estándares genéricos han sido apoyados por diferentes organismos que, desde hace años establecieron sus propias especificaciones. Actualmente se está produciendo una convergencia hacia estándares comunes e intercambiables que soportan la definición de recomendaciones y nuevos estándares para campos de actividad específicos como el e-learning.

Los organismos de estandarización sobre e-learning más importantes son:

- AICC (*Aviation Industry Computer Based Training Committee*).
- IEEE (*Institute of Electrical and Electronic Engineers*) con el grupo de trabajo LTSC (*Learning Technology Standards Committee*).
- IMS (*Instructional Management System*) Global Learning Consortium.
- ADL (*Advanced Distributed Learning*).
- ARIADNE (*Alliance of Remote Instructional Authoring and Distribution Networks for Europe*).
- ISO (*International Standards Organization*), con el subcomité SC36 denominado "*Information Technology for Learning, Education and Training*".

Estos cubren el espectro de necesidades de definición en el entorno del e-learning: Estructura de cursos, contenidos reutilizables, metadatos, arquitecturas de plataformas e intercambio de datos. Los metadatos caracterizan a los datos y las aplicaciones, describiendo el contenido; los principales usos son catalogar, organizar, mantener los datos e intercambiarlos.

A manera de resumen, a continuación se presenta una tabla con las principales especificaciones relacionadas con los contenidos para e-learning:

Especificaciones	Organización	Descripción
Runtime Communication	ADL, también está siendo estandarizada por IEEE	APIs para la comunicación entre LMS y SCOs
CMI Data Model	AICC, adoptada también por AADL y estandarizada por IEEE	Define vocabulario y respuestas para la comunicación entre LMS y SCOs
Learning Object Metadata	IEEE, adaptada también por IMS, ADL e ISO	Define categorías usadas para describir los contenidos de aprendizaje
Aggregation Model	IMS, adaptado también por ADL	Indica cómo empaquetar los contenidos de un curso

Tabla 2.4. ***Principales especificaciones relacionadas con los contenidos para e-learning***

Estos organismos están trabajando conjuntamente en la elaboración de estándares y especificaciones para el diseño de entornos tecnológicos en lo relativo al proceso de enseñanza mediante Internet; así pues vamos a detallar cómo está ayudando cada organización en el proceso de estandarización oficial.

IEEE

- **Historia de IEEE**

El IEEE (*Institute of Electrical and Electronics Engineers*) [IEEE, 2006] se trata de un organismo que elabora normas para el instituto de estandarización americano (ANSI), muchas de las cuales se convierten posteriormente en estándares ISO. Dentro del IEEE se ha creado el comité LTSC *Learning Technologies Standards Committee*, encargado de preparar normas técnicas, prácticas y guías recomendadas para el uso informático de componentes y sistemas de educación y de formación, en concreto, los componentes de software, las herramientas, las tecnologías y los métodos de diseño que facilitan su desarrollo, despliegue, mantenimiento e interoperación.

Para ello se basó en los trabajos desarrollados por el comité de la AICC y trató de mejorarlo, creando la noción de metadatos para los objetos de aprendizaje, o *Learning Object Metadata* (LOM) [IEEE, 2002], que define elementos para describir los recursos de aprendizaje. IMS y ADL utilizan estos elementos y las estructuras de LOM en sus respectivas especificaciones. En la actualidad ISO está en proceso de asumir LOM como estándar.

- **Ámbito y propósito de IEEE**

Su misión principal es “desarrollar estándares técnicos, prácticas recomendadas y guías para componentes software, herramientas, tecnologías y métodos de diseño que faciliten el desarrollo, implantación, mantenimiento e interoperabilidad de implementación en ordenadores de sistemas educativos” [IEEE, 2001].

- **Documentos Técnicos de IEEE**

LTSC tiene más de una docena de grupos de trabajo (*working groups* o WGs) y grupos de estudio (*study groups* o SGs) que desarrollan especificaciones para la industria del e-learning.

Los siguientes grupos de trabajo son parte de las actividades generales de IEEE LTSC:

- IEEE 1484.1 Architecture and Reference Model
- IEEE 1484.3 Glossary

Los siguientes grupos de trabajo son parte de las actividades relacionadas con los datos y metadatos:

- IEEE 1484.12 Learning Object Metadata
- IEEE 1484.14 Semantics and Exchange Bindings
- IEEE 1484.15 Data Interchange Protocols

Los siguientes grupos de trabajo son parte de las actividades relacionadas con los LMS y las aplicaciones:

- IEEE 1484.11 Computer Managed Instruction
- IEEE 1484.18 Platforms and Media Profiles
- IEEE 1484.20 Competency Definitions

LTSC también trabaja en forma coordinada con otra iniciativa denominada ISO JTC1 SC36, que es un subcomité formado en forma conjunta por la ISO (*International Standard Organization*) y por la IEC (*International Electrotechnical Commission*), dedicado a la normalización en el ámbito de las Tecnologías de la Información para la formación, educación y aprendizaje.

- **Especificaciones de IEEE**

El trabajo más importante de esta organización es el estándar que especifica Metadatos para Objetos Educativos, *Learning Object Metadata* **(LOM)**. Este estándar especifica un esquema conceptual de datos que define la estructura de una instancia de metadatos para un objeto educativo. Para este estándar, un objeto educativo se define como cualquier entidad, digital o no, susceptible de ser usada en aprendizaje, educación o formación.

En lo que respecta a este estándar, una instancia de metadatos para un objeto educativo describe las características relevantes del objeto educativo al que se aplica. Dichas características se pueden agrupar en las categorías general, ciclo de vida, meta-metadatos, técnica, uso educativo, derechos, relación, anotación y clasificación.

El esquema conceptual de datos definido en este estándar permite la diversidad lingüística, tanto de los objetos educativos como de las instancias de metadatos que los describan.

Este esquema conceptual de datos especifica los elementos de datos de los que se compone una instancia de metadatos para un objeto educativo. Se asume que esta parte del estándar será referenciada por otros estándares que definirán descripciones de implementación del esquema de datos; de manera que una instancia de metadatos para un objeto educativo pueda ser usada por un sistema basado en tecnología educativa para gestionar, localizar, evaluar o intercambiar objetos educativos.

No define cómo un sistema basado en tecnología educativa representará o usará una instancia de metadatos de un objeto educativo.

El propósito de este estándar es facilitar la búsqueda, evaluación, adquisición y uso de los objetos educativos, por ejemplo, por alumnos, profesores o procesos automáticos de software. Este estándar también facilita el intercambio y uso compartido de objetos educativos, permitiendo el desarrollo de catálogos e inventarios al tiempo que se toman en consideración la diversidad cultural y los contextos lingüísticos en los que los objetos educativos y sus metadatos serán reutilizados.

Especificando un esquema conceptual de datos común, se asegura que las implementaciones de los Metadatos de Objetos Educativos tendrán un alto grado de interoperabilidad semántica. Como consecuencia, se simplificarán las transformaciones entre implementaciones.

También especifica un esquema base que puede extenderse a medida que se avanza en su desarrollo práctico, por ejemplo, facilitando la planificación adaptativa y automática de los objetos educativos por agentes software.

IMS

- **Historia de IMS**

IMS (*Instructional Management Systems*) [IMS, 2006] es un proyecto iniciado en el año 1997 por la asociación internacional EDUCAUSE [2006]. Dicha organización pretende incidir en los cambios en la educación superior por medio de “la introducción, el uso y la administración de recursos de información y tecnologías en la enseñanza, en el aprendizaje, en la investigación y en la administración institucional” para lo cual han iniciado toda una serie de programas para el desarrollo

de actividades profesionales, como, entre otros, la publicación de revistas, la investigación, la estructuración de iniciativas políticas y estratégicas, o el desarrollo de servicios de información en línea. Una idea de la magnitud de este tipo de esfuerzos se refleja en los siguientes datos: pertenecen a EDUCAUSE más de 1700 colegios, universidades y organizaciones educativas, así como más de 150 corporaciones que tienen que ver con los niveles superiores de la educación.

Actualmente IMS es una organización sin ánimo de lucro (conocido como *IMS Global Learning Consortium*) en la que participan más de 50 miembros y afiliados, provenientes del sector del e-learning. Incluyendo vendedores de hardware y software, instituciones educativas, organismos gubernamentales, proveedores de contenidos multimedia, etc. El consorcio facilita un foro neutral en el que los miembros que son competidores entre sí colaboran estableciendo y/o empleando estándares y especificaciones comunes, para satisfacer las necesidades del mundo real.

En sus inicios el estándar IMS surgió para su empleo en sistemas de educación superior, no obstante las especificaciones actuales no se limitan a éste área y abarcan un amplio rango de contextos de aprendizaje, desde las "escuelas K-12" (en inglés *K-12 school*, término empleado para referenciar a las guarderías hasta los 12 años) hasta los métodos de aprendizaje para grandes empresas y los organismos gubernamentales.

- **Ámbito y propósito de IMS**

El ámbito de las especificaciones IMS, en ocasiones denominadas como "aprendizaje distribuido" (en inglés, "*distribuited learning*"), incluye entornos "*on-line*" y "*off-line*", produciendo lo que se denomina entornos síncronos (en tiempo real) o asíncronos. Esto supone que los contextos de aprendizaje beneficiarios de las especificaciones IMS incluyen entornos específicos para Internet (como los cursos basados en Web) así como situaciones de aprendizaje que requieran de recursos "*off-line*" (como los cursos basados en CD-ROM) Los estudiantes pueden encontrarse en un entorno tradicional (el aula de una escuela o la universidad), en un curso de aprendizaje de una empresa, o en casa. Por ejemplo, la especificación de recursos de aprendizaje IMS Meta-Data que veremos en apartados posteriores, facilita al estudiante la búsqueda de información a través de una herramienta buscadora que actúa tanto sobre los recursos "Web" como sobre el CD-ROM o DVD.

- **Documentos Técnicos de IMS**

Los documentos producidos por IMS establecen especificaciones que definen formatos y protocolos para el intercambio de información entre entornos IMS. Pero no pretende imponer a los desarrolladores la forma en que deben hacer sus aplicaciones educativas. El grupo técnico de desarrollo asumió como punto de partida del trabajo, unos requerimientos o especie de lista de deseos generada por los usuarios y desarrolladores de recursos de aprendizaje y a partir de éstos, se inicia la elaboración de las especificaciones.

El resultado del proceso de diseño de requerimientos, está plasmado en un documento público liberado por EDUCOM/NLII IMS [IMS, 2006]. El documento presenta cuatro secciones: resumen, diseño, requerimientos propiamente dichos y por último implementación. Destacamos a continuación los elementos más interesantes y relevantes del documento:

- Se establecen cuatro tipos de interesados o actores afectados por el proyecto IMS. Son los siguientes: Alumnos, Profesores, Proveedores y Coordinadores.
- Se definen cuatro dimensiones básicas de interacción de estas personas: Distancia, Tiempo, Afiliación y Modo de entrega. Respecto al tiempo no esta limitado a una progresión lineal, incluye interacciones síncronas y asíncronas. El entorno puede situarse en una localización física o distribuirse a través de espacios físicos y electrónicos, lo que implica un modelo de partes interrelacionadas.
- La modulación. Es consecuencia de la concepción de partes interrelacionadas antes mencionada. La idea de la modulación de los materiales y procesos de enseñanza está también fuertemente apoyada por el predominio del diseño orientado a objetos en el área del software, y supone un camino a ambientes de aprendizaje distribuidos.
- El proceso de determinación de requerimientos, parte de las características deseables, y a partir de ellas analiza los requisitos para satisfacerlas. De modo que en el documento se listan una serie de grupos de características y en cada caso se especifican los requerimientos derivados. Ello implica que no hay correspondencia uno a uno, entre características deseables y requerimientos, pues uno de ellos puede servir a más de uno de aquéllos. La siguiente figura ilustra los grupos de requerimientos básicos.

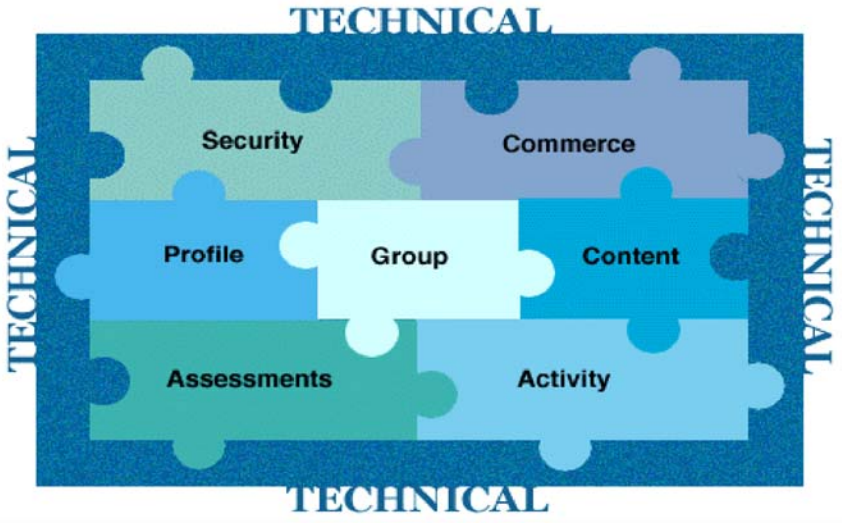

Figura 2.7. Grupos de requerimientos básicos

- Respecto a los requerimientos, el documento finaliza haciendo tres observaciones con relación a la interacción entre los grupos de funcionalidades establecidos:

    - En primer lugar, los requisitos técnicos actúan como un armazón o infraestructura que sostiene todos los requerimientos juntos.
    - El segundo punto importante del diagrama es la posición central ocupada por los requerimientos de gestión de grupo.
    - Finalmente, la descripción de las características como partes modulares reitera la escalabilidad e interoperabilidad de los recursos IMS.

Todas las especificaciones de IMS se detallan en tres documentos:

- Guía de Implementación y Consejos. Es el documento más narrativo de los tres. En él se incluyen: la forma de uso de la especificación, la relación con otras especificaciones, y cualquier tipo de información variada que pueda servir de ayuda. Suele ser el documento que se usa para iniciarse en la especificación.
- Modelo de Información. Documento que muestra la estructura de datos completa. Normalmente posee una tabla detallada de cada uno de los elementos de la especificación. En ella, se enumeran las propiedades de los elementos tales cómo el nombre, la multiplicidad, el tipo o si son obligatorios.
- Documento de Enlace. Documento que ofrece la forma de representar la estructura de datos de la especificación en XML. Muestra el árbol XML con cada uno de sus elementos y sus atributos.

- **Especificaciones de IMS**

Las especificaciones elaboradas por IMS son:

1. **Accessibility**: Las especificaciones de accesibilidad tratan de garantizar que los productos y tecnologías que se desarrollen en el área de la educación sean útiles y accesibles para todo el mundo, en todo momento y en todos los lugares.

2. **Competency Definitions**: Las especificaciones de definición de competencias tratan de establecer las formas de gestión de competencias relacionadas con el sistema e-learning, las cuales engloban, entre otras, la gestión de las competencias de los alumnos, la creación de definiciones de competencias y la asociación de dichas definiciones con los alumnos, y los prerrequisitos. Este modelo de información puede usarse para el intercambio de definiciones entre diferentes sistemas de aprendizaje, sistemas de recursos humanos, contenidos de aprendizaje, repositorios de competencias o habilidades y en cualquier otro sistema relevante. Estas especificaciones también proporcionan referencias únicas de descripciones para su inclusión en otros modelos de información.

3. **Content Packaging:** Estas especificaciones establecen las pautas y descripciones para el empaquetamiento de los materiales de aprendizaje (como pueden ser un curso individual o una colección de cursos), en un paquete que se pueda distribuir. El "*Content Packaging*" sirve de enlace con la descripción, estructura, localización de los materiales y la definición de contenidos especiales "*on-line*".

4. **Digital Repositories**: El propósito de las especificaciones de los repositorios digitales es describir los contenidos y desarrollos de los mismos, y proporcionar modelos de información y protocolos para habilitar la interoperabilidad entre diferentes repositorios, tanto para las operaciones de búsqueda como en las de publicación y almacenamiento a través de la red.

5. **Enterprise**: Estas especificaciones son un modelo de información que da soporte a la interoperatividad entre personas, grupos y afiliaciones entre dos o más sistemas de aprendizaje.

6. **Enterprise Services**: Se trata de la definición de cómo los sistemas de aprendizaje son capaces de gestionar el intercambio de información concerniente a las personas, grupos y componentes del contexto de aprendizaje.

7. **ePortfolio**: Las especificaciones de ePortfolio están diseñadas para que la documentación sea intercambiable entre diferentes sistemas e instituciones. Esta especificación:

a) Soporta el avance de los aprendizajes de larga duración, que son importantes para ciertas iniciativas.
b) Facilita el intercambio de los documentos.
c) Permite a educadores e instituciones mejorar las capacidades de determinadas competencias.
d) Realza el aprendizaje e incrementa el desarrollo del empleado.

8. **General Web Services**: La base de los Servicios Web Generales promueve la interoperabilidad de las especificaciones de los servicios Web basados en software y plataformas de diferentes vendedores. Se centra en la especificación de un conjunto de servicios Web y en los problemas más comunes encontrados en su implementación. Trata de dirigir la interoperabilidad en la capa de aplicación, en particular, la descripción de los comportamientos expuestos vía servicios Web.

9. **Learner Information**: Se trata de la información referente al estudiante (entendiendo como tal tanto una persona como a un grupo de ellas) o a los productores del contenido de aprendizaje (creadores, proveedores o vendedores) La especificación del paquete de información de aprendizaje IMS (LIP) establece el modelo de interoperatividad de los sistemas de aprendizaje basados en Internet con aquellos otros sistemas que soportan dicho entorno. El propósito de la especificación es definir un conjunto de paquetes que puedan ser usados con otros servidores de información de aprendizaje. Un servidor de información de aprendizaje intercambia datos con los distintos sistemas de aprendizaje y proporciona una descripción de dichos datos.

10**. Learning Design**: Estas especificaciones proporcionan un amplio espectro de teorías pedagógicas para el estudio "*on-line*". En lugar de tratar de englobar las cuestiones específicas de muchas pedagogías, proporciona un lenguaje flexible y genérico para su descripción. Este enfoque tiene la ventaja de que tan solo es necesario aprender a manejar un conjunto de herramientas de diseño sobre las cuales aplicar las distintas pedagogías. Esta especificación amplía el concepto de "*Learning Object*" en el de "*Unit of Learning*".

11. **Meta-data:** Es el modelo de información IMS empleado para describir los contenidos que los Metadatos han de contener.

12. **Question and Test**: Es un modelo de información que describe los datos correspondientes a las preguntas y cuestionarios junto con sus soluciones.

13. **Resource List Interoperability (RLI)**: Es la especificación que detalla como los metadatos estructurados pueden ser intercambiados entre los diferentes sistemas que almacenan recursos. Las especificaciones se basan en un servicio abstracto y en un modelo de datos que describe de forma general los recursos, una colección de esos recursos, y los comportamientos asociados con el servicio de gestión de recursos. El modelo de datos se expresa en el lenguaje XML, combinando elementos del estándar IEEE-LOM e ISO 690-2.

14. **Shareable State Persistence**: Esta especificación describe una extensión para los sistemas en tiempo de ejecución (como SCORM) que facilita el almacenamiento del estado de la información compartida. Actualmente no hay un método de almacenamiento de la información de estado en los sistemas de tiempo real que permita que la información sea recuperada por el mismo objeto ni mediante otro que no sea propietario. Esta capacidad es crucial para la persistencia de una información, a veces en un estado complejo, que es generada por una variedad de contenido interactivo (ej.: simulaciones) y que actualmente se almacena y recupera a través de métodos propietarios.

15. **Simple Sequencing**: Define un método para representar el comportamiento y funcionalidades que los sistemas deben implementar. Incluye reglas que describen las ramificaciones o el flujo de instrucciones según los resultados de la interacción entre el estudiante y los contenidos.

16. **Vocabularies Definition Exchange (VDEX)**: Define una gramática especial para el intercambio de vocabulario, más concretamente para el intercambio de una lista simple de valores legibles por la máquina, o términos, junto con información que ayuda para que el ser humano pueda entender el significado o aplicación de los diferentes términos. VDEX se emplea para expresar los datos válidos en instancias del IEEE-LOM, IMS Meta-data, IMS LIP, ADL, SCORM, etc.

17. **Bindings, Schemas, WSDL**: IMS emplea los esquemas XML como principal método de control para las especificaciones de los XML-bindings (marcos, entornos de trabajo) Las especificaciones de los servicios emplean WSDL y para los modelos de datos se emplean esquemas XML (*Schemas*).

18. **Abstract Framework**: Es un mecanismo para permitir a IMS describir el contexto dentro del cual se desarrollan sus especificaciones de todas sus tecnologías e-learning. Este framework no se ocupa de definir la arquitectura IMS, sino de definir un conjunto de servicios para los cuales se definen sus especificaciones de interoperabilidad. En los casos donde IMS no puede producir

una especificación, trata de adoptar o recomendar una especificación conveniente de otra organización.

AICC

- **Historia de AICC**

La industria de la aviación (*AICC - Aviation Industry CBT Committee*) [AICC, 2006] ha sido tradicionalmente un gran consumidor de formación, por lo que en 1992 decidieron crear un comité que desarrollase una normativa para sus proveedores de formación basada en computador. De este modo garantizaban la armonización de los requerimientos de los cursos, así como la homogeneización de los resultados obtenidos de los mismos.

Fue el primer organismo creado para desarrollar un conjunto de normas que permitiese el intercambio de cursos CBT (*Computer Based-Training*) entre diferentes sistemas.

- **Ámbito y propósito de AICC**

Sus dos principales aportaciones son sus propuestas para entornos de ejecución y para estructuración de cursos, que constituyen la base de SCORM.

1. Estructuración de cursos

El comité de aprendizaje basado en computador del AICC contribuyó a la estandarización de las estructuras de cursos con su documento "*Guidelines for Interoperability*".

En la nomenclatura del AICC, las partes de un curso que se pueden reubicar para definir el orden en el que serán mostradas al estudiante se denominan elementos de estructura. Hay dos tipos de elementos de estructura: las unidades asignables, que es el elemento educacional más pequeño que se le puede mostrar al alumno (una página HTML, un simulador, un test, etc.) y bloques, que son agrupaciones de unidades asignables y otros bloques. Existe además otro elemento de construcción denominado objetivo, que se puede utilizar para definir requisitos en un curso. Las unidades asignables, bloques y objetivos se denominan elementos del curso. La especificación es neutral en cuanto al número de niveles que se pueden establecer en la jerarquía de un curso. Es posible anidar bloques del curso hasta el nivel que se desee. Sin embargo, AICC ha establecido una jerarquía de referencia de 10 niveles. El modelo de referencia define reglas de nombrado útiles para evitar inconsistencias.

2. El entorno de ejecución

AICC define los entornos de ejecución cómo *Computer Management Instruction* (CMI), mientras que se utilizan los términos *Computer Based Training* (CBT), lecciones o Unidades asignables para identificar los contenidos entregados a los alumnos.

El entorno de ejecución de AICC ha evolucionado: primero estaba orientado a equipos autónomos utilizando comunicación mediante ficheros, y posteriormente ha desarrollado una interfaz basada en HTTP que permite comunicación entre redes de computadores.

El primer avance que produjo AICC, y en el que resultó ser pionero, fue la separación del CBT y del CMI. Las razones en las que se basó para llegar a esta separación son:

- Los instructores están acostumbrados a la utilización de un CMI, la adopción de un nuevo CMI para la entrega de contenidos es un proceso costoso.
- El mantenimiento de varios CMI´s es muy costoso.
- El CMI determina, en gran medida, la apariencia de la interfaz de usuario que se ofrece a los alumnos. Es importante que esta apariencia sea la misma con independencia del contenido que esté gestionando.
- El CMI que mejor se adapta a las necesidades de una organización puede no proporcionar los contenidos del curso adecuados y viceversa. En caso de que se necesiten nuevos contenidos, sería preferible integrarlos en la plataforma existente.

En esta situación, AICC consideró que deberían proporcionarse estándares para la transferencia de un curso de un sistema dado a uno nuevo, incluyendo la estructura del curso, los criterios de navegación y los contenidos. Además también deberían ser estandarizadas tanto la comunicación entre el CMI y las lecciones, como la forma en la que debe ser gestionada la información de los estudiantes.

- **Especificaciones de AICC**

Las especificaciones del AICC cubren nueve áreas principales, que van desde los *Learning Objects* (LO) hasta los *Learning Management Systems* (LMS). Normalmente, cuando una compañía afirma que cumple con las especificaciones AICC, significa que cumple con al menos una de estas "*guidelines*" y recomendaciones (*AICC Guidelines and Recommendations, AGRs*).

La lista completa de AGRs es la siguiente:

1. **AGR 001 AICC Publications**: Este documento describe todas las publicaciones de AICC. Identifica y proporciona un extracto de las pautas actuales y de las recomendaciones de AICC y de los documentos técnicos del documento.
2. **AGR 002 Courseware Delivery Stations**: Este documento contiene las recomendaciones de la industria de la aviación para la adquisición de una estación de entrenamiento computerizado. Las recomendaciones proporcionan los requisitos para la CPU de una estación determinada de entrenamiento: la velocidad de reloj, el bus, la fuente de alimentación, el sistema operativo, la memoria RAM, la memoria ROM, el adaptador gráfico, el monitor, el ratón, el teclado, el sistema de audio digital y de red.
3. **AGR 003 Digital Audio**: Este documento recomienda las pautas que promueven la interoperabilidad del audio digital. La interoperabilidad en este ámbito proporciona que el audio del curso pueda ser leído en diversos PC's con diferentes tarjetas de sonido de la estación de entrenamiento. También implica compatibilidad con diferentes formatos de audio.
4. **AGR 004 Operating/Windowing System**: Este documento proporciona una recomendación formal a la industria de la aviación para el sistema operativo usado y para las ventanas e iconos del curso.
5. **AGR 005 CBT Peripheral Devices**: Este documento proporciona las pautas a seguir para conseguir la interoperabilidad de los dispositivos o periféricos, tales como la entrada de video XY (una pantalla, un ratón, o un Trackball), y los drivers.
6. **AGR 006 Computer-Managed Instruction**: Este documento contiene las recomendaciones para lograr la interoperabilidad CMI en sistemas de ficheros locales. La interoperabilidad en este aspecto se refiere a la capacidad de un sistema CMI para manejar lecciones del CBT de distintos orígenes, así como los datos de intercambio entre sistemas CMI.
7. **AGR 007 Courseware Interchange**: Este documento recomienda las pautas para el intercambio de los elementos del curso del CBT. Estos elementos incluyen: Texto, gráficos, movimientos, audio, y lógica. Estas pautas abarcan: 1) los componentes principales de los datos del curso del CBT, y 2) los formatos de datos estándares para esos componentes.
8. **AGR 008 Digital Videos**: Este documento recomienda las pautas para la creación, la distribución, y el uso del vídeo digital del curso del CBT.
9. **AGR 009 Icon Standards**: Este documento contiene las recomendaciones para la interfaz de usuario y del diseño gráfico del curso del CBT.
10. **AGR 010 Web-Based Computer-Managed Instruction**: Contiene las recomendaciones sobre la interoperabilidad de las plataformas de formación y los cursos.

Aunque AICC ha publicado varias guías, la más seguida es la AGR 010 que trata de la interoperabilidad de las plataformas de formación y los cursos.

En esta guía se resuelven dos de los problemas fundamentales:

- La carga sin problemas en un LMS de cursos creados por terceros. Este objetivo se consigue definiendo el curso como una entidad totalmente independiente de la plataforma, y creando un sistema (ficheros) de descripción del curso que pueda ser entendido por cualquier plataforma.
- La comunicación entre el LMS y el curso, de tal modo que el curso pueda obtener información necesaria sobre el usuario, y después transmitir los resultados de las interacciones y evaluaciones realizadas por el mismo a la plataforma, con objeto de su almacenamiento y tratamiento estadístico.

Este segundo objetivo es logrado mediante la definición de un mecanismo de comunicación entre el curso y la plataforma, y un conjunto de datos mínimos que deben ser transmitidos del curso a la plataforma y viceversa. AICC describe dos mecanismos, uno más sencillo y extendido basado en el protocolo http, y otro mediante una API.

AICC cuenta con un programa de certificación (a diferencia de las otras iniciativas) y dispone de un "*test suite*" que le permite a las compañías verificar que sus productos son compatibles con otros sistemas que cumplen con las especificaciones AICC.

Actualmente la AGR 010 de AICC es el "estándar de facto" en la industria del e-learning.

ADL

- **Historia de ADL**

ADL (*Advanced Distributed Learning*) [ADL, 2006], es un programa del Departamento de Defensa de los Estados Unidos y de la Oficina de Ciencia y Tecnología de la Casa Blanca para desarrollar principios y guías de trabajo necesarias para el desarrollo y la implementación eficiente, efectiva y en gran escala, de formación educativa sobre nuevas tecnologías Web.

Este organismo recogió especificaciones de IMS, IEEE LTSC y AICC y las refundió y mejoró en su propio estándar: SCORM (*Sharable Content Object Reference Model* - Modelo de Referencia para Objetos de Contenidos

Intercambiables), cuya última versión es la 1.3, más conocida como SCORM 2004 [ADL, 2004].

De lo recogido por ADL de las distintas organizaciones se destacan los siguientes elementos:

- El sistema de descripción de cursos en XML de IMS.
- El mecanismo de intercambio de información mediante una API (Aplication Programing Interface) de AICC.
- Los metadatos de objetos provistos por el estándar LOM (Learning Objects Metadata) de IEEE.

- **Ámbito y propósito de ADL**

SCORM proporciona un marco de trabajo y una referencia de implementación detallada que permite a los contenidos y a los sistemas usar SCORM para comunicarse con otros sistemas, logrando así interoperabilidad, reusabilidad y adaptabilidad [ADL, 2004].

SCORM permite cubrir con suficientes garantías los aspectos siguientes: descripción de los contenidos; empaquetamiento y organización de los contenidos; presentación y secuenciación de los contenidos; y, por último, seguimiento del proceso de aprendizaje.

Sin embargo, la especificación de SCORM no cubre todos los aspectos del e-learning, por ejemplo, no especifica cómo es almacenada la información y qué informes son generados, qué modelos pedagógicos y de aprendizaje deben ser usados, o cómo la información del estudiante es recopilada.

- **Especificaciones de ADL**

Las especificaciones de SCORM están organizadas como “libros” separados (figura 2.8).

SCORM ha sido dividido en cuatro libros:

- **Libro 1: Scorm Overview**. Contiene una descripción general de la iniciativa de ADL, un análisis de SCORM, y un resumen de las especificaciones técnicas contenidas en las siguientes secciones.

- **Libro 2: Scorm Content Aggregation Model (CAM)**. Contiene una guía para identificar y agregar recursos dentro de un contenido de aprendizaje

estructurado. Este libro describe una nomenclatura para el contenido de aprendizaje. Define responsabilidades y requerimientos para construir agregaciones de contenidos, como pueden ser cursos, lecciones, módulos, etc. Contiene información sobre la creación de contenidos aplicando metadatos y el secuenciamiento y navegación en el contexto del empaquetamiento de los contenidos (*SCORM Content Packaging*). Este libro está fuertemente relacionado con el libro 3 (*Scorm Run-Time Environment*).

Los metadatos SCORM describen los diferentes componentes del *Scorm Content Aggregation Model* (Agregaciones de contenidos, Actividades, SCO's y Assets). Los metadatos es una forma de etiquetar esos componentes para facilitar su búsqueda.

Un paquete de contenido, en un sentido general, une objetos de contenido con la organización de los mismos, lo cual es descrito en un manifiesto. El manifiesto (imsmanifiest.xml) es la parte esencial de SCORM Content Packages y está definido en XML. Describe los contenidos del paquete y debe incluir una descripción de la estructura del contenido, aunque ésta es opcional.

Está basado en la especificación IMS Learning Resource Meta-data Information Model, la cual está basada, a su vez, en el IEEE LTSC Learning Object Metadata (LOM) Specification, que fue el resultado de un esfuerzo en conjunto entre el IMS Global Learning Consortium y ARIADNE.

- **Libro 3: Scorm Run-Time Environment (RTE).** Incluye una guía para lanzar contenidos y hacerles un seguimiento en un ambiente basado en Web de un entorno de ejecución que incluye: un protocolo específico para la ejecución de contenidos Web, un API entre el contenido y el LMS y un modelo de datos que define el flujo de datos intercambiado entre el entorno LMS y el contenido que se ejecuta en el entorno de ejecución.

  El *Launch* (Motor de presentación) define las relaciones entre los SCO's y el LMS. La API JavaScript de SCORM es una pieza de código que en un ambiente virtual de educación (*Virtual Learning Environment*, VLE) provee el navegador de un estudiante cuando éste requiere contenido e-learning de un objeto SCORM. El código captura acciones específicas del estudiante, por medio de instrucciones, en el contenido SCORM y las pasa por el VLE. Las acciones incluyen eventos tales como: cuán lejos ha llegado un estudiante a través de una pieza de contenido, puntuaciones en test y cuánto tiempo le llevó trabajar con el material.

Este libro es derivado del CMI001 *Guidelines for Interoperability* de la AICC. (Basado en los estándares de IEEE API 1484.11.2 y IEEE Data Model 1484.11.1)

- **Libro 4: Scorm Sequencing and Navigaton (SN).** Este libro describe cómo el contenido debe ser secuenciado a través del sistema junto con los eventos de navegación. El contenido puede ser descrito por un conjunto de actividades predefinidas, definidas durante el diseño de los contenidos. Este libro describe como un LMS de SCORM interpreta las reglas de secuenciamiento expresadas en un entorno de desarrollo y sus efectos en el *Run-Time Environment*.

  Describe las ramificaciones y el camino que siguen las actividades de aprendizaje, representadas mediante un Árbol de Actividad, basadas en los resultados de las interacciones del estudiante con los objetos de contenido y una estrategia de secuenciamiento. Un Árbol de Actividad es una estructura conceptual de actividades manejadas por el LMS propias de cada alumno. En SCORM, una actividad de aprendizaje hace referencia a objetos de contenidos que cursa el estudiante. Está basado en la especificación propuesta para el secuenciamiento de contenidos de IMS.

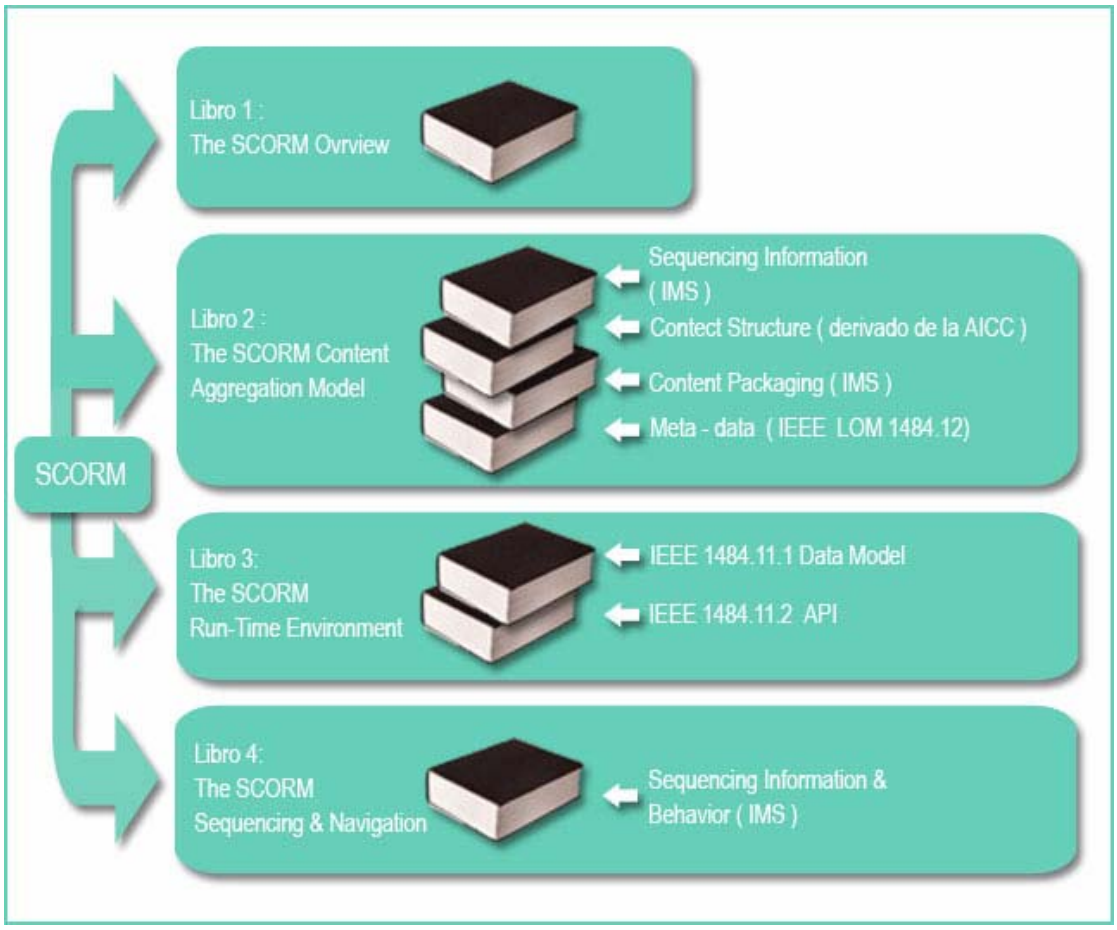

Figura 2.8. Organización de las especificaciones SCORM

SCORM reúne los esfuerzos de diferentes organizaciones para llegar a un modelo general que puede ser tomado cómo referencia para la creación de sistemas de aprendizaje. En la figura 2.8 se puede apreciar esta labor de integración y recolección, pues en ella se muestra qué organización ha tomado cada una de las aportaciones que propone ADL para SCORM.

ARIADNE

- **Historia de ARIADNE**

ARIADNE (*Alliance of Remote Instructional Authoring & Distribution Networks for Europe*) tiene su origen en un proyecto tecnológico del sector de "Telemática para la Educación y Enseñanza" para la investigación y desarrollo llevado a cabo por la Unión Europea y comenzado en el año 1996. El proyecto se centra en el desarrollo de herramientas y metodologías para la generación, gestión y reutilización de los componentes pedagógicos basados en computadores y en la educación con soporte telemático. La validación de los conceptos y herramientas se lleva a cabo en diversos entornos (tanto académicos como corporativos) en toda Europa. El núcleo central del proyecto es la distribución de una librería con componentes digitales reutilizables para la educación, llamado KPS (*Knowledge Pool System*). Se trata de una red europea de recursos educativos distribuidos, alrededor de la cual se han creado una serie de herramientas [ARIADNE, 2006].

- **Ámbito y propósito de ARIADNE**

La contribución más significativa del proyecto ARIADNE en lo concerniente a los estándares educativos, ha sido el conjunto de especificaciones educativas de metadatos ARIADNE (el "encabezamiento pedagógico"), que es uno de los componentes principales del estándar IEEE/LTSC LOM.

El proyecto ARIADNE establece las siguientes categorías de usuarios [ARIADNE, 2006]:

- Autores que crean nuevo material pedagógico: bien empleando las herramientas de software pedagógico facilitadas por ARIADNE, o bien reutilizando material existente en el repositorio KPS.
- Educadores que almacenan e indexan su material pedagógico en el repositorio KPS.
- Motores pedagógicos, que crean o modifican cursos, empleando el editor de currículos.

- Gestores de cursos, que administran los cursos empleando las funciones del sistema ARIADNE.
- Administradores del KPS, que emplean el conjunto de funcionalidades técnico - administrativas del KPS.
- Estudiantes de las categorías expuestas, que se ciñen al sistema de currículos KPS gracias a la interfaz de aprendizaje proporcionada por ARIADNE. (individualizada y siempre actualizada -"*up-to-date*"-).

Actualmente, el mantenimiento, uso, y desarrollo adicional en la mejora de las herramientas ARIADNE son competencias de la fundación ARIADNE, la cual surge al concretarse el plan de explotación de ARIADNE y formalizarse el soporte que éste recibe de la Unión Europea y Suiza.

- **Características generales**

Las principales características que presenta ARIADNE son las siguientes:

- **Usuarios implicados:**
    - Autores de documentos pedagógicos: Como universidades, directores de educación, estudiantes, etc.
    - Productores y Administradores de cursos de aprendizaje: como educadores, directores de educación, etc.
    - Usuarios finales: Estudiantes.

- **Tecnologías y/o enfoques usados:**
    - Compartir y reutilizar componentes pedagógicos a través de la indexación o almacenamiento de los mismos en KPS.
    - Emplear canales telemáticos adecuados, ajustándose a las diferentes situaciones.
    - Facilitar el uso de los componentes pedagógicos, currículos estructurados y visualización individual de los cursos.

- **Ventajas para la ciudadanía:**
    - Mejores esquemas de formación continua.
    - Fácil acceso a la formación para las categorías sociales menos favorecidas.

- **Beneficios para los usuarios de las aplicaciones:**

    - Estudiantes: Escenarios de aprendizaje atractivos y mucho más efectivos.
    - Autores del material pedagógico: Mejor productividad y nuevas filosofías de colaboración.
    - Investigadores: Mejores comunicaciones y esquemas de colaboración.

- **Beneficios previstos para las industrias europeas:**

    - Niveles de enseñanza continua mejores, más rápidos y mucho más económicos.

- **Contribución a las políticas de la Unión Europea:**

    - Posible factor de homogenización de las políticas de educación y enseñanza en toda Europa. Promoción de la colaboración entre los educadores europeos y simplificación de la comunicación entre los estudiantes europeos.

ISO

La organización internacional de estandarización (ISO, *International Organization for Standardization*) tiene un comité (JTC1) especializado en tecnologías de la información, al que a su vez, pertenece un subcomité (SC36) que se encarga de elaborar estándares relacionados con las tecnologías de la información para educación y formación [ISO-SC36, 2006]. Actualmente, este subcomité, cuenta con la colaboración de 28 países, de los cuales 22 son miembros “P”, es decir, miembros participativos involucrados en actividades técnicas y con derecho a voto, entre ellos España con AENOR; y 6 son miembros “O”, sólo observadores, que pudiendo estar involucrados en actividades técnicas, no votan. El subcomité SC36 está compuesto por siete grupos de trabajo activos más un grupo de compensación. A continuación se presenta un breve resumen del objetivo y tareas en la que están involucrados cada uno de estos grupos de trabajo:

- **SC36/WG1 - Vocabulary**

Este grupo se encarga de definir los principales términos asociados a las aplicaciones educativas, presentando definiciones y términos, los cuales denotan conceptos relevantes en este campo, constituyendo un punto crucial para el desarrollo de estándares.

Su origen se debe, en principal medida, al reciente crecimiento de los productos relacionados con el estudio, la educación y el aprendizaje basados en la tecnología, ya que conduce al empleo de nombres diferentes para el mismo (o similar) concepto. La necesidad de interoperabilidad y reutilización de los productos y componentes utilizados en ámbitos educativos, requiere cierta unificación, para que de esta manera sea posible identificar y referirse a estos conceptos y productos mediante la utilización de una terminología común.

Los principales interesados que podrían beneficiarse del estándar terminológico propuesto por este grupo son: los productores de estándares en el campo de la tecnología asociada al aprendizaje, los productores de herramientas, productos y servicios educativos, así como agencias que ofrecen servicios en este campo, compradores y usuarios de los productos (educadores, tutores y alumnos).

La terminología estandartizada facilitará el entendimiento mutuo y la comunicación, mientras que el empleo de estándares en la descripción de productos promoverá su utilización mundial.

Entre las principales ventajas que se pueden destacar están las relacionadas con la uniformidad en la descripción de los productos, sus características, componentes y requerimientos, facilitando el empleo internacional del producto y la traducción de sus descripciones en otros idiomas.

- **SC36/WG2 - Collaborative Technology**

Este grupo encuentra su justificación en la evolución gradual que han sufrido los sistemas de aprendizaje, los cuales han pasado de ser sistemas independientes a conformar ambientes educativos distribuidos.

Su propósito es especificar la estructura y los componentes de colaboración del lugar de trabajo, (tales como áreas de discusión, áreas de actividades de colaboración, espacios de trabajo para el aprendizaje electrónico, pizarras compartidas, objetos de usuario, etc.) así como sus cualidades y los enlaces que se producen entre ellas.

Este proyecto considera agentes en un ambiente de aprendizaje colaborativo. Los agentes pueden representar a alumnos, o pueden ser agentes independientes como por ejemplo, los que manejan las herramientas compartidas, los que supervisan la comunicación entre los alumnos o entre otros agentes, etc.). El proyecto debe tratar las características específicas de comunicación de los agentes que están colaborando en un ambiente de aprendizaje, por lo que será necesario que se especifique el

protocolo de comunicación agente-agente, así como el protocolo de sincronización existente entre los distintos sistemas de aprendizaje.

- **SC36/WG3 - Participant Information**

El alcance de este grupo reside en la información asociada a los participantes en lo que a los sistemas de aprendizaje se refiere. La información puede incluir elementos relacionados con la cultura, la lengua, los dispositivos, las preferencias cognitivas, las capacidades o las interfaces persona-ordenador.

Su propósito es proporcionar un modelo de datos y los enlaces a las categorías específicas de información. La información personalizada puede ser utilizada, por ejemplo, para adaptar los sistemas educacionales a las necesidades de cada usuario.

- **SC36/WG4 - Management and Delivery of Learning, Education, and Training (MDLET)**

Este grupo trabaja en el estándar ISO/IEC 2382, que representa solamente una parte de todo el conjunto de elementos que componen el modelo de datos necesario para importar contenido dentro de un LMS, permitiendo además establecer las comunicaciones entre el contenido y el LMS.

Esta parte es la encargada de definir el *framework* de los modelos de datos y enlaces que permitirán importar dicho contenido en el LMS, así como establecer su comunicación. Estos modelos de datos y los enlaces serán definidos en las diferentes partes del estándar, difiriendo del trabajo realizado en el proyecto IEEE P1484.11 en que adopta un mayor rango de pedagogías educativas y añade soporte para el contenido empaquetado, exigencia identificada de la práctica comercial.

- **SC36/WG5 - Quality Assurance and Descriptive Frameworks**

Este grupo trata de describir y caracterizar procesos, componentes y atributos relacionados con la calidad y la arquitectura de entornos tecnológicos en el campo del aprendizaje.

El propósito de este grupo es proveer de un *framework* común con definiciones que incluyan conceptos, especificaciones, términos y definiciones a describir, especificando y entendiendo las características y métricas de calidad.

- **SC36/WG6 - International Standardized Profiles (ISP)**

Un ISP es un documento internacionalmente reconocido y armonizado que identifica un estándar o un grupo de ellos, junto con las opciones y parámetros necesarios para acoplar una función o conjunto de funciones.

Por otro lado, los perfiles definen una combinación de estándares que en conjunto funcionan como una función "bien-definida". Los perfiles estandartizan el uso de opciones y otras variaciones en los estándares, y proporcionan una base para el desarrollo de sistemas de pruebas internacionalmente reconocidas. Los ISPs no se generan simplemente para "legitimar" una opción particular de estándares y opciones, sino para promover la interoperabilidad real en sistemas.

- **SC36/WG7 - Culture/Language/Human-Functioning Activities**

Este grupo está íntimamente relacionado con el grupo de trabajo que garantiza la accesibilidad para DC (*Dublin Core*) [DC, 2006]. Todo el trabajo realizado hasta el momento por el grupo de trabajo de DC, ha sido realizado en colaboración con otros grupos que también trabajan en el campo de la accesibilidad de metadatos. El trabajo comenzó con dos perfiles de metadatos desarrollados por IMS: uno encargado de describir las necesidades y las preferencias de usuarios individuales y otro para describir los recursos que ellos podrían querer usar. Este trabajo fue adoptado por este grupo y está actualmente en su etapa final antes de convertirse estándar ISO.

Los grupos de trabajo presentados anteriormente trabajan en la definición de los estándares que se resumen en la tabla 2.5.

Estándar	Descripción
ISO/IEC 2382	Information technology -- Vocabulary -- Part 36: Learning, education, and training
ISO/IEC 19780	Information Technology -- Learning, Education and Training -- Collaborative Technology -- Learner to Learner Interaction Scheme.
ISO/IEC 19788	Information Technology -- Metadata for Learning Resources
ISO/IEC 23988	A code of practice for the use of information technology (IT) in the delivery of assessments
ISO/IEC 24725	Information technology -- Learning, education and training -- Profiles of standards and specifications
ISO/IEC 24751	Individualized Adaptability and Accessibility in E-learning, Education and Training
ISO/IEC 19796	Information technology -- Learning, education and training -- Quality management, assurance and metrics

Tabla 2.5. Estándares en los que trabaja ISO/IEC JTC1/SC36

2.1.4 Arquitecturas

En este apartado se presentan algunas de las arquitecturas más importantes que sirven de base en la construcción de sistemas de e-learning.

IEEE [2001] ha desarrollado un estándar de definición de arquitecturas de e-learning con los siguientes objetivos:

- Proporcionar un marco general que ayude a entender los existentes y futuros sistemas.
- Promover la interoperabilidad y portabilidad identificando las interfaces críticas.
- Incorporar un horizonte tecnológico de, al menos, 5 o 10 años adaptable a nuevas tecnologías.

El estudio del actual campo de las aplicaciones de e-learning supone un reto por estar en constante crecimiento, por la cantidad de posibilidades que debe ofrecer y por su necesidad de flexibilidad y adaptación a nuevas herramientas (figura 2.9).

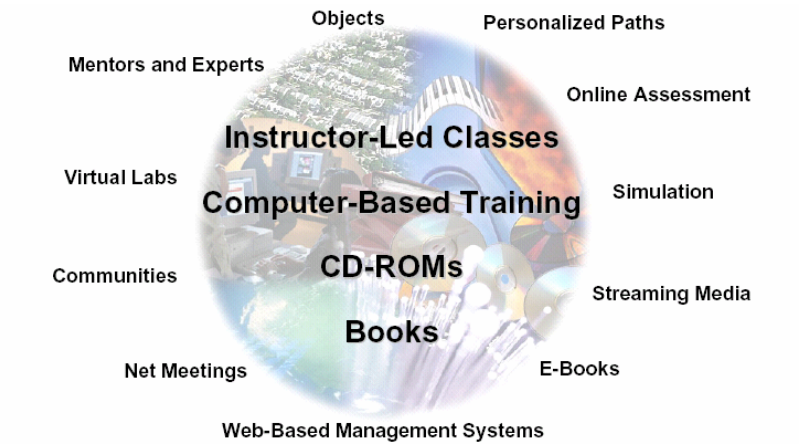

Figura 2.9. Nuevas herramientas de e-learning

En todo caso hemos constatado que casi todos los sistemas de e-learning deben estar soportados por una arquitectura que cumpla con las siguientes características (sintetizadas ya por CISCO en [Cisco, 2001]):

- **Abierta**. Debe soportar la interoperabilidad entre distintos proveedores de soluciones y basada en los estándares de organizaciones como AICC, IMS, ADL e IEEE.

- **Escalable**. Sus funciones deben poder ser ampliadas cuando sea necesario.

- **Global**. Debe poder ser utilizada en cualquier lugar del mundo y en cualquier momento con igual facilidad.

- **Integrada**. Debe integrarse con distintas infraestructuras de red y otras aplicaciones de seguridad, recursos humanos, etc.

- **Flexible**. Debe poder adaptarse a nuevos requisitos y procesos, nuevas tecnologías y nuevos proveedores de soluciones.

- **Adaptable**. En un mundo en constante desarrollo, debe ser de rápida y fácil implantación en organismos, empresas y entidades educativas.

Otros autores o entidades añaden distintas justificaciones para el diseño de arquitecturas de e-learning. Por ejemplo, Rosenberg [2001] se basa en la necesidad de flexibilidad y adaptabilidad de estos sistemas a los cambios del entorno, de las tecnologías y de los contenidos; la empresa de creación de contenidos SkillSoft [2006], se basa en la necesidad de integrar tecnología, contenidos y servicios; por su parte, Microsoft, ya en 1999, se basaba en la necesidad de obtener sistemas integrados, controlables, fáciles de usar, adaptables a las nuevas tecnologías y escalables en función de las necesidades de crecimiento [Microsoft, 1999].

Presentamos a continuación aquellas arquitecturas que nos han parecido más relevantes en los últimos años; de forma sintética se describirán únicamente sus características principales.

2.1.4.1 Arquitectura LTSA de IEEE

IEEE ha realizado un gran esfuerzo para crear un estándar para el desarrollo de arquitecturas de sistemas de e-learning. Su última versión es la IEEE P1484.1/D9 [IEEE, 2001], en este documento establece una arquitectura de alto nivel para sistemas de e-learning y sus componentes llamada LTSA (*Learning Technology Systems Architecture*).

En su arquitectura, IEEE especifica cinco niveles, aunque solo establece como obligatorio en este estándar el nivel 3 (*System Components*). En la figura 2.10 podemos ver gráficamente estos niveles.

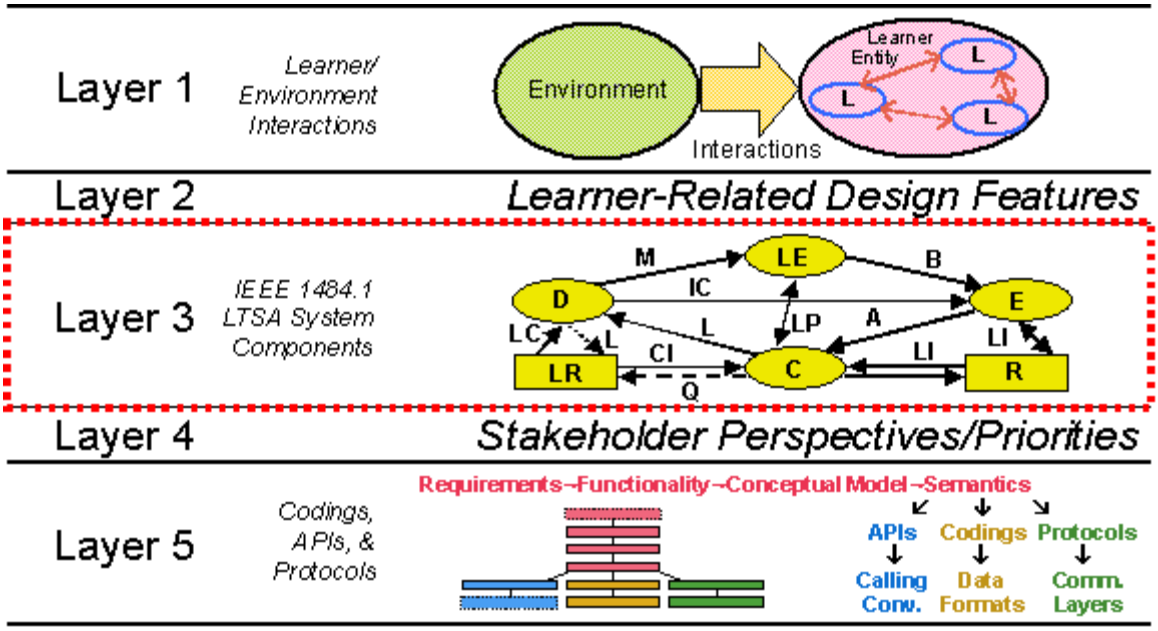

Figura 2.10. Arquitectura LTSA de IEEE

- **Nivel 1. Learner and Environment Interactions.** Es el nivel más genérico: el alumno tiene conocimientos nuevos o diferentes después de una experiencia educativa. Hay una interrelación entre el alumno y el sistema de e-learning. Se refiere a la adquisición, transferencia, intercambio, descubrimiento, etc., de conocimientos o información a través de la interacción con el entorno.

- **Nivel 2. Learner-Related Design Features.** Relacionado con los efectos de la naturaleza humana del usuario sobre el diseño de los sistemas de aprendizaje informáticos.

- **Nivel 3. System Components.** Describe los componentes básicos de la arquitectura.

- **Nivel 4. Implementation Perspectives and Priorities.** Describe cómo una gran variedad de organismos, industrias, empresas y entes educativos analizan los sistemas de e-learning, en cuanto a: verificación y validación de los principales componentes; importancia de cada componente en función de las distintas perspectivas y prioridad entre los niveles.

- **Nivel 5. Operational Components and Interoperability.** Proporciona una visión global de cómo componentes e interfaces que permiten la interoperabilidad (codificación, APIs y protocolos) puede estar relacionados con la arquitectura de sistemas de e-learning.

El desarrollo del estándar se centra en el único nivel normativo existente: el nivel tres representado en la figura 2.11.

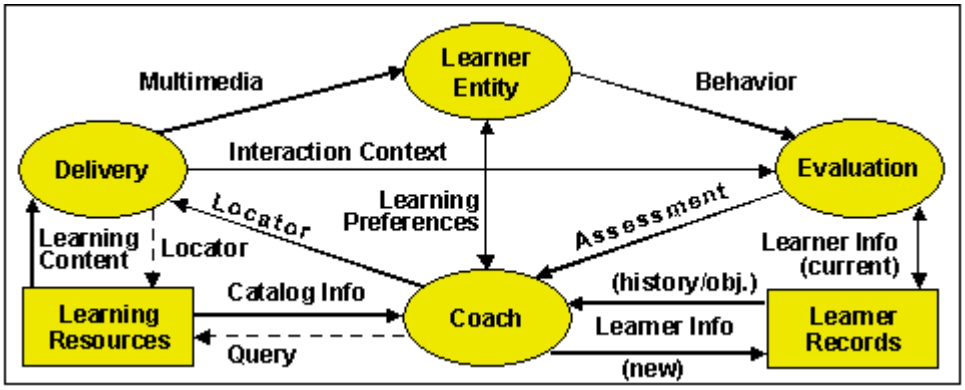

Figura 2.11. Nivel 3- System Components

El Sistema de Componentes del LTSA identifica los interfaces críticos de interoperabilidad para sistemas de e-learning.

El propósito de este estándar es describir y entender los componentes generales identificando funciones generales. Las implementaciones reales de sistemas de e-learning pueden no corresponder componente a componente con este estándar, pero conceptualmente deben realizar las mismas funciones. Por ejemplo, por motivos comerciales, las funciones de evaluación, entrega y entrenamiento pueden agruparse en la misma herramienta aunque conceptualmente sean componentes separados.

En la figura 2.11 podemos ver tres tipos de componentes:

- Procesos (con forma de elipse): Alumno, Evaluación, Entrenamiento y Entrega.

- Almacenajes (rectángulos): Registro de Alumnos y Catálogo de Recursos de Aprendizaje.

- Flujos (flechas): Preferencias de Aprendizaje, Comportamiento, Información de Conocimientos, Información del Alumno, Consulta, Información del Catálogo, Localizador, Contenidos, Multimedia y Contexto de Interacción.

De forma resumida, la operativa de los sistemas que implementan esta arquitectura es la siguiente:

1. Los estilos, estrategias y métodos de aprendizaje los definen los alumnos mediante preferencias de aprendizaje.

2. Los alumnos son observados y evaluados a través de sus interacciones con el sistema.
3. La evaluación produce información sobre el alumno y su conocimiento.
4. La información sobre el alumno se almacena en la base de datos histórica de alumnos.
5. El proceso de entrenamiento revisa la información sobre el alumno y su conocimiento para establecer objetivos de aprendizaje futuros.
6. El proceso de entrenamiento busca los recursos de aprendizaje necesarios para cada contenido.
7. El proceso de entrenamiento extrae los localizadores del catálogo de recursos de aprendizaje y los pasa al proceso de entrega.
8. El proceso de entrega extrae los contenidos de aprendizaje y los transforma en una presentación interactiva multimedia.

Cada componente viene explicado a continuación. Los procesos se describen definiendo entradas, funcionalidad y salidas. Los almacenes se describen en función de los datos que contienen y los métodos de almacenamiento, búsqueda y actualización. Los flujos se describen mediante el tipo de información que fluye y el tipo de conectividad (unidireccional o bidireccional, estática o dinámica).

- **Alumno**

Entidad que representa al alumno. Puede ser una persona, un grupo de personas vistas como alumnos individuales o un grupo de personas vistas como alumnos que trabajan de forma colaborativa.

Sus relaciones con el resto de entidades son:

- Entrada: recibe las presentaciones multimedia.
- Salida: da datos sobre su comportamiento.
- Entrada/Salida: define, junto con el Proceso de Entrenamiento, sus preferencias de aprendizaje.

- **Evaluación**

Proceso con el que se mide al Alumno.

Sus relaciones con el resto de entidades son:

- Entrada: recibe el conocimiento observable del Alumno y el contexto de interacción con la Entrega.
- Salida: envía información sobre el conocimiento del Alumno al Proceso de Entrenamiento.
- Entrada/Salida: se puede recuperar y almacenar información referida a la situación actual del Alumno en el registro de alumnos.

- **Entrenamiento**

Proceso en el que se buscan y seleccionan los contenidos de aprendizaje que deben entregarse en función de datos recibidos sobre el alumno y los recursos de aprendizaje disponibles.

Sus relaciones con el resto de entidades son:

- Entrada: recibe los datos sobre el conocimiento del Alumno obtenidos del proceso de Evaluación, los datos pasados, presentes y futuros del Alumno del registro de alumnos y la información resultado de sus búsquedas en el catálogo de recursos de aprendizaje.
- Salida: envía datos sobre el Alumno al registro de Alumnos, envía al catálogo de recursos de aprendizaje las búsquedas con las que encontrar los recursos apropiados y envía el resultado de estas búsquedas, en forma de localizadores, al proceso de Entrega.
- Entrada/Salida: recibe y negocia las preferencias de aprendizaje del Alumno.

- **Entrega**

Proceso en el que se transforma la información contenida en los recursos de aprendizaje que el proceso de Entrenamiento ha decidido utilizar. Esta transformación produce los contenidos educativos que se presentan al Alumno, que pueden estar en formatos muy variados: presentaciones y preguntas, sistemas inteligentes de tutorización, videoconferencias, modelos conceptuales, etc.

Sus relaciones con el resto de entidades son:

- Entrada: recibe, desde el proceso de Entrenamiento, localizadores que deben ser recuperados y los recursos de aprendizaje correspondientes desde el catalogo.
- Salida: envía localizadores al catálogo para recuperar recursos de aprendizaje, presenta los contenidos relativos a estos recursos en forma multimedia al Alumno y envía al proceso de Evaluación el contexto de interacción.

- **Registro de Alumnos**

Constituye la base de datos de Alumnos, con datos pasados, presentes o futuros sobre los objetivos, logros, actividades, etc. del alumno. Los procesos de Evaluación y de Entrenamiento pueden recuperar y almacenar datos en esta base de datos.

- **Catalogo de Recursos de Aprendizaje**

Constituye la base de datos de recursos de aprendizaje; incluye presentaciones, tutorías, herramientas, experimentos de laboratorio y cualquier otro tipo de material educativo. El proceso de Entrenamiento puede hacer búsquedas y recuperar información sobre lo buscado en esta base de datos. El proceso de Entrega, a través de los localizadores obtenidos por el proceso de Entrenamiento, puede obtener los contenidos que desee del catálogo de recursos de aprendizaje.

- **Preferencias de Aprendizaje**

Flujo bidireccional entre el Alumno y el proceso de Entrenamiento, en el que se incluyen datos relativos a las preferencias culturales del alumno y sus limitaciones físicas o mentales. En estos datos también pueden influir entes externos con autoridad, como padres, profesores, entes educativos, etc.

- **Comportamiento**

Flujo desde el Alumno al proceso de Evaluación en el que se refieren todas las actividades que ha realizado el alumno: respuestas escritas (o directamente con voz), opciones seleccionadas, etc.

- **Información de Conocimientos**

Flujo de información que el proceso de Entrenamiento recibe del proceso de Evaluación con datos sobre la situación actual del alumno.

- **Información del Alumno**

Flujo bidireccional (recuperación de datos o almacenamiento de datos) entre el registro de alumnos y los procesos de Evaluación o Entrenamiento. En este flujo de información se pasan datos pasados, presentes o futuros sobre los objetivos, logros, actividades, preferencias, etc. del alumno.

- **Consulta**

Flujo desde el proceso de Entrenamiento hacia el Catálogo de recursos de aprendizaje; contiene los criterios de búsqueda necesarios para poder localizar los recursos de aprendizaje que necesita.

- **Información del Catalogo**

Flujo que el proceso de Entrenamiento obtiene del catálogo de recursos de aprendizaje como resultado de sus búsquedas en el catálogo; contiene los datos que

describen los recursos de aprendizaje obtenidos (metadatos) y sus localizadores (identificativos que apuntan al recurso).

- **Localizador**

Flujo de información desde el proceso de Entrenamiento hacia el proceso de Entrega (pidiéndole que ponga a disposición el contenido correspondiente) o desde el proceso de Entrega al catálogo de recursos de aprendizaje (solicitándole que le envíe el recurso de aprendizaje que identifica el localizador). La información que contiene es un identificativo o enlace a un recurso de aprendizaje, por ejemplo una dirección URL, etc.

- **Contenidos**

Flujo de información desde el catálogo de recursos de aprendizaje al proceso de Entrega, en el que se envía el recurso solicitado.

- **Multimedia**

Flujo de información desde el proceso de Entrega al Alumno, en el que se le presentan los contenidos educativos en diversas formas: gráfico, texto, video, audio, etc.

- **Contexto De Interacción**

Flujo de información desde el proceso de Entrega hasta el proceso de Evaluación, que proporciona la información necesaria al proceso de Evaluación para interpretar los datos de comportamiento. Esta información incluye datos sobre el entorno de aprendizaje que ha utilizado el Alumno.

2.1.4.2 Arquitectura de ARIADNE

Las herramientas que facilita el proyecto ARIADNE [2005] para el desarrollo de aplicaciones de aprendizaje se sostienen sobre la siguiente estructura (Figura 2.12) .

Los elementos más importantes de la arquitectura son:

- Gestor de KPS: Forma parte de la capa de datos, en la cual se almacenan los LO ("*Learning Objects*", objetos de aprendizaje) y los Metadatos que los describen.
- SILO (*Search and Index Learning Objects*): Es una herramienta con la que se busca ofrecer una forma simple de acceso al KPS.
- WEBLE (*Web-Based Learning Environment*): Proporciona herramientas para la gestión de los cursos y acceso al KPS.
- KPS Client: Es la herramienta de acceso al KPS.

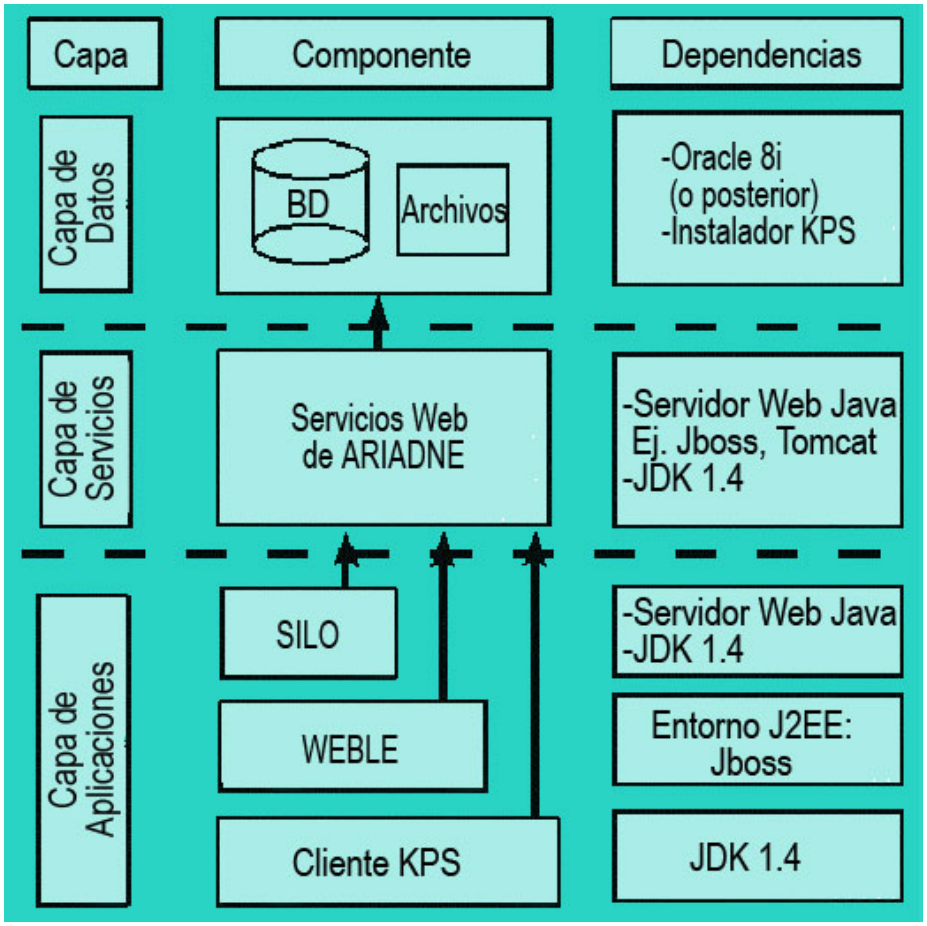

Figura 2.12. Arquitectura del sistema ARIADNE

En esta arquitectura se pueden observar dos tipos de dependencias diferentes:

- **Dependencias entre los componentes ARIADNE**: Estas dependencias se representan con flechas negras. Por ejemplo, las dependencias del componente SILO con los servicios Web. Sin embargo, esto no quiere decir que SILO no pueda instalarse sin haber instalado previamente los servicios Web. Actualmente existen múltiples nodos en la red ARIADNE que albergan estos servicios. Esto quiere decir (desde el punto de vista práctico) que, por ejemplo, se puede instalar un servidor Tomcat, instalar SILO y configurarlo para emplear los servicios Web que Tomcat proporciona.
- **Dependencias de los componentes ARIADNE y software de terceras partes**: El conjunto de recuadros de la parte derecha del esquema de la figura 2.12 representan el conjunto de software necesario para instalar los componentes ARIADNE. El componente WEBLE, por ejemplo, necesita ser instalado en un servidor Web que soporte J2EE; SILO tan sólo necesita un servidor Web Java como puede ser Tomcat.

Veremos con más detalle sus características en el capítulo tres dedicado al planteamiento del problema.

2.1.4.3 Arquitectura de CISCO

Cisco ha desarrollado un modelo de arquitectura para sistemas de e-learning, caracterizado por ser un sistema abierto, escalable, global, integrado y flexible [Cisco, 2001]. Su visión es mucho más comercial que la del IEEE, como corresponde a su carácter de empresa privada, y va más encaminada a una implementación concreta, la suya propia. En todo caso, vamos a presentarla brevemente por el interés evidente de su nivel intermedio.

La arquitectura completa se divide en tres niveles tal como se ve en la figura 2.13.

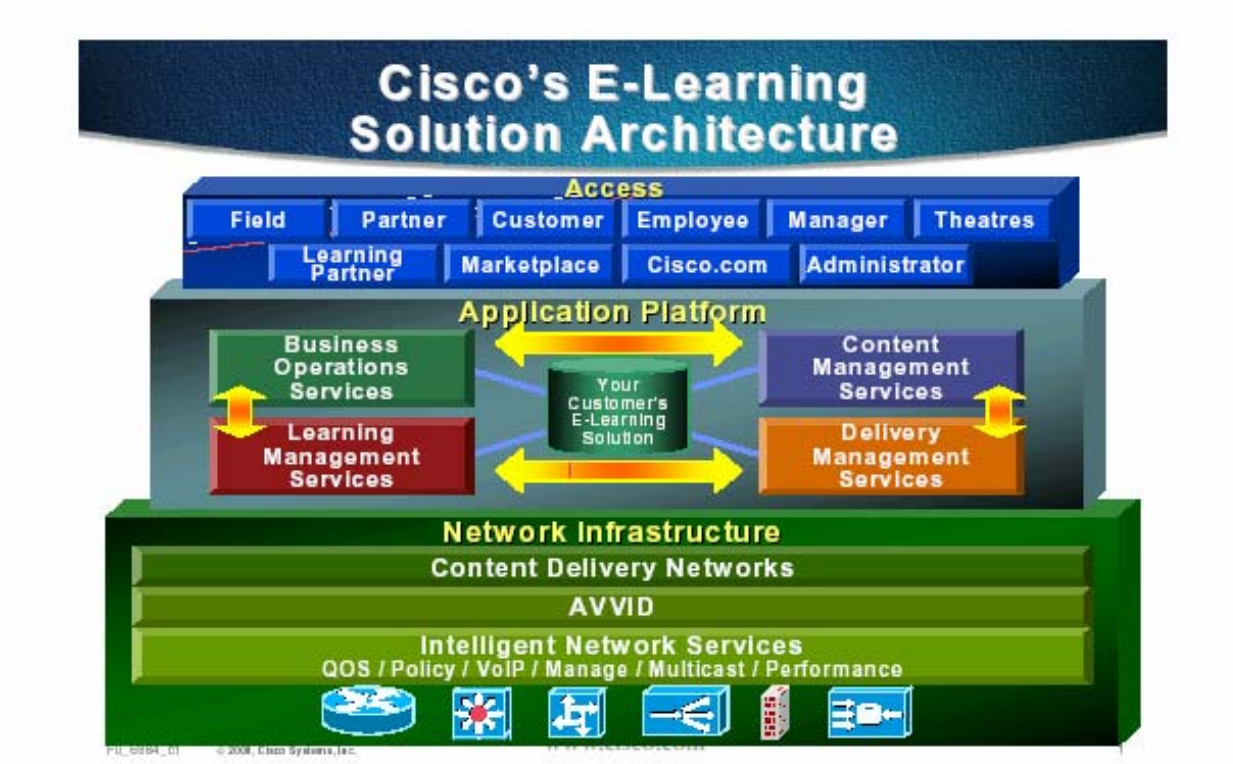

Figura 2.13. Arquitectura de Cisco Systems

Cada uno de los niveles tiene la siguiente funcionalidad:

Nivel 1. The Underlying Network Infrastructure

Este es el último nivel o nivel de red en el que se apoya el nivel intermedio. Según [Cisco, 2000], la arquitectura de red propuesta tiene por objeto proporcionar disponibilidad, seguridad y buenos resultados en tiempo y calidad de acceso; integrando tecnologías de:

- IP Multicasting, para transmitir datos multimedia desde distintos servidores a muchos clientes diferentes a la vez y en cualquier momento.

- Quality of Service (QoS), para proteger la transmisión de datos crítico mientras se realizan tareas de e-learning muy costosas en cuanto a recursos de red, como transmisión de videos, clases virtuales, etc.
- Seguridad, para proteger los datos y aplicaciones de accesos indebidos: gestión de *password*, encriptación de datos a transmitir, utilización masiva de recursos y tecnología VPN (*Virtual Private Network*) que gestiona la seguridad de redas privadas dentro de la red general.
- Mecanismos de entrega y distribución de contenidos:
    - CDN (*Content Delivery Network*), que utilizando la tecnología de red SODA (*Self-Organizing Distributed Architecture*) permite a los usuarios recibir localmente datos procedentes de servidores de intranet y extranet optimizando los accesos.
    - CDM (*Content Distribution Manager*), que maneja la distribución de los contenidos seleccionando los servidores por proximidad física y disponibilidad y según las restricciones establecidas por el administrador de la red.

Es un nivel excesivamente cercano a las infraestructuras físicas y que carece de interés a los efectos de esta tesis, si bien la importancia de sus propuestas es más que patente en entornos que dependen en gran medida del correcto funcionamiento de la tecnología.

Nivel 2. The Application Blueprint

Este es el nivel intermedio que realiza la gestión propia de e-learning: contenidos, estudiantes, cursos, etc. Es un modelo modular donde las tareas y responsabilidades se dividen en áreas funcionales relacionadas e integradas. Según se desarrolla en [Cisco, 2001], estos módulos son los siguientes:

- **Operaciones de Negocio**-BOS (*Business Operations Services*). Está formado por un conjunto de herramientas y aplicaciones integradas que soportan los siguientes servicios:
    - Gestión de la efectividad del aprendizaje y de la experiencia del alumno.
    - Soporte a las consultas *on-line*, las llamadas gratuitas de apoyo y la distribución por e-mail.
    - Servicio de ayuda *on-line* en las aplicaciones.
    - Gestión de la seguridad de acceso aplicaciones y contenidos.
    - Generación de informes de progreso, utilización y resultados.
    - Análisis de necesidades: *Gap Analysis* mide la diferencia entre los conocimientos necesarios de un estudiante según sus competencias y sus conocimientos reales; y *Cost Model Analysis* que, extrapolando

datos de los resultados anteriores, determina la clase de contenidos que deben ser creados en otros módulos.

- **Gestión de Contenidos**-CMS (*Content Management Service*). Permite verificar, registrar utilizando metadatos, ensamblar, manejar y publicar contenidos. Está formado por los siguientes submódulos:
  - *Workflow Application*. Aplicación de flujo de datos de gestión de contenidos, en la que es posible personalizar el flujo de datos según el grupo de trabajo que la utilice.
  - *Authoring Tool Integration Service*. Permite que distintos tipos de contenidos: texto, test, gráficos, etc. se integren en cualquier nivel jerárquico de la estructura de contenidos.
  - *Registry Services*. Almacena la ubicación, los metadatos descriptivos y la estructura de metadatos asociada de cada contenido. Físicamente pueden almacenarse en el *Content Storage System* o en otra ubicación segura.
  - *Object Mining Service*. Localiza contenidos mediante distintos criterios de búsqueda. Aplicación muy útil para poder aplicar reusabilidad.
  - *Assembler*. Permite, utilizando los resultados del módulo anterior, crear plantillas que son registradas y almacenadas. Permite ensamblar contenidos utilizando estas plantillas como base.
  - *Content Storage Services*. Proporciona la gestión de almacenamiento físico de los contenidos: manejo de versiones, bloqueos, histórico, informes, etc.
  - *Publishing Services*. Cuando un curso está listo, proporciona los datos para su puesta a disposición que hará el módulo DMS.

- **Gestión de la Demanda**-DMS (*Demand Management Services*). Se encarga de poner a disposición de los alumnos los cursos terminados. Está formado por un conjunto de herramientas y aplicaciones integradas que soportan los siguientes servicios:
  - Presentar al alumno los contenidos de un curso en función de una parametrización personalizada.
  - Manejar la distribución de contenidos en función de las reglas establecidas por el administrador.
  - Controlar los contenidos que se distribuyen y en qué forma. Estos datos se pasan al módulo LMS que es el que los utiliza para futuras peticiones e histórico.

- **Gestión del Aprendizaje**-LMS (*Learning Management Services*). Es el módulo al que acceden los alumnos y que lleva el control de sus acciones. Está formado por los siguientes submódulos:
    - *Personalization*. Crea planes de enseñanza personalizados en función del perfil del usuario y de las preferencias personales.
    - *Search/Browse*. Permite realizar búsquedas y consultas en el catálogo de cursos.
    - *Registration*. En unión al módulo *Search/Browse* permite registrarse en un curso y gestionar las listas de espera, notificaciones, cambios de programación, etc.
    - *Learner Tracking*. Contiene la historia del aprendizaje de cada alumno: cursos realizados y situación en los cursos actuales. Estos datos le permiten proponer cursos futuros en función del perfil de cada alumno.
    - *E-Commerce*. Junto con el módulo *Search/Browse* permite realizar el pago de los cursos proporcionando varios métodos de pago y gestión de datos financieros.
    - *Assessment*. Analiza para cada estudiante su *pre-assessment* (diferencia entre lo que sabe y lo que debería saber para una determinada competencia) y su *post-assessment* (confirmación de su conocimiento real). Con estos datos personaliza el material de estudio y informa comprensivamente del currículum del alumno, para seguimiento personal o de la comunidad educativa.
    - *Manager's Toolkit*. Maneja la función de gestor educativo que puede aprobar o denegar altas en cursos, añadir o quitar alumnos, seguir el progreso del aprendizaje, etc.
    - *Survey*. Recoge datos sobre la satisfacción de los usuarios para enviárselos al módulo BOS.
    - *Resource Management*. Programa y asigna la utilización de recursos: equipos informáticos, profesores, clases, eventos virtuales, etc.

Nivel 3. The Access Layer.

Este es el nivel más alto o nivel de acceso a la aplicación, donde el tipo de persona o entidad que accede puede ser muy variado. Este nivel proporciona la posibilidad de definir diferentes portales de acceso personalizados.

2.1.5 Conclusiones

De todo lo expuesto anteriormente debemos destacar las características de los sistemas de aprendizaje, tanto los LMS como los LCMS, y sus diferencias; la definición

de objeto de aprendizaje y de su evolución en un objeto de aprendizaje reutilizable; y la posibilidad de utilizar repositorios digitales para almacenar objetos de aprendizaje y hacer que sean compartidos y por tanto reutilizables.

Como se comentó en un apartado anterior, uno de los mayores problemas a los que se enfrenta hoy en día la industria del e-learning es la interoperabilidad entre distintos sistemas de aprendizaje que quieren compartir su contenido en forma de objeto de aprendizaje reutilizable.

Para que esta interoperabilidad se pueda dar, es necesario que los estándares se apliquen desde la creación de los objetos de aprendizaje hasta la construcción y utilización de las herramientas que dan soporte al e-learning. Sin embargo, no hay ningún estándar que garanticen los objetivos de accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales docentes basados en las redes.

Actualmente, el uso de estándares en e-learning se enfrenta a muchos y variados problemas, entre los que destacamos los siguientes:

- Los estándares deben abarcar todo el proceso, pasando por captación de contenidos, almacenamiento de objetos, trasformación de objetos mediante herramientas y distribución en forma de lecciones o cursos.

- Avances tecnológicos incompatibles o poco vinculados entre sí.

- El formato de los campos varía en cada plataforma.

- Especificaciones que después de su implementación no satisfacen todas sus expectativas.

- Necesidad de adaptación de otras formas de educación a distancia a aplicaciones basadas en Web.

- Formas de implementación muy diferentes (audio, realidad virtual, video, gráficos, CD-ROM, Internet, redes internas, texto, etc.).

- Es necesario certificar cada contenido en cada plataforma individual.

- Especificaciones muy genéricas y de poco valor en casos complejos o excesivamente particulares [Martínez et al., 2001].

- La industria ha estado creciendo sin tener una idea muy clara de los componentes del e-learning y sus interacciones, por lo que la necesidad de definir una arquitectura global es crítica para la evolución de los estándares.

Como se comentó en apartados anteriores no existe un estándar global para e-learning, sino una serie de grupos que desarrollan especificaciones que la industria trata de seguir aún cuando no hayan sido todavía adoptadas formalmente como estándares. Parece que el mercado está convergiendo hacia las especificaciones de ADL-SCORM, el cual integra distintos esfuerzos realizados por organismos como AICC, IEEE e IMS. Sin embargo, SCORM no cubre todos los aspectos, como por ejemplo la interoperabilidad entre distintos sistemas de aprendizaje.

Por otra parte, los trabajos propuestos por el grupo IMS son realmente importantes en el ámbito del e-learning, principalmente por:

- Amplia visión. Cubren todos los aspectos relacionados con la teleformación.
- Dinamismo. Incorporan novedades de una manera casi inmediata sin tener que esperar a realizar tareas tan completas como puedan ser las que desarrollan los grupos que proponen estándares y no especificaciones, como el IEEE.
- Permanente actualización. A pesar de lo anterior cuando, con posterioridad, la especificación se convierte en un estándar, IMS incorpora las modificaciones de modo inmediato por estar en permanente contacto y seguimiento con los otros grupos (de hecho forman parte de ellos).
- Claridad. Las especificaciones que proponen resultan siempre claras en su concepción y estructura, en forma de documentos XML fáciles de utilizar.

De las diversas especificaciones y estándares que se han propuesto, interesan para nuestro trabajo, de manera prioritaria, las siguientes:

- La especificación de IMS sobre repositorios digitales y servicios empresariales, con especial atención a los protocolos de interoperabilidad.
- Por lo que respecta a actividades relacionadas con datos y metadatos, seguimos la propuesta de IMS, que es fundamentalmente la que recoge IEEE en su grupo de trabajo P1484.12, que apareció como estándar LOM *Learning Objects Metadata* y que ISO está en proceso de adopción como estándar ISO.
- Para terminología general nos basamos en el Glosario propuesto por IEEE a través de su grupo P1484.3. En la actualidad el CEN (Comité Europeo de Normalización) está trabajando en la adecuación de los glosarios existentes (todos en Inglés) a las diferentes lenguas nacionales que forman parte de la unión europea.

Estas normas se verán con detenimiento, especialmente el LOM, en el apartado 4.4 de la tesis donde es necesaria su utilización.

Con respecto a las propuestas arquitecturales, aunque se han incluido pocos ejemplos, puede decirse que IEEE, CISCO e IMS "*Abstract Framework*" (explicado en el apartado 2.4), son verdaderas arquitecturas y no ya implementaciones concretas de una arquitectura.

Para la definición de la arquitectura planteada en la tesis se han tenido en cuenta los trabajos del grupo de IEEE "P1484.1 *Architecture and Reference Model*" en sus especificaciones de Nivel 3, sin seguirlos de modo exhaustivo, ya que abarca elementos que no nos conciernen de modo directo. En esta especificación se habla de los componentes, y sus relaciones, que deberían figurar en todo sistema de teleformación. A sus planteamientos, criterios y propuestas hemos intentado ser fieles en esta tesis.

Dentro del modelo de arquitectura planteado por CISCO nos hemos centrado en el estudio del nivel 2 "*The Application Blueprint*", y dentro de este en el apartado de gestión de contenidos.

La arquitectura definida por IMS en su *Abstract Framework* es una referencia clara para los propósitos de esta tesis ya que está basada en una arquitectura orientada a servicios, que es precisamente el enfoque seguido en la definición de la que se propone en la tesis. Si además de eso añadimos que es el modelo de arquitectura que utiliza IMS para la definición de sus especificaciones, debemos tomarla como un modelo a aplicar.

Como aspectos comunes a todas las arquitecturas presentadas podemos destacar que todas están claramente modularizadas en cuanto a configuración y funcionalidades, y que, en general, están divididas en capas o niveles. En la arquitectura presentada también utilizaremos este estilo de representación, pero adaptado a las especificaciones de las arquitecturas orientadas a servicios cuyas características veremos en el siguiente apartado.

2.2 SERVICIOS WEB

2.2.1 Introducción a los Servicios Web

Internet introduce un nuevo entorno donde el software se puede ofrecer y acceder como *servicio*. Los servicios Web proporcionan la plataforma ideal para conseguir la completa integración de los procesos de negocio de una determinada organización con el resto de implicados en dichos procesos (clientes, proveedores, etc.).

En los últimos años la mayoría de los procesos de negocio han cambiado en flexibilidad, interconectividad y autonomía debido a las condiciones del mercado, a los nuevos modelos organizacionales y a los escenarios de uso de los sistemas de información. En este contexto, Internet y la Web están cambiando la forma en la que se

ofrecen los negocios y los servicios a la sociedad global, y en la que estos negocios interoperan. Esta tendencia nos lleva a sistemas de información conectados e integrados a través de la infraestructura que proporciona Internet.

La evolución vertiginosa de Internet y el negocio electrónico necesita de tecnologías software que faciliten la conexión de información, sistemas, dispositivos y personas. Desde la aparición de la Web, los procesos de negocio ofrecidos han ido mejorando progresivamente. En una primera fase (aproximadamente 1995) se inicia el interés por el desarrollo de *portales*, fomentando la presencia de empresas. En una segunda fase (aproximadamente 1997) se implanta el *comercio electrónico* y las transacciones a través de Internet (*eCommerce*). En una tercera fase (desde 1999 hasta la actualidad) se inicia lo que se llama la *economía digital*, con el desarrollo e implantación de los negocios (*eBusiness*), las empresas (*eEnterprises*) y los mercados electrónicos (*eMarkets*).

Los *mercados electrónicos* constituyen la nueva generación de negocios digitales proporcionando bienes y servicios de carácter electrónico que sustituyen a los tradicionales. Dichos servicios pueden integrarlos otras empresas en sus modelos de negocio: mensajería, reserva de vuelos, etc. En este contexto se esta evolucionando hacia empresas y relaciones virtuales a través de Internet, con clientes cada vez más exigentes.

Los servicios Web constituyen un esfuerzo para construir plataformas distribuidas para la Web que den un soporte adecuado, como mecanismo de implementación, a la economía digital. Los servicios Web son la última evolución tecnológica, desde el ya clásico modelo cliente-servidor, el *middleware* distribuido mediante sistemas basados en RPC (*Remote Procedure Call*), los monitores TP (*Transaction Processing*), los *brokers* de objetos, los monitores de objetos o la orientación a mensajes (MOM); pasando por el *middleware* de integración de aplicaciones (EAI), los sistemas de *Workflow* y, por último, la tecnología Web clásica. La nueva evolución tecnológica debe: *permitir la interoperabilidad Universal, la amplia y rápida adopción y un soporte eficiente de entornos abiertos*. [Pelechano, 2005]

2.2.2 Evolución de los Sistemas Distribuidos

El principal objetivo de este apartado será el de evaluar cuál es el estado del arte de la tecnología de los Servicios Web. Para poder llegar a comprender una tecnología, el primer paso debe ser siempre el de conocer sus precursores. Por qué comenzaron a surgir sistemas de ámbito distribuido, cuáles fueron las necesidades de los usuarios que permitieron su desarrollo, y cómo se han ido mejorando dichos sistemas, hasta llegar a formar los métodos con los que contamos actualmente.

Según Mahmoud [2002] y Horstmann y Cornell [2003], los sistemas distribuidos surgen para dar solución a las siguientes necesidades:

- Repartir el volumen de información.
- Compartir recursos, ya sea en forma de software o hardware.

La construcción de sistemas distribuidos presenta una solución que aumenta nuestras capacidades de comunicación, ya no estamos sujetos a las restricciones de la máquina, ahora somos capaces de utilizar los recursos de toda una red.

Los dos tipos principales de sistemas distribuidos son *los sistemas computacionales distribuidos* y *los sistemas de procesamiento paralelo*.

En programación distribuida, un conjunto de ordenadores conectados por una red es usado colectivamente para realizar tareas distribuidas. Por otro lado, en los sistemas paralelos, la solución a un problema importante es dividida en pequeñas tareas que son repartidas y ejecutadas para conseguir un alto rendimiento.

Los sistemas distribuidos se pueden implementar usando dos modelos: el modelo cliente-servidor y el modelo basado en objetos.

El modelo cliente-servidor contiene un conjunto de procesos clientes y un conjunto de procesos servidor, y además se precisan unos recursos (software). Todos estos recursos son manejados por los procesos servidor. Cliente y servidor deben “hablar” el mismo lenguaje para conseguir una comunicación efectiva: el primero solicita al segundo unos recursos, y este último los concede, le hace esperar o lo deniega según los permisos que tenga.

En el modelo Orientado a Objetos, en el que se centrará este documento, hay una serie de objetos que solicitan servicios (clientes), a los proveedores de los servicios (servidores) a través de una interfaz de encapsulación definida. Un cliente envía un mensaje a un objeto (servidor) y éste decide qué ejecutar. RMI y CORBA son algunos de los estándares de interoperabilidad utilizados por esos sistemas basados en objetos.

Por lo tanto, según lo visto hasta ahora, podemos definir un sistema distribuido como *un sistema que consiste en una colección de computadores autónomos unidos por una red, y con un sistema que les permite compartir recursos de hardware, software y datos.*

Revisando la evolución histórica, en la década de los 80 del siglo pasado, los protocolos de comunicación no eran objeto de interés por parte de los desarrolladores.

Realizar aplicaciones que dentro de una misma máquina se comunicaran entre sí, era más que suficiente.

Sin embargo, en torno a 1990 alcanzaron popularidad tecnologías como COM (*Component Object Model*) introducido por Microsoft [COM, 2006] y CORBA (*Common Object Request Broker Architecture*) introducido por OMG (*Object Management Group*) [CORBA, 2006]. En general, COM y CORBA son modelos para escribir y encapsular código binario. Estos son componentes que pueden ser fácilmente llamados desde cualquier aplicación que soporte COM o CORBA. Sin embargo, estos modelos no son fácilmente interoperables, de tal manera que objetos COM puede solamente llamar a objetos COM, y lo mismo ocurre con CORBA.

Conectar una maquina a otra se transformó en una prioridad cuando las redes locales se generalizaron. Fue entonces cuando OMG estableció IIOP (*Internet Inter-ORB Protocol*) como protocolo de comunicación para CORBA. Microsoft, mientras tanto creo DCOM (*Distributed COM*), más tarde Sun Microsystems lanzó al mercado RMI (*Remote Method Invocation*) [RMI, 2006].

RMI fue el primer *framework* que se elaboró con la idea de crear sistemas distribuidos para Java. Éste viene integrado en cualquier máquina virtual Java posterior a la versión 1.0 y está pensado para hacer fácil la creación de sistemas distribuidos a partir de una aplicación cuyas clases ya están implementadas. RMI es una forma de RPC (*Remote Procedure Call*).

La invocación de métodos remotos permite que un objeto que se ejecuta en una máquina pueda invocar métodos de un objeto que se encuentra en ejecución bajo el control de otra máquina (por supuesto, no hay problemas para las relaciones entre los objetos cuando ambos son locales). En definitiva, RMI permite crear e instanciar objetos en máquinas locales, y al mismo tiempo crearlos en otras máquinas (máquinas remotas), de forma que la comunicación se produce como si todo estuviese en local. RMI se convierte así en una alternativa muy viable a los *sockets* de bajo nivel con una serie de particularidades destacables:

- RMI permite abstraer las interfaces de comunicación a llamadas de métodos locales, sin necesidad de conocer los detalles del protocolo, y las aplicaciones distribuidas son de fácil desarrollo.

- RMI permite trabajar olvidándose del protocolo.

- RMI es flexible y extensible, destacando su recolector de basura dinámico.

Mientras RMI es una técnica de programación cuya implementación se realiza a través de unas clases o interfaces definidas, con CORBA el concepto cambia totalmente: CORBA es una especificación para crear y usar objetos distribuidos, no un lenguaje de programación. Desarrollar aplicaciones distribuidas con CORBA es similar a hacerlo con RMI, porque una interfaz debe ser definida primero, pero las interfaces de RMI son definidas en Java mientras que las de CORBA emplean IDL. Pero, sin embargo, los objetos CORBA son diferentes porque pueden ejecutarse en cualquier plataforma y se pueden situar en cualquier punto de la red.

El uso de CORBA es más complejo de lo que hemos visto hasta ahora, pero también bastante más completo. Otra de las características y al mismo tiempo diferencias con RMI, es que CORBA fue desarrollado con un lenguaje independiente, donde los objetos se mueven en un sistema heterogéneo. Por su parte, RMI fue desarrollado para un lenguaje simple como Java, donde los objetos se desenvuelven en un entorno con un lenguaje homogéneo. Además, CORBA soporta parámetros de entrada y salida pero RMI no, lo cual es una ventaja clara de CORBA sobre RMI; sin embargo debemos tener en cuenta que los objetos CORBA no tienen recolector de basura y, por lo tanto, será obligación del programador preocuparse de ello, al contrario que con RMI, tarea que se realiza dinámicamente.

Con estos protocolos se pueden llamar a componentes que se encuentren en otros computadores a través de la red. Estas llamadas se realizan bajo la forma de RPC (*Remote Procedure Call*). Es necesario aclarar que estos protocolos no son interoperables.

RPC permite a los desarrolladores de software hacer funciones o llamadas a métodos a través de la red, lo cual permite rudimentarias aplicaciones distribuidas sobre la red. Es un mecanismo a nivel de capa de aplicación, que comunica una aplicación con otra. La llamada de RPC es idéntica a la sintaxis de una función local. La función toma la forma de interacción cliente-servidor. RPC actúa con el mismo objetivo que Java RMI, pero se diferencia en que:

- No hay contexto de objeto. Métodos remotos son llamados a través de la red como una API remota. RMI también logra esto de forma natural.

- El cliente y el servidor pueden ser implementados con independencia de lenguajes de programación.

La solución a estos problemas pasa por disponer de una arquitectura que solucione el problema de la interoperabilidad entre las diferentes tecnologías y plataformas software utilizadas, tanto en la parte cliente desde donde se invocan los servicios, como en los servidores en los que se ubican.

Es necesario desarrollar un sistema de comunicación “aplicación a aplicación” que permita la total comunicación desde cualquier máquina a cualquier otra sin importar el sistema operativo, entorno de lenguajes o modelos de objetos distribuidos con el que esté desarrollado. Por supuesto, deberá estar basado en una serie de estándares, para asegurar una completa interoperabilidad entre los distintos sistemas desarrollados, y que de esta manera permitan a los desarrolladores implementar aplicaciones distribuidas, construidas a partir de combinaciones de módulos software, que serán llamadas desde distintos sistemas distribuidos en regiones geográficas distintas. [WebLogicPro, 2004]

2.2.3 Los Servicios Web son la solución

Si intentamos centrar el estado actual del desarrollo de aplicaciones basadas en la Web, podemos encontrar una gran cantidad de tecnologías, muchas de ellas incompatibles entre sí, y lo que es peor, inaccesibles a través de Internet. Sin embargo, las arquitecturas basadas en tecnología de componentes, están tomando un papel principal dentro del desarrollo Web.

Los servicios Web, se proponen como una alternativa para facilitar la intercomunicación entre diferentes arquitecturas de componentes, ofreciendo una visión de dichas arquitecturas, basada en servicios, totalmente compatible con Internet.

La aparición de los servicios Web, y de las Arquitecturas Orientadas a Servicios (SOA) [SOA, 2006], supone el establecimiento de nuevos mecanismos de comunicación B2B (*Business to Bussiness*), B2C (*Business to Consumer*), B2E (*Business to Employee*). El organismo encargado de definir estos estándares, y asegurar este comportamiento, es el WS-I (Web *Services Interoperability Organization*, [WS-I, 2005]). Gracias a este organismo, va a ser posible que sistemas desarrollados en diferentes plataformas y diferentes lenguajes de programación, puedan interactuar.

La Arquitectura Orientada a Servicios (SOA), que será tratada detalladamente en un apartado posterior, pretende extender la idea de servicio Web, de forma que una invocación, de manera totalmente transparente al usuario, implique la ejecución de más de un servicio Web.

Algunas definiciones de servicio Web:

- Una aplicación modular que se autodescribe, que puede ser publicada, localizada, invocada o usada desde cualquier parte de la Web y que está basada en estándares abiertos como XML, UDDI, SOAP o WSDL. [Newcomer, 2002].

- Una aplicación accesible a otras aplicaciones a través de la Web. [Pelechano, 2005].

- Un sistema software identificado por una URI (*Uniform Resource Identifier*), cuyos interfaces públicos y enlaces se definen y describen utilizando XML. Su definición puede ser descubierta por otros sistemas software. Estos sistemas pueden interactuar con el servicio Web de la forma prescrita por su definición, usando mensajes basados en XML a través de protocolos y estándares de Internet [W3C, 2004a] [W3C, 2004b].

- Una interfaz que describe un conjunto de operaciones, accesibles a través de la red mediante mensajería XML estándar. [IBM, 2006]

- Cualquier aplicación accesible por HTTP/HTTPS, con la que se puede interactuar usando mensajes SOAP, se registra en un registro UDDI y tiene una descripción WSDL [HP, 2006].

- Un proveedor de información o capacidades expuestas en una red a través de interfaces, protocolos consistentes y estándares [MS, 2006].

El modelo de servicios Web potencia el desarrollo de aplicaciones distribuidas. Un ejemplo podría ser el caso de una agencia de viajes que tiene asociado su sistema de reservas *on-line* con el sistema de aerolíneas, reserva de hoteles y alquiler de coches; de tal manera que un viajero puede al mismo tiempo hacer una reserva de un vuelo, una habitación de hotel y un coche de alquiler, todo esto a través servicios Web intercomunicados entre sí para conseguir dicha finalidad, constituyendo lo que se conoce como arquitectura orientada a servicios.

La arquitectura de los servicios Web permite que ciertos servicios sean dinámicamente descritos, publicados, descubiertos e invocados en un ambiente de computación distribuida, haciendo uso de los estándares WSDL, UDDI, SOAP y XML respectivamente.

Según Pelechano [2005], los servicios Web, para tener éxito y promover su rápida implantación a nivel mundial, deben cumplir los siguientes requisitos:

- Estar basados en estándares.

- Requerir una cantidad mínima de infraestructura (conjunto mínimo de estándares).

- Permitir la integración de aplicaciones de manera flexible.

- Estar centrados en mensajes y documentos y no en API's.

2.2.4 Descripción general de los Servicios Web

Según nos comenta Newcomer [2002] y Deitel et al. [2002], los Servicios Web constituyen el siguiente paso en la evolución de la tecnología orientada a objetos, y representan una revolución al alejarse de las arquitecturas tradicionales tipo cliente-servidor para obtener nuevas arquitecturas distribuidas tipo igual-a-igual (*Peer to Peer*, P2P).

Como ya se ha indicado, estos servicios se basan en un conjunto de estándares (WSDL, UDDI, XML y SOAP, que serán comentados más adelante) que permiten a los desarrolladores implementar aplicaciones distribuidas, utilizando herramientas muy distintas para crear aplicaciones que utilizan una combinación de módulos de software que son llamados desde diversos sistemas distribuidos en regiones geográficas distintas.

Los servicios Web son aplicaciones auto-contenidas y modulares que pueden ser:

- Descritas mediante un lenguaje de descripción de servicio, como el lenguaje WSDL (*Web Service Description Language*).

- Publicadas, al incluir las descripciones y políticas de uso en algún registro conocido, utilizando el método de registro UDDI (*Universal Description, Discovery and Integration*) [UDDI, 2005].

- Encontradas, también utilizando el estándar UDDI, al enviar peticiones al registro y recibir los detalles necesarios para la localización y enlace (*binding*) sobre aquel servicio que se ajusta a los parámetros de la búsqueda.

- Asociadas, al utilizar la información contenida en la descripción del servicio para crear una instancia de servicio disponible (o *Proxy*).

- Invocadas sobre la red, al utilizar la información contenida en los detalles de enlace de la descripción del servicio; en un documento WSDL. Durante la invocación, como veremos más adelante haremos uso del protocolo SOAP.

- Compuestas con otros servicios para integrar servicios y aplicaciones nuevas, en lo que constituirá la base de SOA (*Service-Oriented Architecture*), que será explicada en detalle en un apartado posterior.

Todos los estándares comentados se basan en XML: los documentos WSDL, son documentos XML; el protocolo SOAP, que permite la comunicación entre servicios, internamente trata información XML en sus dos variantes, por un lado la mensajería SOAP, tratando información XML explícita y por otro lado RPC, que la trata, pero de una manera implícita.

En cuanto a los componentes de una arquitectura orientada a servicios Web, podemos hablar de (figura 2.14):

- Servicio: La aplicación es publicada para que sea utilizada por todos aquellos solicitantes que cumplan los requisitos especificados por el proveedor de servicios. Evidentemente la implementación se realizará sobre una plataforma accesible en la red. El servicio se describe a través del lenguaje de descripción de servicios, WSDL. Tanto la descripción como las políticas de uso han sido publicadas anteriormente en un registro.

- Proveedor de Servicio: Desde el punto de vista comercial, es quien presta el servicio. Desde el punto de vista de la arquitectura, es la plataforma que provee el servicio.

- Registro de Servicios: Es un repositorio de descripciones, donde los proveedores publican sus servicios y la forma de accederlos. Permitirá además a los solicitantes realizar distintos tipos de búsquedas, obteniendo de éstas, los detalles necesarios para poder localizarlos y utilizarlos.

- Solicitante de servicios: Desde el punto de vista comercial, la empresa que requiere cierto servicio. Desde el punto de vista de la arquitectura, la aplicación cliente que busca e invoca un servicio.

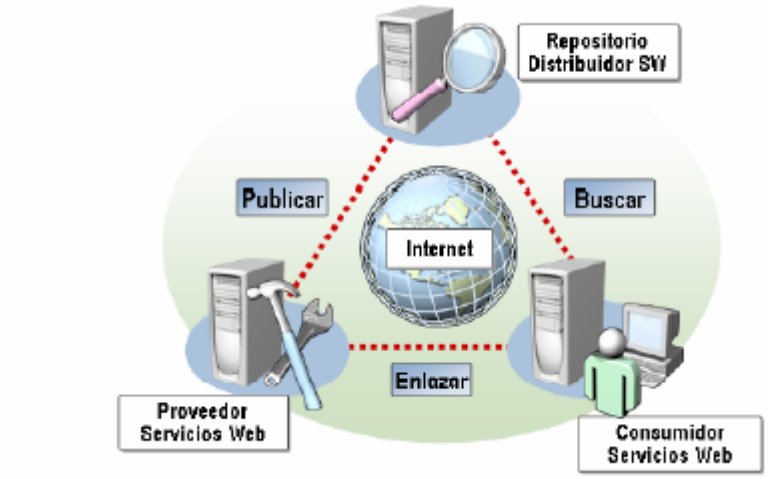

Figura 2.14. Relaciones entre los componentes que conforman una arquitectura orientada a servicios Web

Las operaciones que se pueden realizar con los servicios Web, son las siguientes:

- **Publicar/Cancelar:** Un proveedor de servicios podría publicar un determinado servicio comercial (*e-business*) a uno o más registros de servicios, además de, evidentemente, cancelar dicha publicación en un momento dado.

- **Búscar:** Los solicitantes de servicios (clientes) podrán interactuar con uno o más registros para descubrir un conjunto de servicios comerciales con los que puedan interactuar para encontrar una solución a sus problemas.

- **Enlazar (*Bind*):** Los solicitantes de servicios negocian con los proveedores la forma de acceder e invocar a sus servicios comerciales (*e-business*).

Según hemos comentado, un servicio Web es una colección de funciones que son presentadas como una sola entidad y que es anunciada en la red para ser usada por otros programas. Son por lo tanto, bloques de construcción para crear sistemas distribuidos abiertos.

Actualmente, las aplicaciones que se están realizando, poseen la capacidad de buscar y seleccionar servicios dinámicamente en tiempo real, basándose en parámetros como el costo, la calidad, o la disponibilidad. Esto supone una gran ventaja a la hora de utilizar sistemas basados en servicios Web, ya que el sistema, de forma automática, elegirá el servicio que más se ajuste a sus necesidades, y por lo tanto el rendimiento del sistema se verá incrementado.

Entre las razones por las cuales los servicios Web jugarán un rol principal en la siguiente generación de sistemas distribuidos, están las siguientes:

- **Interoperabilidad:** Cualquier servicio Web puede interactuar con cualquier otro servicio. El protocolo estándar SOAP permite que cualquier servicio pueda ser ofrecido o utilizado independientemente del lenguaje o ambiente en que se haya desarrollado.

- **Omnipresencia:** Los servicios Web se comunican utilizando HTTP y XML. Cualquier dispositivo que trabaje con éstas tecnologías puede ser tanto cliente y acceder a los servicios Web. Por ejemplo, una máquina de venta de refrescos, incluso podría comunicarse vía inalámbrica con el servicio Web de un proveedor local y ordenar un pedido de suministro.

- **Barrera mínima de participación:** Los conceptos que hay detrás de los servicios de Web son fáciles de comprender y se ofrecen infinidad de herramientas de desarrollo (*ToolKits*); como por ejemplo las ofrecidas por IBM, Sun Microsystems, Apache o Systinet. Todas ellas permiten a los desarrolladores crear e implementar rápidamente servicios Web.

- **Apoyo de las Industrias:** Las principales compañías apoyan el protocolo SOAP y la tecnología derivada de los servicios Web.

2.2.5 Diferentes retos que tienen que afrontar los servicios Web

Para que los servicios Web tengan éxito, se requiere vencer algunos retos técnicos, entre los cuales podemos destacar:

- **Descubrimiento:** ¿Cómo se anuncia un servicio Web para ser descubierto por otros servicios? ¿Qué sucede si el servicio es modificado o se cambia una vez que ha sido anunciado? Existen dos estándares que están diseñados para ello, por un lado tenemos WSDL y por otro UDDI. ¿Pero que pasa cuando un servicio está ya registrado, y cambia su localización (dirección IP por ejemplo)? ¿Es necesario modificar el registro? La contestación a esta pregunta es afirmativa. Uno de los mayores problemas de esta arquitectura es la necesidad de regenerar el documento WSDL ante cambios en las localizaciones de los servicios. Sin embargo, existen en el mercado innumerables herramientas que generan este documento de forma automática, pero no se puede obviar que este es uno de los puntos débiles de esta tecnología.

- **Confiabilidad:** Evidentemente, algunos sistemas de servicios Web serán más confiables que otros. ¿Cómo se puede medir ésta confiabilidad y ser comunicada? ¿Qué sucede cuando un servidor de servicios temporalmente queda inutilizable? ¿Cómo se localiza o utiliza un servicio alternativo de otra compañía?, o ¿ponemos en espera a los clientes hasta que se vuelva a habilitar? ¿Cómo se sabe en que otra compañía se puede confiar? Actualmente ya existen en el mercado algunas herramientas específicamente diseñadas para medir la calidad de los servicios Web, pero sigue siendo necesaria una estandarización sobre este tema. Los resultados sobre la calidad de diferentes servicios Web servirán como parámetro de comparación y ayudarán al consumidor a decantarse por un servicio u otro. En la figura 2.15 se muestra la pila de estándares que ofrece los mecanismos de confiabilidad y calidad de servicios (QoS: *Quality of Service*):

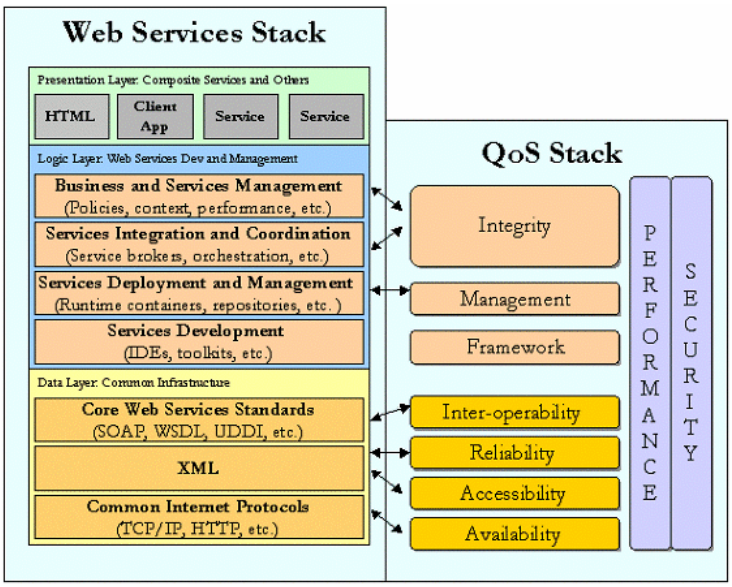

Figura 2.15. La pila de tecnologías que garantizan la QoS

- **Seguridad:** Algunos servicios Web están públicamente disponibles y con poca seguridad, pero la mayoría de los servicios comerciales utilizan comunicaciones encriptadas con autentificación. Es probable que el protocolo HTTP sobre SSL provea la seguridad básica, pero los servicios individuales requerirán de un mayor nivel de especificidad. ¿Cómo autentifica un servicio Web a sus usuarios? ¿Cómo distingue un servicio los niveles de privilegios entre los diversos usuarios? Hoy en día, este es uno de los puntos fuertes de esta tecnología. Recientemente han surgido protocolos para garantizar la seguridad en las comunicaciones a través de servicios Web. Este punto será tratado con profundidad en el apartado sobre la arquitectura SOA.

- **Transacciones:** Los sistemas tradicionales de transacciones utilizan un método de compromiso dividido en dos fases: primero se recuperan todos los recursos de los participantes y se aseguran estos hasta que se lleva a cabo la transacción completa; terminada la transacción, se liberan los recursos. Este método puede funcionar bien en ambientes cerrados donde las transacciones son de corta duración, pero no en ambientes abiertos donde las transacciones pueden llevar horas, o incluso días.

- **Administración:** ¿Qué tipos de mecanismos se requieren para administrar un sistema altamente distribuido? ¿Es posible delegar la administración de algunos servicios Web a otros? Si, como se ha comentado anteriormente, la tendencia de los desarrollos orientados en servicios es que éstos puedan intercomunicarse con otros y completar así la funcionalidad requerida, evidentemente, esto de forma totalmente transparente al usuario, por lo tanto sería válido este tipo de delegaciones de administración.

- **Contabilidad:** ¿Cómo se define qué tiempo puede un usuario acceder y ejecutar un servicio Web? ¿Cómo se pueden cobrar los servicios Web? ¿Cómo se podría comercializar el servicio, bajo suscripción o pago por evento? Para la comercialización y el cobro de los servicios Web existen una serie de tecnologías asociadas como son: "*Pricing Port Type*" (precio por servicio), "*Payment Provider Service*" (proveedor del servicio de pago) y WS-Agrenment (especificación para la política de contrato).

- **Pruebas:** ¿Cómo se puede depurar un servicio Web que proviene de diferentes compañías, que es hospedado en diferentes ambientes y en diversos sistemas operativos? Esto presenta un problema difícil de superar, pero de hecho existen sistemas que han resuelto este tipo de problemática. Dos ejemplos de ello son, la sociedad humana y los organismos biológicos. Ambos sistemas poseen las siguientes propiedades:

  - ✓ Tolerantes a fallos.
  - ✓ Paralelismo masivo.
  - ✓ Distribuidos.
  - ✓ Bien organizados.
  - ✓ Auto-reparables
  - ✓ Diseñado en capas.
  - ✓ Diseñado a partir de componentes simples.

Tomando estos ejemplos, es posible crear una "sociedad de servicios Web" donde los componentes colaboran unos con otros para lograr sus metas individuales.

2.2.6 Estándares y tecnologías subyacentes que conforman los servicios Web

La infraestructura mínima que requieren los servicios Web se puede definir en términos de Pelechano [2005]:

- Lo que va en "la red": Formatos y protocolos de comunicación.

- Lo que describe lo que va en la red: Lenguajes de Descripción de Servicios.

- Lo que nos permite encontrar y almacenar dichas descripciones: Descubrimiento de Servicios.

Los servicios Web basan todo su funcionamiento y características en los siguientes estándares y protocolos:

- **XML** (*eXtensible Markup Language*) [W3C, 1998], publicó en 1998 y ha revolucionado la forma en que se estructura, describe e intercambia información. Independientemente de las múltiples formas en que se utiliza hoy en día XML, todas las tecnologías de servicios Web se basan en XML. Se trata del estándar central de esta arquitectura, sobre el cual se apoyan el resto. El diseño de XML deriva de dos fuentes principales: SGML (*Standard Generalized Markup Language*) [ISO, 1986] y de HTML (*HyperText Markup Language*) [ISO, 2000].

- **UDDI** (*Universal Description, Discovery and Integration*) [UDDI, 2006], es un directorio que contiene un registro/repositorio de descripciones de servicios Web. Este estándar permite a las empresas registrarse en un tipo de directorio de Internet que les ayuda a anunciar sus servicios, de tal forma que el resto de compañías puedan localizar sus servicios y realizar transacciones en la Web. El proceso de registro y consultas se realiza utilizando mecanismos basados en XML y HTTP(S). Por lo tanto, la especificación UDDI tiene dos objetivos esenciales, en primer lugar servir de soporte a los desarrolladores para encontrar información sobre servicios Web y poder construir clientes; y por otro lado, facilitar el enlace dinámico de servicios Web, permitiendo consultar referencias y acceder a servicios de interés.

- **SOAP** (*Simple Object Access Protocol*) [W3C, 2003], es un estándar del World Wide Web Consortium que define un protocolo que da soporte a la interacción (datos + funcionalidad) entre aplicaciones en entornos distribuidos y heterogéneos, es interoperable (neutral a la plataforma, lenguajes de programación, independiente del hardware y protocolos). Funciona sobre la infraestructura (estándares) existentes en Internet. SOAP define cómo organizar la información XML de una manera estructurada y tipada para intercambiarla entre los distintos sistemas. El protocolo SOAP simplifica el acceso a los objetos, permitiendo a las aplicaciones invocar métodos de objetos o funciones, que residen en sistemas remotos.

- **WSDL** (*Web Service Description Language*) [W3C, 2001a], creado originalmente por IBM, Microsoft y Ariba. Tiene un rol y un propósito similar al de los IDL (*Interface Definition Language*) de las plataformas *middleware*. Un archivo WSDL es un documento XML que describe los servicios Web, en particular sus interfaces. Como característica que lo diferencia de los IDL, es que WSDL debe definir los mecanismos de acceso (protocolos) a los servicios Web. Otra característica diferenciadora es la necesidad de definir (en la especificación) la localización del servicio (puntos finales). La separación de interfaces y enlaces de protocolos, y la necesidad

de incluir información de localización permite la definición de especificaciones modulares. WSDL permite definir interfaces más complejas y expresivas; permitiendo definiciones de interacciones asíncronas y diferentes paradigmas de interacción, y la posibilidad de combinar o agrupar operaciones.

En la figura 2.16, se muestra un diagrama con la arquitectura de los sistemas basados en servicios Web, en el que se representan los estándares y tecnologías descritas.

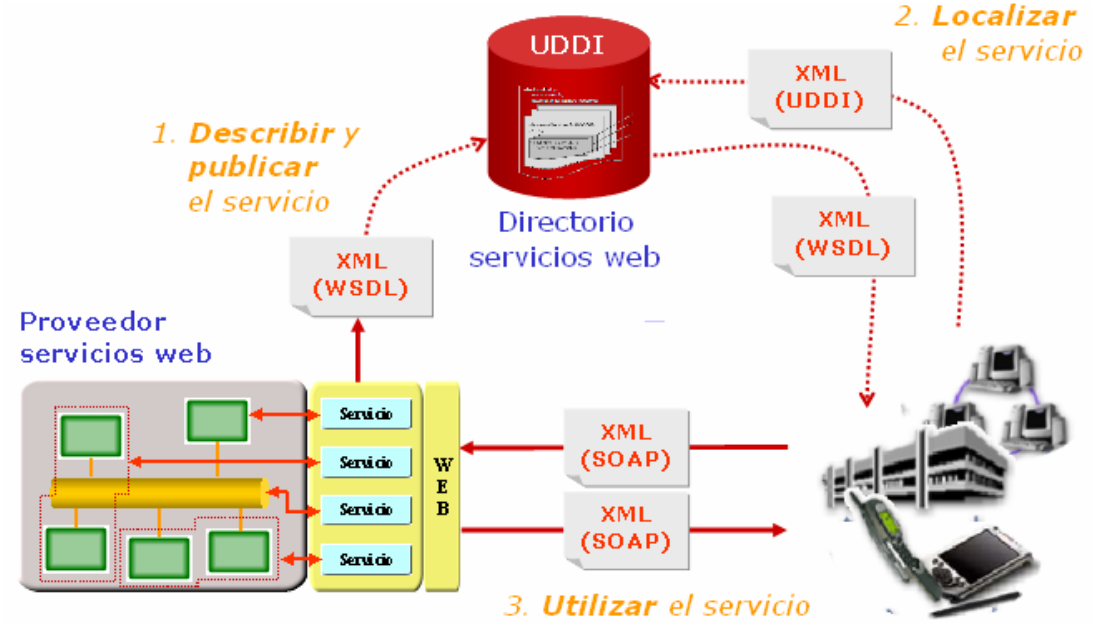

Figura 2.16. El modelo integral de Servicios Web

Una vez comentado a grandes rasgos las características de la arquitectura, pasaremos a comentar más detalladamente cada uno de estos estándares.

XML (eXtensible Markup Language)

La primera versión de XML (1.0) se publicó en 1998. Esta versión ha pasado por sucesivas transformaciones, hasta originar la actual versión 1.1.

El estándar XML, como ya se comentó anteriormente, está basado en el estándar SGML (*Standard Generalized Markup Language*), que empezó a elaborarse desde principios de los años 70, y a su vez basado en el GML creado por IBM en 1969. Esto quiere decir que el concepto en el que se basa, existe desde hace bastante tiempo.

XML está además asociado a la recomendación DOM (*Document Object Model*) [W3C, 2005], aprobado también en 1998. DOM no es más que un modelo de objetos

(en forma de API) que permite acceder a las diferentes partes que pueden componer un documento XML o HTML. Desde su aparición, muchas personas se han decantado por su estudio y desarrollo, como otros muchos temas relacionados con Internet; aunque también tiene sus detractores.

Desde comienzos de su publicación, XML ha sufrido un gran auge. Esto significa que ha pasado de mera especulación a ser una realidad empresarial: los programas que lo soportan han crecido del mismo modo exponencial, y a día de hoy, no hay empresa de software que se precie que no anuncia la compatibilidad de sus productos más vendidos con este nuevo estándar: Microsoft (Office 2000, Office XP, SQL Server), Oracle (Oracle 10g, Web Application Server) o IBM-Lotus (Notes) son tres claros ejemplos de ello. Aún más increíble es pensar que hay empresas que se han creado entorno a él, u otras que han movido su actividad hacia su ámbito (de SGML a XML, por ejemplo, como ArborText); o Software AG cuyo eslogan es "*The XML Company*".

XML es un metalenguaje, es un lenguaje para definir lenguajes de estructuración de texto basados en marcas explícitas o etiquetas, que son entendibles por una persona y que pueden ser interpretadas por un computador. Los elementos de estos lenguajes dan información sobre lo que contienen, pero no obligatoriamente sobre la estructura física y presentación como ocurre en HTML. La idea que subyace bajo el XML es la de crear un lenguaje muy general que sirva para todo tipo de ámbitos en los que se desenvuelva.

HTML está diseñado para presentar la información directamente a los humanos, y esto sin duda es algo bueno, pero es un lenguaje complicado de procesar para los programas informáticos. HTML no es un lenguaje idóneo, no indica la información que está representando, se preocupa principalmente del formato en el que tiene que ir el contenido, pero no dejar ver que lo que está mostrando es el título de un libro o el precio de un artículo. XML hace precisamente esto: describe el contenido de lo que etiqueta, de cara a un sistema informático o a un usuario, es esa la verdadera información que desea obtener de un documento, y no el formato con el que aparecerá reflejado durante su ejecución.

Básicamente XML no ha nacido sólo para su aplicación en Internet, sino que se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, etc. Sin ir más lejos, algunos lenguajes definidos en XML, se utilizan en campos como la química, la física, las matemáticas, el dibujo, tratamiento del habla, y otras muchas.

De todos modos, HTML continuará jugando un importante papel, porque los componentes de presentación y enlace de XML no han madurado tan rápidamente como el propio XML, y todavía deben eliminarse muchas complejidades. Estos componentes

vienen especificados en XSLT y XLink: el primero es el componente de XML que especifica la forma en que aparecen los documentos en pantalla (*Extensible StyleSheet Language*, o XSL) que apareció a finales de 1999 como una recomendación de W3C [W3C, 1999a]. El segundo componente es el que permite especificar enlaces entre distintos recursos (*Extensible Linking Language*, o XLink) [W3C, 2001b].

Estos estándares ofrecen mejoras sustanciales sobre los tradicionales hiperenlaces y presentación de datos que utilizan HTML y hojas de estilo en cascada (*cascading style sheets*, CSS) [W3C, 1996]. Pero en un contexto empresarial, XML ya es útil. Puede reemplazar los procesos de producción, trabajando para transferir y organizar los datos en el servidor a nivel de *middleware*.

HTML no está estructurado, mientras que los documentos XML lo están obligatoriamente. Por ejemplo, cada documento XML requiere de la existencia de un elemento raíz, y los siguientes conjuntos de etiquetas deberán ser agrupados convenientemente, lo que da como resultado una jerarquía de datos. Los documentos XML, a diferencia de HTML, pueden ser validados por archivos externos XSD (XML Schema) [W3C, 2004c]. Esto, asegura que cada conjunto de etiquetas está apropiadamente definido y validado.

En resumen, XML difiere de HTML en tres grandes aspectos:

- Los proveedores de información pueden definir nuevos nombres de etiquetas y atributos, a su voluntad.

- Pueden anidarse las estructuras del documento a cualquier nivel de complejidad.

- Cualquier documento XML puede contener una descripción opcional de su gramática para el uso de aplicaciones, que necesitan realizar una validación estructural.

Una vez se han comentado las principales características de XML, se puede hablar de sus principales ventajas, las cuales citamos a continuación:

- Se consigue mayor precisión en las búsquedas, ya que cuando se utilizan metadatos, la efectividad de los motores de búsqueda se incrementa notablemente.

- Se pueden reemplazar, en muchos casos, los documentos en papel que se intercambian las empresas, como facturas, contratos o pólizas de seguros.

- Facilitar las tareas de clasificación y procesamiento de información, como la identificación de un usuario que realiza operaciones de comercio electrónico, los documentos, etc., que pueden unificarse en un solo estándar. Las páginas generadas con este nuevo lenguaje son ideales como plataforma de transferencia de datos.

- Permitir que los datos se encuentren a nivel local para ser utilizados en cálculos. En primer lugar, la información es leída por un navegador, entonces los datos pueden ser manipulados mediante lenguajes de *script* u otros que soporten el modelo de objetos XML.

- Proporcionar al usuario una visión estructurada de la información. Los datos que lleguen a la máquina del cliente pueden visualizarse de muy diversas formas siendo posible su personalización para determinados grupos de usuarios, de manera dinámica, mediante el establecimiento de preferencias y parámetros de confirmación.

- Es posible integrar datos procedentes de fuentes diversas. Normalmente, dada la naturaleza de la red, en las aplicaciones Web suele darse la necesidad de tener que tratar con datos que se originaron en varias fuentes. XML facilita las tareas de compartición, procesamiento y distribución de los mismos, ya que una de las cualidades que caracterizan este estándar es que es extensible y puede utilizarse para describir los datos de una gran cantidad de aplicaciones.

- Se mejora el rendimiento a través de actualizaciones graduales de la información. Los desarrolladores no tienen que enviar la estructura de los datos cada vez que se produce un cambio.

Existen varias especificaciones del W3C que dan soporte al estándar XML; las principales son las siguientes:

- **DTD** (*Document Type Definition*): Definición del tipo de documento. Es, en general, un archivo o archivos que encierra una definición formal de un tipo de documento y, a la vez, especifica la estructura lógica de cada documento. Define tanto los elementos de una página como sus atributos. El DTD del XML es opcional.

- **XML Schema** [W3C, 2004c]: Al igual que los DTDs, los *Schemas* describen la estructura de la información. El motivo de la creación de este nuevo estándar para realizar la labor de los DTDs es, básicamente, la utilidad. Durante un tiempo, y a falta de otra solución más ajustada, se emplearon los mecanismos que proporcionaba SGML para modelizar la información en

XML; pero el descubrimiento de nuevas aplicaciones de XML, al margen de la estructuración de documentos, forzó la creación de otras soluciones que ayudasen a solventar los nuevos problemas a los que se enfrentaba el mercado.

- **XSL** (*eXtensible Stylesheet Language*) [W3C, 1999b]: Define o implementa el lenguaje de estilo de los documentos escritos para XML. Permite modificar el aspecto de un documento. Se pueden lograr múltiples columnas, texto girado, orden de visualización de los datos de una tabla, múltiples tipos de letra con amplia variedad en los tamaños. Este estándar está basado en el lenguaje de semántica y especificación de estilo de documento (DSSSL, *Document Style Semantics and Specification Language*) [ISO, 1996] y, por otro lado, se considera más potente que las hojas de estilo en cascada (CSS, *Cascading Style Sheets*), usada en un principio con el lenguaje DHTML. Se espera que el CSS sea usado para visualizar estructuras simples de documentos XML, ofreciendo nuevas formas de composición y una más rápida visualización. Por otra parte, XSL podrá ser utilizado donde se requiera más potencia de diseño, como documentos XML que encierran datos estructurados (tablas, organigramas, etc.). XSL es, en realidad, dos estándares muy diferentes:

    - XSLT (XSL Transformations): Es un lenguaje que describe cómo debe transformarse un archivo XML en otro archivo XML. Por ejemplo, un archivo que contenga etiquetas que describen una receta, puede convertirse mediante este sistema, a otro archivo que contiene etiquetas que solamente describen la posición del texto en una página impresa. Obviamente, en semejante transformación hay información que se pierde. La idea es que aplicando diferentes archivos XSLT se puedan obtener archivos XML diferentes, adecuados a diferentes presentaciones o usos de la información.

    - XSL FO (XSL Formatting Objects): Un archivo XSLFO es una especie de HTML, lleno de etiquetas de colores, tamaños y posiciones. Es un archivo que no preserva nada la semántica de la información original, solamente describe cómo debe mostrarse en pantalla, o en papel. Es similar en concepto al lenguaje PostScript, o quizás al TeX.

- **XLL** (*eXtensible Linking Language*): Define el modo de enlace entre diferentes enlaces. Se considera que es un subconjunto de HyTime (*Hipermedia/Timed-based structuring Language*) [ISO, 1992] y sigue algunas especificaciones del TEI (*Text Encoding Initiative*) [TEI, 1994]. Este

lenguaje de enlaces extensible, tiene dos importantes componentes: XLink y XPointer. Va más allá de los enlaces simples que sólo soporta el HTML. Se puede implementar con enlaces extendidos.

- **XUA** (XML User Agent): Estandarización de navegadores XML. Todavía está en proceso de creación de borradores de trabajo. Se aplicará a los navegadores para que compartan todas las especificaciones XML.

SOAP (Simple Object Access Protocol)

SOAP (en su versión actual 1.2, recomendado por W3C en 2003 [W3C, 2003]) define un protocolo que da soporte a la interacción (datos + funcionalidad) entre aplicaciones en entornos distribuidos y heterogéneos, y es interoperable, es decir, neutral a la plataforma, lenguajes de programación, independiente del hardware y protocolos. Funciona sobre la infraestructura (estándares) existentes en Internet. SOAP define cómo organizar la información XML de una manera estructurada y tipada para intercambiarla entre los distintos sistemas.

Según Pelechado [2005], SOAP especifica:

- Un **formato de mensaje** para una comunidad unidireccional, describiendo cómo se empaqueta la información en documentos XML.

- Un conjunto de convenciones para usar mensajes SOAP, para implementar el patrón de interacción RPC, definiendo **cómo** los clientes pueden **invocar** un procedimiento remoto enviando un mensaje SOAP y cómo los servicios pueden **responder** enviando otro mensaje al llamador.

- Un conjunto de **reglas** que una entidad que procesa mensajes SOAP debe seguir, definiendo en particular los elementos XML que una entidad debe leer y entender, así como las acciones que deben tomar si no entienden el contenido (reglas de codificación de datos).

- Una descripción de cómo se debe transportar un mensaje SOAP sobre HTTP y SNMP. Se definirán *bindings* a otros protocolos de transporte en futuras versiones de la especificación.

Los mensajes SOAP son fundamentalmente mensajes en una dirección, desde un "emisor" (cliente) hacia un "receptor" (servidor), y las comunicaciones son del tipo *request/response*. Todos los mensajes son documentos XML con su propio esquema, que incluye sus propios espacios de nombres (*namespaces*), elementos y atributos.

SOAP define dos namespaces: *envelope* y *encoding*. Como característica destacable, podemos decir que no puede tener DTD asociados. Los mensajes SOAP constan de tres secciones: *envelope, header* y *body*.

- **Envelope:** elemento raíz del mensaje.
- **Header:** información de identificación del contenido.
- **Body:** es el contenido del mensaje.

Se pueden realizar extensiones al protocolo; esto es posible gracias a la adicción de módulos de funcionalidad. Este enfoque permite a los desarrolladores usar los módulos y funcionalidades que necesiten, sin la necesidad de tener que implementar la totalidad de estos. En pocas palabras, el protocolo puede ser extensible. Algunas de las extensiones más importantes que se pueden realizar son las que se muestran a continuación:

- **Attachments:** Hace referencia a la posibilidad de incluir un documento no XML, o archivo binario, enviar documentos de fax, imágenes de medicina, dibujos de ingeniería, o cualquier otro tipo de imágenes, codificadas en Base64.
- **Routing/Intermediaries:** Relacionadas con el proceso de encaminar mensajes SOAP a través de intermediarios. Este mecanismo ofrece la posibilidad de agregar varios servicios Web y ofrecerlos como parte de un paquete. Esto es una tarea que permite hacer los servicios Web escalables a través del direccionamiento, incluso, hacia múltiples servidores.
- **Reliable Messaging:** Determina cuanto tiempo espera un servidor cuando envía un mensaje y espera por la respuesta.
- **Security:** Esta extensión nos permitirá dar un marco de seguridad a la comunicación. Algunos de los aspectos podrían ser aplicar SSL, firma digital, etc. Mediante XML Signature [W3C, 2002] podemos proporcionar integridad, autenticación de mensajes, y/o servicios de autenticación de firmas.
- **Quality of Service (QoS):** Es una medida para calificar la calidad del servicio, determinando la usabilidad y utilidad del servicio.
- **Context/Privacy:** Referencia a los mecanismos para guardar el contexto y la privacidad del entorno de los usuarios [W3C, 2004d].

- **Transaction Support:** Permite que un grupo de operaciones o acciones se comporten como si fueran una simple unidad (o todo falla o todo es un éxito).

Según lo expuesto, la estructura de un mensaje SOAP, es la que se muestra en la figura 2.17:

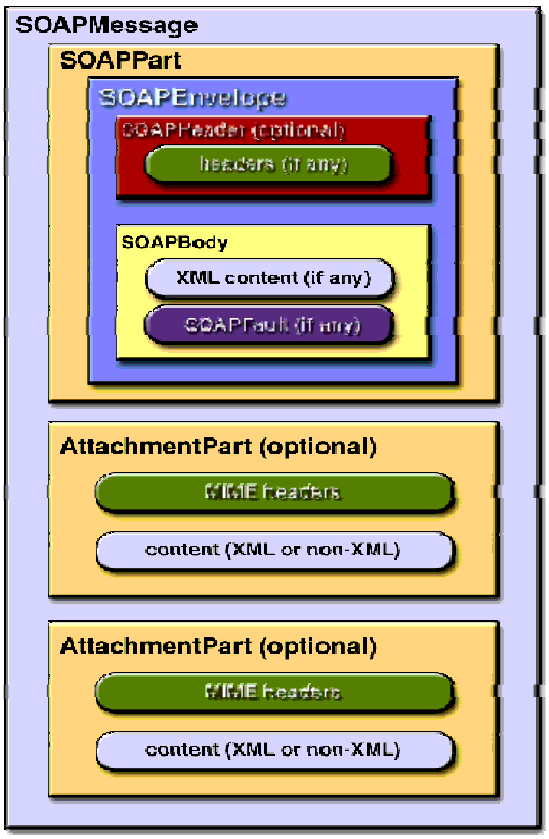

Figura 2.17. Estructura de un mensaje SOAP

En cuanto al rendimiento de los mensajes de SOAP utilizando HTTP y XML, comparado con el rendimiento de mensajes binarios utilizados por CORBA, DCOM y RMI, se puede afirmar que en el caso de los últimos, tanto el destinatario y remitente tienen conocimiento completo del contenido del mensaje y no codifican metainformación como los nombres o tipo de argumentos. Quizás este método sea eficiente, pero dificulta a los intermediarios el procesamiento de mensajes. Y debido a que cada sistema utiliza una codificación binaria distinta, dificulta la construcción de sistemas interoperables.

Dado que SOAP utiliza XML para codificar mensajes, es relativamente sencillo procesar los mensajes en cada paso del proceso de llamada. Además, la facilidad de

depuración de mensajes SOAP permite la convergencia rápida de las diversas implementaciones de SOAP, lo cual es importante en la interoperabilidad a gran escala.

A continuación se muestran unos ejemplos de mensajes SOAP obtenidos con la herramienta SOAPSpy, ofrecida dentro del servidor de servicios Web de Systinet, denominado *Wasp Server* [Systinet, 2006]. Este mensaje SOAP ha sido producido durante la llamada a un servicio Web que obtiene un fichero de metainformación de un objeto de aprendizaje pasado como parámetro.

En el primer ejemplo (figura 2.18) podemos observar cómo se proporciona al servidor un dato que representa el lugar dónde se encuentra el objeto de aprendizaje, y el servicio Web, nos responde indicando que la operación de obtención de su metainformación se ha realizado con éxito.

Figura 2.18. Ejemplo 1 de mensaje SOAP

En el segundo ejemplo (figura 2.19), se puede observar el caso en el que el objeto de aprendizaje no posee en su interior ningún fichero de metainformación.

Input message

Following message was sent to the service http://localhost:6060/DescomprimirLO/

```
<?xml version="1.0" encoding="UTF-8"?>
<e:Envelope xmlns:e="http://schemas.xmlsoap.org/soap/envelope/">
  <e:Body>
    <ns0:p0 xmlns:ns0="http://systinet.com/xsd/SchemaTypes/">c:\Repositoriol\a.zip</ns0:p0>
  </e:Body>
</e:Envelope>
```

Output message

Following message has been received

```
<?xml version="1.0" encoding="UTF-8"?>
<e:Envelope xmlns:d="http://www.w3.org/2001/XMLSchema" xmlns:e="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns:wn0="http://systinet.com/xsd/SchemaTypes/"
 xmlns:wn1="http://idoox.com/interface" xmlns:wn2="http://systinet.com/wsdl/aplicacion/common/">
  <e:Body>
    <wn0:DescomprimeLOBean_Response i:type="wn2:DescomprimeLOBean">
      <wn2:codError i:type="d:int">104</wn2:codError>
      <wn2:hayError i:type="d:boolean">1</wn2:hayError>
      <wn2:mensaje i:type="d:string">El fichero 'c:\Repositoriol\a.zip' no tiene fichero de meta-informaci?n</wn2:mensaje>
    </wn0:DescomprimeLOBean_Response>
  </e:Body>
</e:Envelope>
```

Figura 2.19. Ejemplo 2 de mensaje SOAP

WSDL (Web Service Description Language)

Creado originalmente por IBM, Microsoft y Ariba (actualmente en la versión 1.2, aunque la recomendación W3C es de la versión 1.1 [W3C, 2001a]). Tiene un propósito similar al de los IDL (*Interface Definition Language*) y de las plataformas *middleware*, es decir, ofrecer un software de conectividad que permite ofrecer un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Un archivo WSDL es un documento XML que describe los servicios Web, en particular sus interfaces. Como característica que lo diferencia de los IDL, es que WSDL debe definir los mecanismos de acceso (protocolos) a los servicios Web. Otra característica diferenciadora es la necesidad de definir (en la especificación) la localización del servicio (puntos finales). La separación de interfaces y enlaces de protocolos, y la necesidad de incluir información de localización permite la definición de especificaciones modulares. WSDL permite definir interfaces más complejas y expresivas, permitiendo definiciones de interacciones asíncronas y diferentes paradigmas de interacción, y la posibilidad de combinar o agrupar operaciones.

WSDL es un formato XML que describe los servicios de red como un conjunto de *endpoints* que procesan mensajes contenedores de información orientada tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de forma abstracta, y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un *endpoint* de red. Los *endpoints* concretos relacionados se combinan en *endpoints* abstractos (servicios).

El lenguaje WSDL es extensible, lo que permite la descripción de *endpoints* de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse.

El documento WSDL de un servicio proporciona dos piezas de información básicas: (1) una parte o interface abstracta (independiente de la aplicación) y (2) una parte específica, que define los enlaces a protocolos e información de los puntos finales de acceso al servicio.

La parte abstracta está compuesta por:

- Definiciones de *port types*: análogos a los interfaces en IDL. Cada *port type* es una colección lógica de *operations*.

- *Operations*: Cada *operation* define un intercambio simple de mensajes.

- *Message*: Un *message* es una unidad de comunicación que representa un intercambio de datos en una única transmisión lógica.

- *Types*: Un sistema de tipos (*types*) usados por las *operations* (por defecto XML Schema).

La parte específica está compuesta por:

- Definiciones de *Bindings*: se especifica la codificación de los mensajes, y los enlaces a protocolos de todas las operaciones y mensajes definida en un *port type*.

- Definiciones de *Ports*: se especifica en qué dirección (URI) se puede acceder a la implementación del *port type*. Definen un punto final (lugar de la red) donde está el servicio.

- Definiciones de *Services*: definen una agrupación de *Ports*.

Esta es la forma que tiene WSDL de describir los servicios Web, en términos de los mensajes que acepta y genera. Actúa como contrato entre un consumidor (cliente) y dicho servicio. WSDL puede describir puntos finales y sus operaciones sin especificar el formato de los mensajes o los protocolos de red (SOAP, HTTP-GET/POST y MIME) a los cuales el punto final esta ligado.

El lenguaje de descripción de servicios Web es, por lo tanto, el equivalente a un resumen diseñado en XML, en el cual se describen los servicios Web, indicando dónde se ubican y la forma de invocarlos. A continuación se muestra un ejemplo de documento WSDL.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- PARTE ABSTRACTA -

```
<definitions name="aplicacion.server.descomprimir.DescomprimirLO"
   targetNamespace="http://systinet.com/wsdl/aplicacion/server/descomprimir/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:map="http://systinet.com/mapping/"
   xmlns:ns0="http://systinet.com/xsd/SchemaTypes/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://systinet.com/wsdl/aplicacion/server/descomprimir/">
   <types>
      <xsd:schema elementFormDefault="qualified"
         targetNamespace="http://systinet.com/wsdl/aplicacion/common/"
         xmlns:map="http://systinet.com/mapping/"
         xmlns:tns="http://systinet.com/wsdl/aplicacion/common/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <xsd:complexType name="DescomprimeLOBean">
            <xsd:annotation>
               <xsd:appinfo>
                  <map:java-type
                  name="aplicacion.common.DescomprimeLOBean"/>
               </xsd:appinfo>
            </xsd:annotation>
            <xsd:sequence>
               <xsd:element name="codError" type="xsd:int"/>
               <xsd:element name="hayError" type="xsd:boolean"/>
               <xsd:element maxOccurs="1" minOccurs="0" name="mensaje"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:schema>
      <xsd:schema elementFormDefault="qualified"
         targetNamespace="http://systinet.com/xsd/SchemaTypes/"
         xmlns:tns="http://systinet.com/xsd/SchemaTypes/"
         xmlns:xns4="http://systinet.com/wsdl/aplicacion/common/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <xsd:import
         namespace="http://systinet.com/wsdl/aplicacion/common/"/>
         <xsd:element name="p0" nillable="true" type="xsd:string"/>
         <xsd:element name="DescomprimeLOBean_Response" nillable="true"
            type="xns4:DescomprimeLOBean"/>
      </xsd:schema>
   </types>
   <message name="DescomprimirLO_descomprime_Request_Soap">
      <part element="ns0:p0" name="p0"/>
   </message>
   <message name="DescomprimirLO_descomprime_Response_Soap">
      <part element="ns0:DescomprimeLOBean_Response" name="response"/>
   </message>
   <portType name="DescomprimirLO">
      <operation name="descomprime" parameterOrder="p0">
         <input message="tns:DescomprimirLO_descomprime_Request_Soap"/>
         <output message="tns:DescomprimirLO_descomprime_Response_Soap"/>
      </operation>
   </portType>
```

- PARTE ESPECÍFICA -

```
   <binding name="DescomprimirLO" type="tns:DescomprimirLO">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="descomprime">
         <map:java-operation name="descomprime"

signature="KExqYXZhL2xhbmcvU3RyaW5nOylMYXBsaWNhY2lvbi9jb21tb24vRGVzY29tcHJpb
WVMT0JlYW47"/>
         <soap:operation

soapAction="http://systinet.com/wsdl/aplicacion/server/descomprimir/DescomprimirLO#descom
prime?KExqYXZhL2xhbmcvU3RyaW5nOylMYXBsaWNhY2lvbi9jb21tb24vRGVzY29tcHJpbWV
MT0JlYW47"
            style="document"/>
         <input>
            <soap:body parts="p0" use="literal"/>
         </input>
         <output>
            <soap:body parts="response" use="literal"/>
         </output>
      </operation>
   </binding>
   <service name="DescomprimirLO">
      <port binding="tns:DescomprimirLO" name="DescomprimirLO">
         <soap:address location="http://portatil:6060/DescomprimirLO/"/>
      </port>
   </service>
</definitions>
```

En el ejemplo mostrado se pueden ver claramente las partes que conformarían los dos ámbitos del mensaje WSDL. Dicho mensaje se corresponde con el mostrado en la sección en que se explicaba el protocolo SOAP.

En la actualidad existen editores avanzados de documentos WSDL, como por ejemplo uno de los ofrecidos por Lomboz [ObjectWeb, 2006], un *plug-in* montado sobre la plataforma de desarrollo Eclipse [2006]. Gracias a este editor podemos ver más claramente el contenido de un documento. En las figuras 2.20 y 2.21, se pueden ver perfectamente los *types*, los *services* y los *Port Types*. También se muestran los dos tipos de mensajes que se pueden obtener de este servicio Web: en la figura 2.20, el de petición, que recibe un dato de tipo “cadena de caracteres”, que corresponde al nombre de un objeto de aprendizaje y en la figura 2.21, el mensaje de respuesta que ofrecerá el servicio Web, respuesta que estará formada por un objeto que contendrá tres datos: un entero, un booleano y una cadena de caracteres.

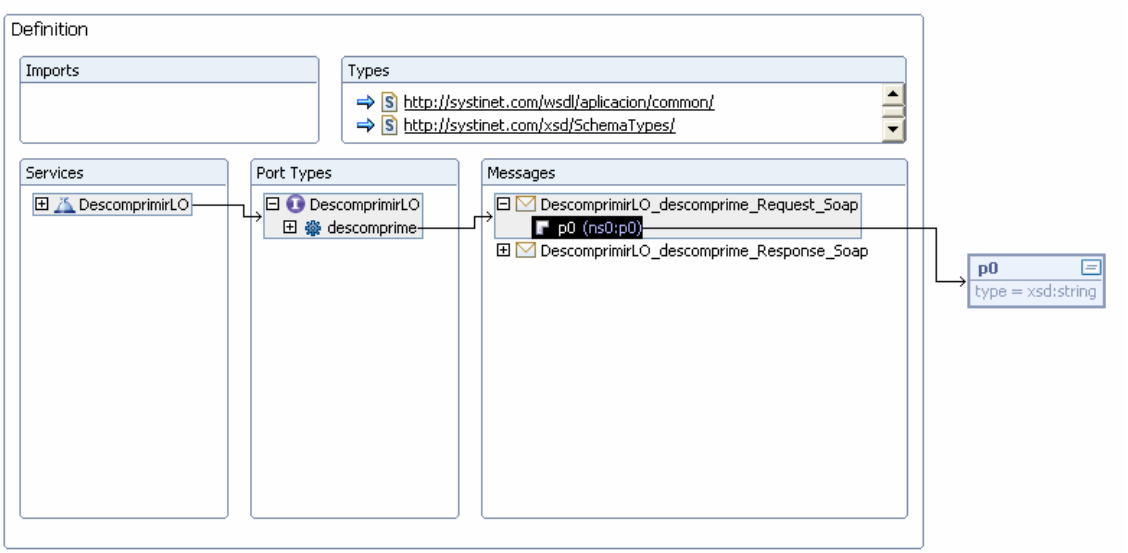

Figura 2.20. Ejemplo de petición en un documento WSDL

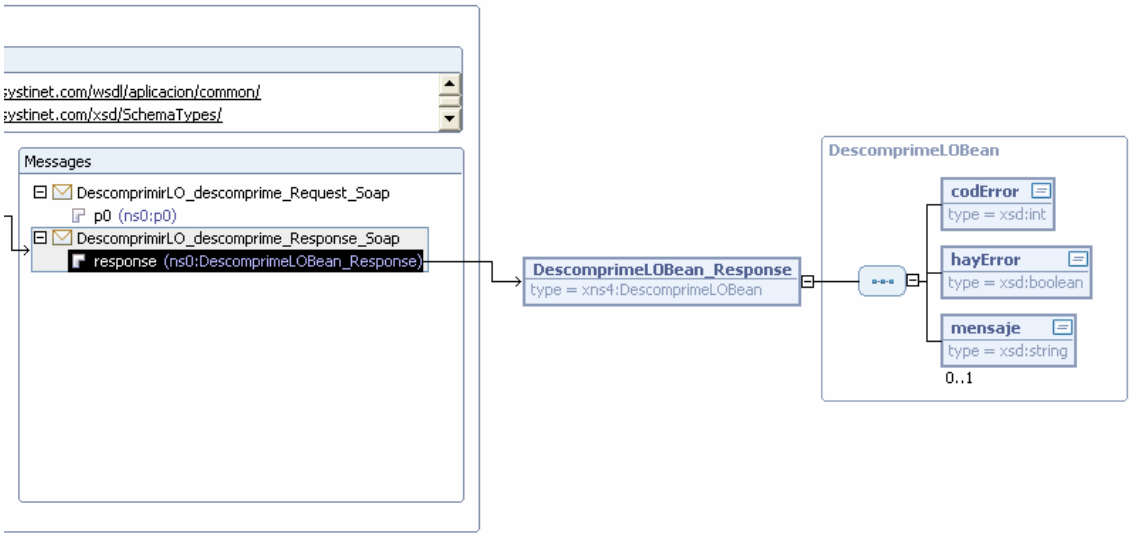

Figura 2.21. Ejemplo de respuesta en un documento WSDL

UDDI (Universal Description, Discovery and Integration)

Siempre ha sido un reto la comunicación entre los negocios a nivel de aplicaciones, dada la enorme cantidad de plataformas, herramientas, mecanismos y procesos que cada cual utiliza. XML se presenta como la solución para el intercambio de datos de una forma transparente. También, la evolución de protocolos como SOAP, ya comentado anteriormente, proporciona una plataforma para el intercambio de servicios sobre la red. Si el mecanismo de comunicación entre las plataformas es el XML, y la forma de comunicación es SOAP, la cuestión que se plantea es cómo encontrar a las organizaciones que proporcionan servicios con los que comunicarse. La respuesta es el UDDI.

UDDI, aunque fue creado originalmente por IBM, Microsoft y Ariba, desde su versión 3 (en 2002), la organización OASIS [OASIS, 2002] se encarga de su mantenimiento.

UDDI se concibió como un registro de negocio, es decir un servicio de directorio y un nombrado sofisticado conjuntamente. Especifica un marco para describir y descubrir servicios Web. UDDI define estructuras de datos y API's para publicar descripciones de servicios en el registro, y métodos de consulta para buscar descripciones publicadas. Las API's de UDDI están especificadas con WSDL y con SOAP Binding, lo que permite acceder a ellas como servicios Web.

La especificación UDDI tiene dos objetivos esenciales: (1) ser un soporte a los desarrolladores para encontrar información sobre servicios Web y poder construir aplicaciones clientes que los invoquen, y (2) facilitar el enlace dinámico de servicios Web, permitiendo consultar referencias y acceder a servicios de interés.

La información en un registro UDDI se almacena en ficheros XML con una estructura jerárquica. Los elementos básicos de esta estructura son:

- *businessEntity*: Es el elemento "*top-level*", describe un negocio o una entidad que ha registrado un servicio en UDDI.

- *businessService*: Describe un servicio Web que ha sido expuesto por una entidad de negocio, soporta el nombrado de un servicio Web y lo asocia con una entidad de negocio y con la información de *binding*. Soporta la asignación de categorías al servicio Web (industria, productos, códigos geográficos, etc.).

- *bindingTemplate*: Describe la información técnica necesaria para enlazar con un servicio Web en particular. Este elemento soporta el nombrado de un servicio Web y su asociación con una entidad de negocio e información de binding. La información de *binding* se describe como un punto de acceso que posee un atributo llamado UrlType utilizado para especificar los siete puntos de entrada: *mailto, http, https, ftp, fax, phone, other*.

- *tModel*: Estructura de metadatos genérica para representar cualquier concepto o construcción (definiciones de protocolos, ficheros WSDL, XML Schemas, espacios de nombres, esquemas de categorías, etc.).

Conceptualmente, la información proporcionada por una organización que ofrece servicios Web en un registro UDDI consta de tres componentes:

- Sección Blanca: Es muy similar a la información que aparece en un directorio telefónico, que incluye dirección, contacto y otros identificadores conocidos.

- Sección Amarilla: Es muy similar a su equivalente telefónico, e incluye categorías de catalogación industrial tradicionales, ubicación geográfica, etc. Mediante el uso de códigos y claves predeterminadas, los proveedores de servicios se pueden registrar y así facilitar a pospotenciales usuarios de sus servicios la búsqueda usando estos índices de clasificación.

- Sección Verde: Contiene la información técnica de los servicios ofrecidos por los proveedores. Se incluyen referencias de especificaciones de servicios Web así como información complementaria para los mecanismos diversos de búsqueda basados en URL.

Un ejemplo de registro UDDI global, lo tenemos en el ofrecido por la empresa Systinet [2006], ya comentada con anterioridad. Como se puede ver en la figura 2.22, existen varios campos que nos posibilitan la búsqueda de servicios, permitiendo así, la obtención de documentos WSDL y el enlace con servicios ofrecidos por otras compañías.

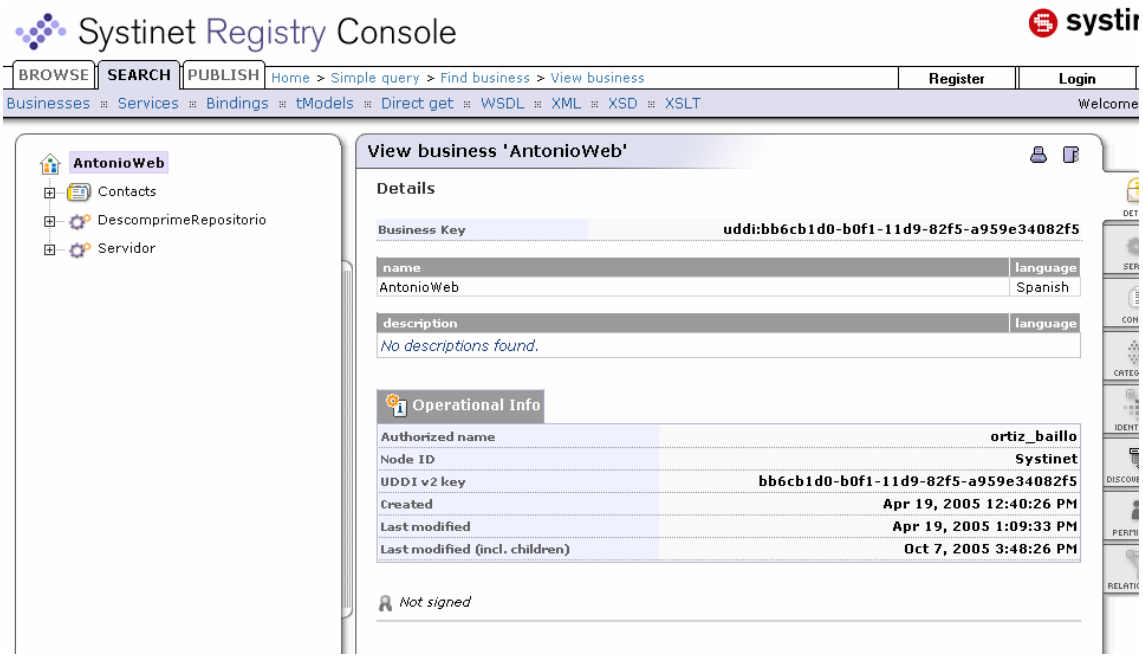

Figura 2.22. Registro UDDI de Systinet

2.2.7 Extensiones y perspectivas de futuro

Según Pelechado [2005], los servicios Web y SOA "prometen la integración universal de aplicaciones"; sin embargo esta afirmación actualmente es bastante vaga, ya que observando con detalle el estado actual nos damos cuenta de algunas "lagunas" en cuanto a fiabilidad, seguridad, orquestación, soporte a sistemas legados y a la semántica. En general, los servicios Web son útiles para necesidades simples, como obtener información de los portales Web, pero las aplicaciones críticas y complejas demandan nuevos estándares todavía en desarrollo.

En la definición de estándares existen principalmente dos grupos implicados en el ámbito de los servicios Web, muy citados a lo largo de todo el documento:

- World Wide Web Consortium (W3C).
- Organization for the Advancement of Structured Information Standards (OASIS).

Estos nuevos estándares, extensiones y protocolos se están proponiendo en el contexto de un marco modular que permita a los desarrolladores utilizar sólo los módulos o aspectos necesarios para sus servicios Web. No existe una propuesta estándar de pila de protocolos; aunque una de las más actualizadas [Wilkes, 2005] es la que se presenta en la tabla 2.6.

Business Domain Specific extensions	Various	**Business Domain**
Distributed Management	WSDM, WS-Manageability	**Management**
Provisioning	WS-Provisioning	
Security	WS-Security	**Security**
Security Policy	WS-SecurityPolicy	
Secure Conversation	WS-SecureConversation	
Trusted Message	WS-Trust	
Federated Identity	WS-Federation	
Portal and Presentation	WSRP	**Portal and Presentation**
Asynchronous Services	ASAP	**Transactions and Business Process**
Transaction	WS-Transactions, WS-Coordination, WS-CAF	
Orchestration	BPEL4WS, WS-CDL	
Events and Notification	WS-Eventing, WS-Notification	**Messaging**
Multiple message Sessions	WS-Enumeration, WS-Transfer	
Routing/Addressing	WS-Addressing, WS-MessageDelivery	
Reliable Messaging	WS-ReliableMessaging, WS-Reliability	
Message Packaging	SOAP, MTOM	
Publication and Discovery	UDDI, WSIL	**Metadata**
Policy	WS-Policy, WS-PolicyAssertions	
Base Service and Message Description	WSDL	
Metadata Retrieval	WS-MetadataExchange	

Tabla 2.6. Pila de protocolos para los servicios Web.

Entre este tipo de protocolos y estándares podemos destacar algunos muy necesarios para la total implantación de los servicios Web:

- Fiabilidad del envío de mensajes: WS-ReliableMessaging, WS-Eventing, WS-Addressing.
- Seguridad: WS-Security ratificado por OASIS y WS-I [WS-I, 2006], SAML, WS-Policy, WS-SecurityPolicy, WS-Trust, WS-SecureConversation.
- Orquestación: BPEL complementada con WS-Transaction, WS-Coordination, WS-Choreography y WS-CAF; diseñados para facilitar la implementación de transacciones y procesos de negocio complejos.

El grado de consenso de la industria en el ámbito de los protocolos de los servicios Web es elevado (al menos en lo que respecta a la infraestructura básica). Aunque se han desarrollado propuestas alternativas en algunas áreas, la formación de grupos de trabajo en W3C u OASIS (se han citado varios de ellos a lo largo de este documento) han llevado muchas veces a la convergencia de todas las partes interesadas.

Existen actualmente algunas áreas donde se mantienen propuestas alternativas, por ejemplo en *Reliable Messaging*, *Orchestration* y *Transaction Coordination*. Estas alternativas se han visto reflejadas por grupos de empresas como IBM/Microsoft por una parte y Sun/Oracle por otra.

Sin embargo, ya desde el año 2004 Microsoft y Sun iniciaron un acercamiento y una mayor cooperación en el ámbito de los servicios Web. Un ejemplo de esta situación ha sido que Sun se unió a BEA, IMB, Microsoft y SAP AG para proponer la última versión de WS-Addressing al W3C [W3C, 2004e]. Lo mismo ocurrió con especificaciones como WS-MessageDelivery [W3C, 2004f], con la actualización de WS-Eventing [W3C, 2006], permitiendo la interoperabilidad de ésta con especificaciones como WS-Notification [IBM, 2004].

Este tipo de movimientos auguran una colaboración futura en aspectos como la coordinación, las transacciones y la coreografía.

También ha existido un solapamiento entre los servicios Web y la iniciativa ebXML [OASIS, 2005a], que utiliza SOAP a nivel de transporte, pero tiene su propio registro y orquestación. Aunque ebXML es un estándar aprobado y robusto, su aplicabilidad es más reducida que los servicios Web. Está considerado como una evolución de EDI [EDI, 2006], cuya principal orientación es el dominio de los negocios electrónicos B2B. Los servicios Web están diseñados para dar soporte a un mayor número de requisitos y escenarios de uso, por lo que ebXML probablemente evolucionará para adoptar los protocolos adicionales de los servicios Web cuando estos maduren y sean probados. Parte del trabajo del grupo ebSOA de OASIS [OASIS, 2005b] es evolucionar la

arquitectura de ebXML y dirigir la transición hacia la adopción de más protocolos basados en los servicios Web.

WS-I (*Web Services Interoperability*) es un grupo de la industria abierto que se creó en 2002 para promover la interoperabilidad de los servicios Web a través de plataformas, sistemas operativos y lenguajes de programación [WS-I, 2005]. En la figura 2.23 se puede ver los estándares en los que WS-I trabaja. WS-I juega un papel muy útil dando soluciones a algunas de las siguientes cuestiones:

- Las especificaciones estándares están siempre abiertas a interpretaciones. WS-I proporciona guías y herramientas para ayudar a medir el grado de fidelidad o ajuste de las implementaciones al estándar y permitir su interoperabilidad.

- Como los estándares evolucionan, es necesario entender qué difiere entre versiones para poder interoperar.

- Se publicarán perfiles (*profiles*) de interoperabilidad que reflejen los requisitos que deben cumplir las implementaciones de los estándares.

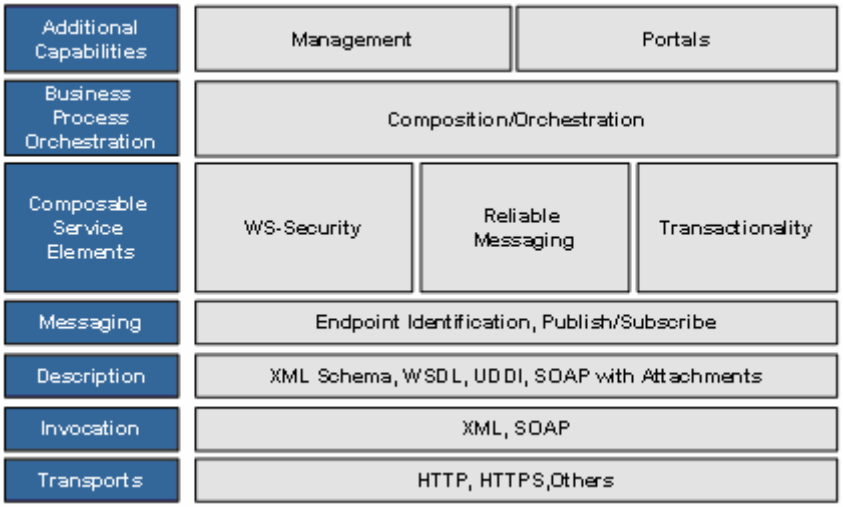

Figura 2.23. Pila de estándares de los servicios Web en los que trabaja WS-I

Aunque la propuesta de protocolos para servicios Web ha sido un área de trabajo muy activa y con un crecimiento muy rápido, su transición a estándares abiertos es inevitablemente mucho más lenta. Existen muy pocos protocolos que hayan finalizado su proceso de estandarización de forma adecuada. En [CBDI, 2006a] se puede encontrar el estado actual de adopción de los protocolos presentados.

2.3 ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

Los sistemas de información son cada vez más complejos y se necesitan sistemas más exigentes que requieren cooperación; y cada vez son menos comunes las soluciones centralizadas y aparecen nuevos métodos de interacción (comunicaciones). El objetivo de las organizaciones es reducir los costes y maximizar la utilización de la tecnología existente; al mismo tiempo las empresas intentan ser más competitivas y avanzar en sus prioridades estratégicas.

Existen dos líneas para conseguir estos propósitos: La heterogeneidad y el cambio. Muchas empresas tienen diferentes sistemas, aplicaciones y arquitecturas de diferentes tecnologías y algunas bastante antiguas. Integrar productos de diferentes proveedores y de diferentes tecnologías es una ardua tarea, por eso cada vez más las aplicaciones están orientadas al servicio. Esto se puede ver en los gráficos del informe del CBDI Forum [CBDI, 2006b], que se muestra en la figura 2.24.

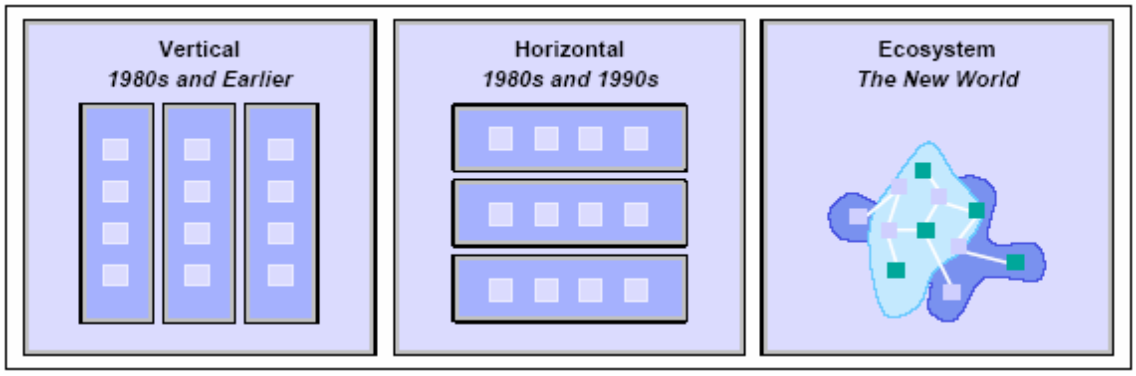

Figura 2.24. Evolución de los procesos de negocio

Por otra parte, en la figura 2.25 se puede apreciar la evolución hacia las arquitecturas orientadas a servicio.

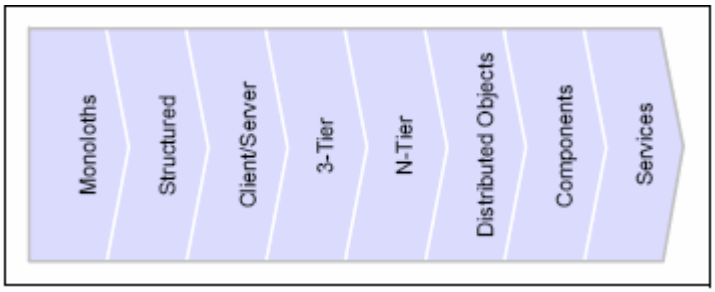

Figura 2.25. Evolución de las arquitecturas de sistemas software

Durante años la industria de la tecnología ha estado luchando para solucionar problemas como la heterogeneidad, la interoperabilidad (para que la solución escogida o la aplicación construida sea independiente de la plataforma y tecnología utilizada), incluso en la forma de recoger los requisitos o que las aplicaciones sean independientes

del protocolo de comunicación. Han ido apareciendo distintas formas de plantear la construcción de aplicaciones, sin lugar a duda una de las más importantes para resolver los problemas anteriores a sido el enfoque de la orientación a objetos, aplicado tanto al análisis, diseño, como programación de aplicaciones, planteando el desarrollo de software como "un problema de dominio y una solución lógica desde la perspectiva de los objetos (cosas, conceptos o entidades)" [Larman, 2001]. Jacobson et al. [1992] definen estos objetos como "características con un número de operaciones y un estado que recuerda los efectos de las operaciones sobre ellos".

En el análisis orientado a objetos, tales objetos son identificados y descritos en el dominio del problema; y en el diseño orientada a objetos son transformados en objetos de software que serán implementados en un lenguaje orientado a objetos. De esta forma se reduce el esfuerzo de análisis y diseño de escenarios complejos y se facilita la reutilización.

A partir de la orientación a objetos han ido apareciendo otros "paradigmas" como la orientación a aspectos [POA, 2006], o la orientación a servicios; un servicio es generalmente implementado como una entidad software que puede ser accedida como una única instancia e interactúa con otras aplicaciones y otros servicios a través de un modelo de comunicación basado en mensajes. Antes de proceder a la descripción del concepto de arquitectura orientada a servicio, conviene recordar la terminología relacionada con los servicios Web:

- **Servicios:** Entidades lógicas con una funcionalidad definida y las reglas establecidas por una o más interfaces que son publicadas.

- **Proveedor de servicio (*Service provider*):** La entidad de software que implementa la especificación del servicio.

- **Consumidor del servicio (*Service consumer (o requestor)*):** La entidad de software que llama al proveedor de servicio. Tradicionalmente llamado "cliente". Un consumidor de servicio puede ser una aplicación u otro servicio.

- **Servicio localizador (*Service locator*):** Un tipo específico de proveedor de servicio que actúa como un registro y permite la búsqueda de interfaces y localizaciones de servicios.

- **Servicio intermediario (*Service broker*):** Un tipo específico de proveedor de servicio que puede pasar las peticiones de un servicio a uno o más proveedores de servicios adicionales.

2.3.1 Concepto de Arquitectura Orientada a Servicios

Existen muchas definiciones de arquitectura, y también existen muchas interpretaciones de estas definiciones. La arquitectura de un sistema, en general, describe su estructura, a través de los siguientes aspectos:

- Los componentes que intervienen como bloques básicos del sistema.
- Conectores que describen los mecanismos de comunicación con otros sistemas y los mecanismos de interconexión entre los propios componentes de la arquitectura.
- Los flujos que muestran como una aplicación utiliza los componentes y los conectores para llevar a cabo su objetivo.

Se utiliza el acrónimo SOA para hacer referencia a la arquitectura orientada a servicio, el término SOA son las siglas inglesas de *Service Oriented Architecture*.

La Arquitectura Orientada a Servicio presenta una ventaja para la construcción de sistemas distribuidos, ya que contempla, la funcionalidad de las aplicaciones como servicios que pueden utilizar otras aplicaciones y otros servicios.

Una Arquitectura Orientada a Servicio está compuesta por elementos funcionales y elementos relacionados con la calidad de servicio, como se puede ver en la figura 2.26 y que se describen a continuación.

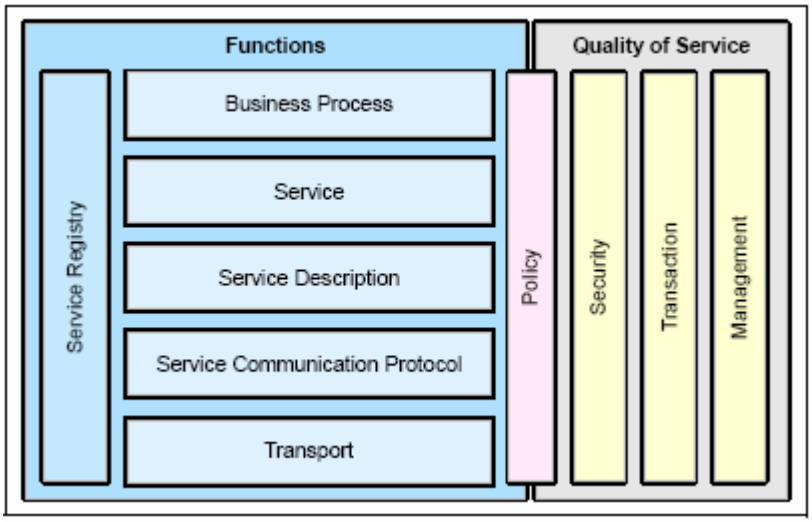

Figura 2.26. Elementos de una Arquitectura Orientada a Servicio

Aspectos funcionales:

- **Transporte:** Es el mecanismo usado para llevar peticiones de servicios desde el consumidor al proveedor del servicio, y las respuestas desde el proveedor del servicio al consumidor del servicio.

- **Protocolo de comunicación del servicio:** Es el mecanismo de comunicación establecido entre el proveedor del servicio y el consumidor del servicio.

- **Descripción del servicio:** Es el esquema establecido para describir qué es el servicio, cómo debe invocarse y que datos son requeridos para la invocación.

- **Servicio:** Describe un servicio que está disponible para utilizarse.

- **Proceso de negocio:** Es una colección de servicios, invocados de una manera particular, en una determinada secuencia y con unas reglas particulares para llevar a cabo la funcionalidad de negocio requerida. Un proceso de negocio puede estar compuesto por servicios de diferente naturaleza e incluso en distintas localizaciones.

- **Registro de Servicio:** Es el repositorio de servicios y las descripciones que son usados por los proveedores de servicio para publicarlos, y para que los consumidores del servicio puedan invocarlos. El registro del servicio puede aportar la funcionalidad a los servicios que necesiten un repositorio centralizado.

Aspectos de la calidad del servicio:

- **Política:** Es un conjunto de condiciones o reglas sobre las cuales un proveedor del servicio hace un servicio disponible a los consumidores.

- **Seguridad:** Es el conjunto de reglas que pueden ser aplicadas para la identificación, autorización y el control de acceso a los consumidores de servicios.

- **Transacción:** Es el conjunto de atributos que pueden ser aplicados a un grupo de servicios para conseguir un resultado consistente. Por ejemplo si un grupo de tres servicios tienen que terminar para completar la función, todos tienen que estar completados y haber terminado su ejecución.

- **Gestión:** Es el conjunto de atributos que pueden ser aplicados para manejar a los proveedores del servicio o a los consumidores.

2.3.2 Colaboración en una Arquitectura Orientada a Servicios

En la figura 2.27 se pueden ver los elementos de colaboración existentes en una Arquitectura Orientada a Servicio:

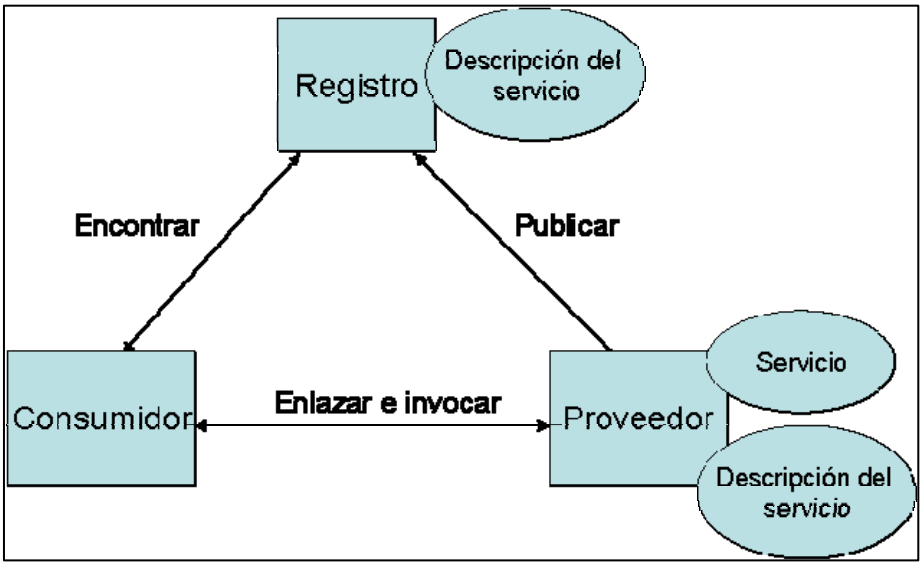

Figura 2.27. Elementos de colaboración en una Arquitectura SOA

Los roles de una Arquitectura Orientada a Servicio son:

- **Consumidor de servicio:** El consumidor de servicio es una aplicación, un módulo de software u otro servicio que requiere un servicio. Inicia la búsqueda en el registro de servicio, enlaza con el servicio a través del transporte y ejecuta la función del servicio de acuerdo con las reglas establecidas.

- **Proveedor de servicios:** El proveedor de servicios es una entidad que se puede acceder a través de la red y que acepta y ejecuta peticiones de los consumidores. Publica las interfaces de los servicios en el registro de servicios para que los consumidores puedan descubrirlos y puedan acceder a ellos.

- **Registro de servicios:** Un registro de servicios es el que permite que los servicios puedan ser descubiertos. Contiene un repositorio con los registros que están disponibles y permite la búsqueda de los proveedores de los servicios a través de las interfaces que han sido establecidas y que son de interés para los consumidores.

Las operaciones dentro de una arquitectura orientada a servicios son:

- **Publicación:** Para que los servicios puedan estar accesibles, un servicio tiene que tener una descripción, que debe ser publicada para que pueda ser descubierta e invocada por un consumidor de servicio.

- **Localización:** Un consumidor del servicio puede localizar un servicio realizando una búsqueda sobre el registro de servicios que cumpla algún criterio.

- **Enlazar e invocar:** Después de recoger la descripción del servicio, el consumidor del servicio puede invocar el servicio de acuerdo con la información de la propia descripción del servicio.

En la figura 2.28 se puede ver más claramente el entorno y forma de colaboración:

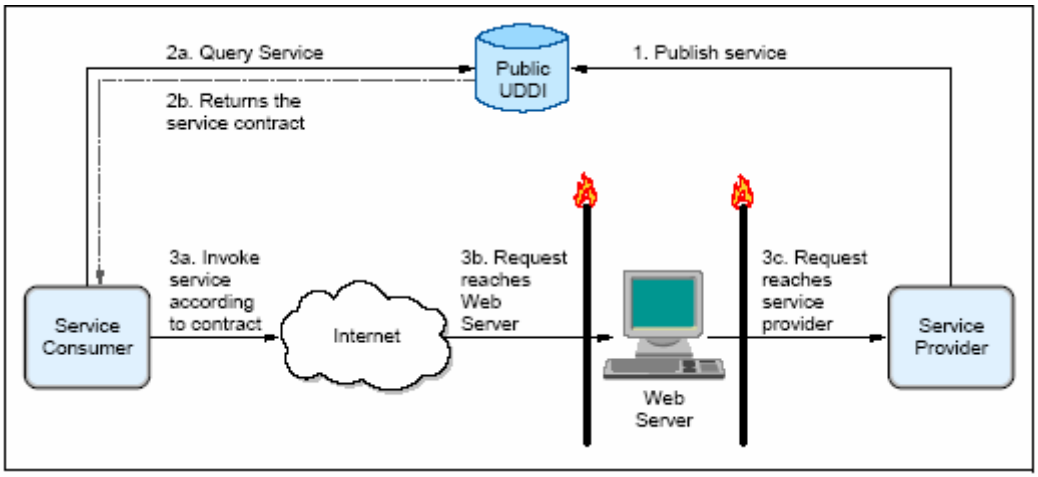

Figura 2.28. Colaboración en una Arquitectura SOA

Cómo ya se comentó en apartados anteriores, para poder realizar estas operaciones, existen distintas tecnologías; así, para realizar la descripción del servicio que especifica la forma de acceder al consumidor y poder interactuar con el proveedor se utiliza WSDL (*Web Services Description Lenguaje*) [W3C, 2001a], esta especificación establece el qué, el dónde y el cómo se accede a un determinado servicio. Y para poder hacer la localización, la integración y poder enlazar e invocar a un Servicio Web se utiliza UDDI (*Universal Description, Discovery and Integration*) [UDDI, 2006].

2.3.3 Características de una Arquitectura Orientada a Servicios

Para que el funcionamiento de una arquitectura orientada a servicio sea dinámica, tiene que cumplir las siguientes características [McGovern et al., 2001]:

- Los servicios tienen que ser modulares.
- Los servicios tienen que soportar la interoperabilidad.
- Los servicios tienen que tener la descripción perfectamente establecida.
- Los servicios tienen que ser transparentes a la localización.
- Los servicios tienen que ser independientes del lenguaje de implementación.
- Los servicios tienen que ser transparentes al protocolo de comunicación.
- Los servicios tienen que ser independientes del Sistema Operativo y del hardware utilizado.

2.3.4 Beneficios de una Arquitectura SOA

Como ya se ha indicado, los procesos de negocio de una organización se van creando teniendo en cuenta la habilidad para cambiar rápidamente, la heterogeneidad y sobre todo la necesidad de reducción de costes. Para recordar la competitividad existente entre las distintas organizaciones, deben poder crear procesos de negocio y ofrecer servicios a sus clientes de forma que se puedan adaptar rápidamente a factores internos como adquisiciones, fusiones y reestructuraciones, o factores externos como son los requerimientos del mercado y de los clientes. Para las compañías es necesario llevar un equilibrio entre el coste, la efectividad, la calidad de sus procesos y la flexibilidad de la infraestructura tecnológica de que disponen.

Con una arquitectura orientada a servicio (SOA) se pueden alcanzar algunos beneficios para ayudar a las distintas organizaciones a conseguir unos procesos de negocios exitosos y dinámicos:

- **Solventar los problemas existentes:** Utilizando una arquitectura SOA se establece una capa de abstracción que permite a una organización continuar ofreciendo una innovación en tecnología, encapsulando los problemas en servicios que ofrecen como funciones de negocio. Las organizaciones pueden

continuar obteniendo valor utilizando recursos externos a través de los servicios en lugar de invertir y reconstruir las soluciones existentes.

- **Fácil de integrar y gestionar la complejidad:** El punto de integración en una arquitectura SOA es la especificación del servicio y no la implementación. De esta forma aporta la transparencia de la implementación y minimiza el impacto cuando la infraestructura o la implementación existente cambian. Mediante la especificación del servicio con las funciones implementadas o los problemas que resuelve, incluso en distintos sistemas, la integración resulta sencilla.

- **Acelerar la puesta en producción de los sistemas (*time-to-market*):** La habilidad de componer nuevos servicios haciendo uso de los ya existentes aporta una ventaja significativa a una compañía que necesita responder de una forma rápida a la demanda de negocio que le requieren. El ciclo de desarrollo se puede llegar a reducir, aunque también es importante saber establecer bien una base de requisitos funcionales y de pruebas.

- **Reducir costes y aumentar la reutilización:** Realizando una base de servicios que se pueden exponer y se pueden reutilizar, incluso si los servicios los ofrecen otras compañías, resultan mucho menos costosos los desarrollos, se necesita menos tiempo de codificación, se aumenta la reutilización en los desarrollos, existe menos duplicación de recursos, y por lo tanto, se aumenta el potencial y se reducen los costes.

- **Estar preparados para el cambio:** La arquitectura SOA permite a las organizaciones estar preparadas para el futuro. Los procesos de negocio se pueden crear mucho más fácilmente si se piensan y se crean en base a servicios gestionados y se dejan las interfaces de interconexión totalmente definidas, permitirá a las compañías seguir creciendo sin necesidad de invertir de nuevo en los mismos desarrollos, y además permite a los sistemas existentes seguir creciendo.

Con esto se puede comprobar que utilizando las arquitecturas SOA se aporta gran valor, pero también hay que recordar que migrar a una arquitectura orientada a servicio no es una tarea trivial. Más que una migración de toda una organización a una arquitectura SOA, se recomienda migrar el conjunto de funciones o artefactos que necesitan interoperar con otras organizaciones, y empezar a utilizar SOA para futuros desarrollos.

También se pueden ver los beneficios de una Arquitectura SOA desde el punto de vista empresarial; algunas de ellas son las siguientes:

- **Eficiencia:** Transforma los procesos de negocio en servicios compartidos con un menor coste de mantenimiento, mayor ROI (*Return Of Investment*).

- **Capacidad de respuesta:** Rápida adaptación y despliegue de servicios, clave para responder a las demandas de clientes, colaboradores y empleados.

- **Adaptabilidad:** Facilita la adopción de cambios añadiendo flexibilidad y reduciendo el esfuerzo.

2.3.5 Servicios Web y Arquitectura Orientada a Servicios

En el apartado 2.2 se presentó con detalle lo que eran los servicios Web y las tecnologías les ofrecen soporte; en este apartado se justifica porque los servicios Web pueden llegar a ser una parte fundamental de una Arquitectura Orientada a Servicio.

Dentro de un proceso de negocio puede ser necesario utilizar funcionalidades de distintos sistemas y distintas localizaciones; para completar estas funcionalidades se utilizan los Servicios Web, que también tienen que ser identificados durante el proceso de análisis de una arquitectura orientada a servicio. Cada servicio tiene que estar bien definido mediante una interfaz (WSDL) para que pueda ser publicado, localizado e invocado. Dependiendo del proceso de negocio, el servicio puede ser publicado para que otras empresas puedan utilizarlo, o internamente para ser utilizado en los procesos de negocio internos de una determinada compañía.

Los servicios Web son una tecnología altamente adaptable a las necesidades de implementación de una Arquitectura Orientada a Servicio. En esencia, los servicios Web son la implementación de una especificación bien definida de una funcionalidad, es decir, son aplicaciones modulares que aportan una lógica del proceso de negocio como servicio que puede ser publicado, localizado e invocado en Internet [IBM, 2006]. Basados en los estándares XML [W3C, 1998], los servicios Web pueden ser desarrollados usando cualquier lenguaje de programación, cualquier protocolo y cualquier plataforma. Los Servicios Web pueden ser localizados y utilizados en cualquier momento, desde cualquier localización y usando cualquier protocolo y plataforma.

Pero es importante remarcar que los servicios Web no son la única tecnología que es usada para implementar una Arquitectura Orientada a Servicio. Existen ejemplos de organizaciones que utilizan con éxito una Arquitectura Orientada a Servicios donde utilizan, además de los servicios Web otras tecnologías de intercambio de mensajes y de acceso a funciones remotas haciendo uso del estándar XML, como puede ser el

protocolo XML RPC [Winer, 1999], que funciona exactamente igual que el protocolo RPC (*Remote Procedure Call*), a través de un túnel HTTP.

2.3.6 Patrones de diseño de una Arquitectura Orientada a Servicios

También para la construcción de aplicaciones orientadas a servicio utilizando arquitecturas SOA, se han desarrollado una serie de patrones de diseño de software [GoF, 2003]. Con la utilización de patrones, que se pueden utilizar con cualquier metodología, la Arquitectura Orientada a Servicio resultante contendrá un nivel de abstracción mayor, asegurándose una mejor consistencia y rendimiento ya se han utilizado buenas prácticas de diseño.

Los patrones de diseño para construir una arquitectura orientada a servicio se pueden dividir en cinco categorías [Monday, 2003]:

1. **Aprendizaje:** Sirven para entender el entorno de los servicios Web. Dentro de esta categoría podemos encontrar:
   - *Service-Oriented Architecture*: Es el patrón que forma la arquitectura de los servicios Web como ya hemos visto anteriormente.
   - *Architecture Adapter*: Se puede ver como un patrón genérico que facilita la comunicación entre arquitecturas.
   - *Service Directory*: Este patrón facilita la transparencia en la localización de servicios, permitiendo realizar robustas interfaces para encontrar el servicio que realmente se quiere.

2. **Adaptación:** Estos patrones son los llamados básicos para conocer el funcionamiento del entorno de los servicios Web. En esta categoría nos encontramos:
   - *Bussines Object*: Un *business object* engloba a un concepto de negocio del mundo real como puede ser un cliente, una compañía o un producto, y lo que pretende este patrón es trasladar el concepto de objeto de negocio dentro del paradigma de los servicios Web.
   - *Business Process*: Este patrón se utiliza para tratar con procesos de negocio. En este momento existen dos especificaciones:
     - *Business Process Execution Lenguaje* (BPEL) propuesto por Bea Systems, IBM y Microsoft.
     - *Business Process Modeling Languaje* (BPML) propuesto por el resto de compañías que no están en el grupo anterior como pueden ser WebMethods, SeeBeyond, etc.
   - *Bussines Object Collection*: Con este patrón se pueden realizar composiciones de procesos de negocio.

- *Asynchronous Business Process*: Este patrón es la evolución del patrón anterior *Bussines Process*.

3. **Cambios:** Aunque los servicios Web permiten llamadas asíncronas, la implementación del servicio puede estar basado en paso de mensajes; también son importantes los servicios basados en eventos, estos patrones se basan en patrones tradicionales como el *Observer* o el patrón Publicación/Suscripción. En esta categoría podemos encontrar:
   - *Event Monitor*: Es un patrón para crear formas efectivas para integrar aplicaciones sin la intervención de otros componentes. El escenario más común donde se utiliza este patrón es para aplicaciones EAI (*Enterprise Application Integration*) [EAI, 2006].
   - *Observer Services*: Este patrón representa la manera más natural de detectar cambios y actuar en consecuencia.
   - *Publish/Subscribe Services*: Es la evolución del patrón *Observer*, mientras que el patrón *Observer* se base en el registro, el patrón Publicación/Suscripción se base en notificaciones, esto permite que distintos servicios puedan enviar la misma notificación.

4. **Redefinición:** Estos patrones permiten acceder al comportamiento de un servicio que está implementado en un lenguaje. Ayudan a entender el entorno del servicio Web y a moldear este entorno de acuerdo con nuestras necesidades. En esta categoría podemos encontrar:
   - *Physical Tires*: Este patrón ayuda a estructurar mejor la lógica de negocio de los servicios Web, e incluso se puede utilizar para controlar le flujo de negociaciones que puede llegar a producirse utilizando el patrón Publicación/Suscripción.
   - *Connector*: Este patrón se suele utilizar con el anterior para resolver los posibles problemas que surgen en la suscripción.
   - *Faux Implementation*: Es una alternativa para resolver los problemas que surgen en la utilización de eventos en los servicios Web. Es simplemente un "*socket abierto*" que recibe conexiones y aporta las respuestas para los distintos eventos.

5. **Flexibilidad:** Para crear servicios más flexibles y optimizados. En esta categoría se encuentran:
   - *Service Factory*: Es uno de los patrones más importantes y permite la selección de servicios y aporta flexibilidad en la instanciación de los componentes que crean los servicios Web. Este patrón también se suele utilizar con el patrón *Service Cache* para aportar una mayor flexibilidad en el mantenimiento de las aplicaciones que utilizan servicios Web, aportando un mayor ROI a las aplicaciones.

- *Data Transfer Object*: Este patrón aporta rendimiento, ya que permite recoger múltiples datos y enviarlos en una única llamada, reduciendo el número de conexiones que el cliente tiene que hacer al servidor.
- *Data Transfer Collection*: Este es una extensión del anterior, ya que el patrón *Data Transfer Object* se puede aplicar a una colección de objetos de negocio. Este objeto puede devolver un grupo de atributos comunes de una colección de objetos.
- *Partial Population*: Este patrón permite a los clientes seleccionar únicamente los datos que son necesarios para sus necesidades y sólo recuperar del servidor lo necesario. Este patrón además de rendimiento aporta mayor ancho de banda en la red.

Algunos patrones utilizan otros; por ejemplo, el patrón *Business Process* usa los patrones *Business Object* y el *Business Object Collection*. Y el *Service-Oriented Architecture* hace uso de los patrones *Service Directory* y *Architecture Adapter*.

2.3.7 Business Process Execution Language (BPEL)

La implementación ideal de un servicio exige resolver algunos inconvenientes técnicos inherentes a su modelo:

- Los tiempos de llamada no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, etc. Esto necesariamente implica la utilización de mensajería fiable.

- La respuesta del servicio se ve afectada directamente por aspectos externos, como problemas en la red, configuración, etc. Estos deben ser tenidos en cuenta en el diseño, desarrollándose los mecanismos de contingencia que eviten la parálisis de las aplicaciones y servicios que dependen de él.

- Debe manejar comunicaciones no fiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etc.

Según lo anterior, se puede imaginar que la construcción de un servicio es una tarea mucho más complicada que la de un simple componente distribuido. Por esto, el servicio debe publicar una interfaz (por ejemplo, utilizando WSDL) fácilmente localizable en la red. Esta interfaz debe servir como un contrato de servicio, donde se describen cada una de las funciones que provee, e incluso los niveles de prestación de servicio (SLA: *Service Level Agreement*). Esta interfaz debe estar claramente documentada de manera que sea muy fácil implementar una conexión.

Cuando se usan múltiples servicios para implementar un sistema, es muy fácil que la comunicación entre estos sea difícil de controlar. Por ejemplo, se puede tener un servicio que llama a otros servicios, algunos de los cuales llama a otros servicios, y de esta manera, muy fácilmente el sistema se vuelve inmanejable y un sistema grande puede terminar con múltiples dependencias. Detectar un problema de rendimiento o funcionalidad se puede volver muy complicado.

La forma tradicional para crear de manera rápida y sencilla nuevas aplicaciones que utilicen las ya existentes son los sistemas de gestión de flujos de trabajo (*workflows*).

Un flujo de trabajo especifica aspectos tales como:

- La secuencia de acciones a realizar por cada entidad.
- Los datos intercambiados entre las “entidades (aplicaciones, servicios)” y la manera en que deben ser transformados.
- Reglas para la toma de decisiones.
- Restricciones a satisfacer.

Algunas ventajas que ofrecen los sistemas de flujos de trabajo son las siguientes:

- Crear un nuevo proceso de negocio no implica programar (al menos, en gran parte).
- El sistema gestor del *workflow* se encarga de algunas tareas complejas como la transaccionalidad o la mensajería.
- La respuesta a cambios es más rápida.
- No es necesario escribir cada vez código hecho a medida para las aplicaciones (esto no siempre es cierto pero, al menos, una vez añadida una funcionalidad, queda disponible para todos los procesos de negocio).
- Facilita la implantación de políticas globales: autenticación, seguridad, etc.

La nueva evolución de los *workflows* se basa en la orquestación de los servicios Web: conectar servicios entre sí para obtener procesos de negocio de alto nivel.

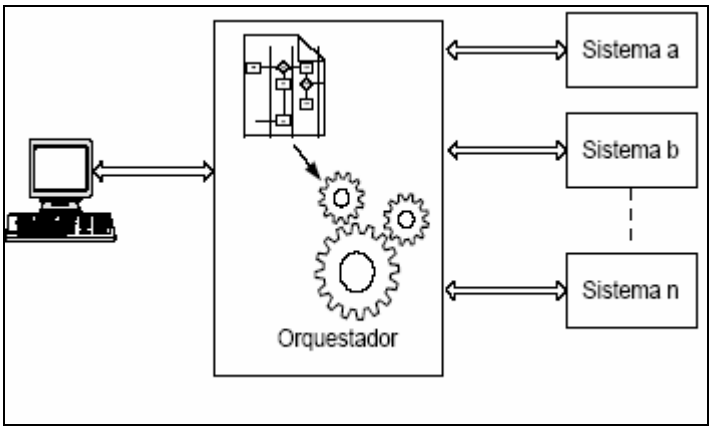

Figura 2.29. Orquestación de Servicios Web

Orquestación es un término relativamente nuevo definido por analogía a la ejecución de una pieza musical por parte de una orquesta. Una orquesta no es un conjunto de músicos tocando cada uno su propio instrumento al mismo tiempo, sino que es mucho más que eso, una orquesta en el acto de ejecutar una obra, es un conjunto de instrumentos que bajo una bien definida coreografía, la interpretan juntos de acuerdo a su arreglo musical. Cada músico debe entrar y salir a su tiempo ejecutando la parte que le corresponde en ese momento, según el director de la orquesta de acuerdo a la coreografía de la obra.

De la misma manera, un proceso de negocio debe ser orquestado bajo su propia coreografía y mediante un orquestador dirigir la ejecución de los componentes que actúan en conjunto para que, ejecutando cada cual lo que le corresponde en el momento indicado, compongan la ejecución de ese proceso de negocio (figura 2.29).

La orquestación proporciona secuencia y sincronización: el "qué" ejecutar y el "cuando" hacerlo. El orquestador dirige la ejecución de una cierta coreografía para articular un proceso de negocio.

2.3.7.1 Orquestación versus Coreografía

Como hemos visto los servicios Web se pueden combinar de dos maneras:

- Orquestación
- Coreografía

En la orquestación, que suele ser usada para procesos de negocios privados o para procesos centrales, donde un proceso coordina un grupo de diferentes operaciones, los servicios involucrados no "conocen" (y no necesitan conocer) todo el proceso del que forman parte. Sólo el coordinador central de la orquestación conoce el objetivo global;

de esta forma, la orquestación es centralizada con definiciones explícitas de las operaciones y con el orden de invocación de los servicios Web (figura 2.30).

Figura 2.30. Composición de Servicios Web con orquestación

En la coreografía no existe un coordinador central, cada servicio Web involucrado en la coreografía conoce exactamente cuando se tiene que ejecutar y las operaciones y con quien tiene que interactuar. La coreografía se utiliza en los procesos de negocio públicos con intercambios de mensajes. Todos los servicios Web necesitan conocer quien es el siguiente servicio que se va a ejecutar, qué operación, que mensaje se intercambia y el tiempo de ejecución (figura 2.31).

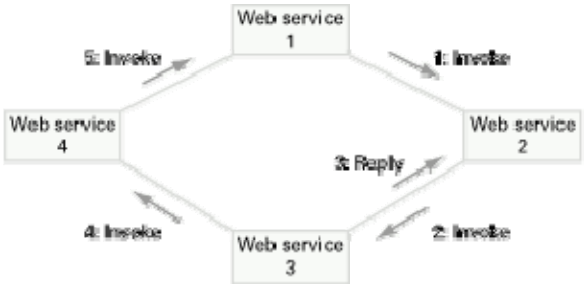

Figura 2.31. Composición de Servicios Web con coreografía

Desde la perspectiva de la composición de los servicios Web para la ejecución de procesos de negocio, la orquestación es más flexible y tiene las siguientes ventajas sobre la coreografía:

- La coordinación es centralizada conducida por un coordinador conocido.

- Los servicios Web pueden ser incorporados sin afectar al proceso al que se incorporan, y no tienen porque conocer cual es el objetivo del proceso al que se están incorporando.

Para especificar la composición de servicios Web se puede utilizar el lenguaje BPEL [2006], que soporta dos formas de describir los procesos de negocio que soportan la orquestación y la coreografía:

- **Procesos ejecutables:** Permiten especificar exactamente los detalles de los procesos de negocio. Estos procesos cumplen con el paradigma de la orquestación.

- **Protocolos abstractos de negocio:** Permiten la especificación del intercambio de mensajes públicos únicamente entre participantes. No incluyen detalles internos de los flujos de los procesos y no son ejecutables. Siguen el paradigma de la coreografía.

Después de las distintas normas propuestas parece que la norma BPEL4WS (Business Process Execution Language for Web Services) [IBM, 2005] va a ser el estándar de la orquestación de los servicios Web. Es utilizado para procesos abstractos (que definen conversaciones y protocolos para el uso de un servicio Web o de la forma de colaborar de varios servicios Web) y para procesos ejecutables (que son flujos de trabajo dónde cada entidad involucrada es un servicio Web y que, a su vez, ofrecen al exterior una interfaz de servicio Web).

Normalmente un proceso BPEL4WS se compone de:

- Un fichero con el proceso a ejecutar (.BPEL)

- Una serie de ficheros WSDL de apoyo (definiciones).

Las actividades en BPEL pueden verse como nodos en un grafo, dónde los enlaces entre nodos determinan el flujo de control entre actividades. Las actividades tienen dos elementos opcionales para establecer dependencias entre los enlaces del grafo que conforman el flujo:

- **Target.** La actividad es destino del enlace especificado. Esto quiere decir que la actividad no comenzará hasta que dicho enlace se active (normalmente, al finalizar otra actividad).

- **Source.** La actividad es fuente del enlace especificado. Cuando la actividad termine, activará el enlace, posiblemente desencadenando la ejecución de otras actividades que tengan ese enlace como target.

Manejando estas dependencias, es posible construir estructuras condicionales, flujos en paralelo, etc. En definitiva, lo que se trata es de fijar unas directivas para que puedan cooperar todos los procesos empresariales, y así organizados mediante la utilización de distintos flujos de trabajo, se consigue el objetivo que se pretende.

El lenguaje BPEL está basado en el esquema XML [W3C, 2004c], en el protocolo SOAP (*Simple Object Access Protocol*) [W3C, 2003], y el lenguaje de descripción de servicios Web WSDL (*Web Services Descripton Language*) [W3C, 2001a] y permite lo siguiente:

- Enviar mensajes XML y recibir mensajes síncrona y asíncronamente desde servicios Web remotos.
- Manipular estructuras de datos XML
- Manejar eventos y excepciones
- Diseñar flujos de procesos
- Deshacer la continuidad del proceso en un flujo definido cuando ocurre una excepción

Diseño de un proceso de negocio

Un proceso especifica el orden exacto en el cual los servicios Web participan y deben ser invocados secuencialmente o en paralelo. También se pueden expresar comportamientos condicionales. Por ejemplo, una invocación de un servicio Web puede depender del valor de una invocación anterior. Se pueden construir bucles, declarar variables o copiar y asignar valores. Como los procesos se pueden representar gráficamente, ya que son esencialmente grafos de actividades, se pueden expresar usando diagramas de actividad UML [Jacobson et al., 2000].

En un escenario típico, los procesos BPEL reciben una petición, y para completarla, el proceso invoca a los servicios Web que participan en el proceso y después responde al que ha hecho la petición original. En todo el ciclo del proceso, como existe comunicación entre distintos servicios Web, hay que tener muy bien definida la descripción WSDL.

A continuación se va a presentar un ejemplo de un proceso BPEL: un proceso BPEL consiste en “pasos” y a cada paso se denomina “actividad”; una actividad se basa en primitivas que representan los constructores básicos y son usadas para tareas comunes como las siguientes:

- Invocar a otros servicios Web, usando la etiqueta <*invoke*>.
- Esperando a que el cliente invoque al proceso enviando un mensaje, usando <*receive*> (recibiendo una petición).

- Generando una respuesta para las operaciones síncronas, usando <*reply*>.
- Manipulando variables, usando <*assign*>.
- Indicando errores y excepciones, usando <*throw*>.
- Esperando por algún espacio de tiempo, usando <*wait*>.
- Terminando el proceso, usando <*terminate*>.

Se pueden combinar estas y otras actividades primitivas para definir algoritmos complejos que especifiquen exactamente los pasos de los procesos de negocio. Para combinar estas actividades primitivas BPEL soporta algunas estructuras, las más importantes son:

- Secuencia <*secuence*>, permite la definición de un conjunto de actividades que serán invocadas en una secuencia ordenada.
- Flujo <*flow*>, para definir un conjunto de actividades que serán invocadas en paralelo.
- Estructura <*switch*>, para implementar caminos alternativos.
- Estructura <*while*>, para definir bucles.
- La habilidad de seleccionar uno o más caminos alternativos, usando <*pick*>.

En un proceso BPEL también se pueden utilizar enlaces, usando <*partnerLink*>, y declarar variables utilizando <*variable*>.

Para entender como funciona un proceso de negocio con BPEL, se va a describir un proceso de negocio para los empleados de una agencia de viajes [Oracle, 2005]: El cliente invoca al proceso especificando el nombre del empleado, el destino, la fecha de salida y la fecha de llegada. El proceso BPEL primero comprueba el estado del empleado, asumiendo que el servicio Web existe a través del cual puede comprobarse esta situación. Después el proceso BPEL comprobará el precio para el billete de vuelo en dos compañías (American Airlines y Delta Airlines), de nuevo se asume que ambas compañías de vuelo tienen servicios Web para poder hacer esta operación. Finalmente, el proceso BPEL seleccionará el precio más bajo y retornará el plan de vuelo al cliente.

Cuando se define un proceso de negocio en BPEL, esencialmente se está definiendo un nuevo servicio Web que es una composición de servicios existentes. La interfaz de la nueva composición BPEL utiliza un conjunto de tipos de puertos a través de los cuales provee operaciones como cualquier otro servicio Web. Para invocar el proceso de negocio descrito en BPEL, se tiene que invocar al servicio Web resultado de la composición. En la figura 2.35 se puede ver el esquema del proceso.

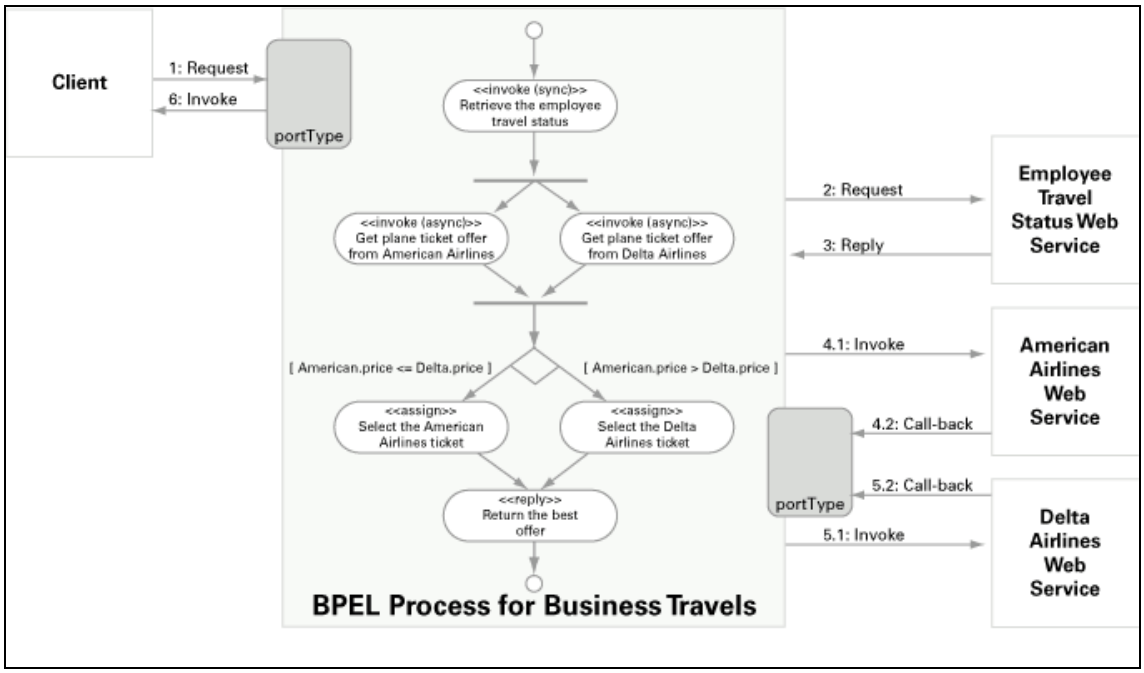

Figura 2.32. Ejemplo de un proceso BPEL

Arquitectura de un servidor BPEL

Para que se puedan ejecutar procesos de negocio BPEL se necesita un entorno de ejecución para que se puedan ejecutar los procesos; actualmente en el mercado existen distintos servidores que se integran con los servidores de aplicaciones, permitiendo desplegar, monitorizar y gestionar los distintos procesos.

En la figura 2.33 se puede ver la arquitectura del servidor BPEL de Oracle, (*Oracle BPEL Process Manager*) [Oracle, 2005]. Está desarrollado en Java y es compatible con cualquier servidor de aplicaciones compatible J2EE como por ejemplo Oracle Application Server OC4J, JBoss o Bea Weblogic.

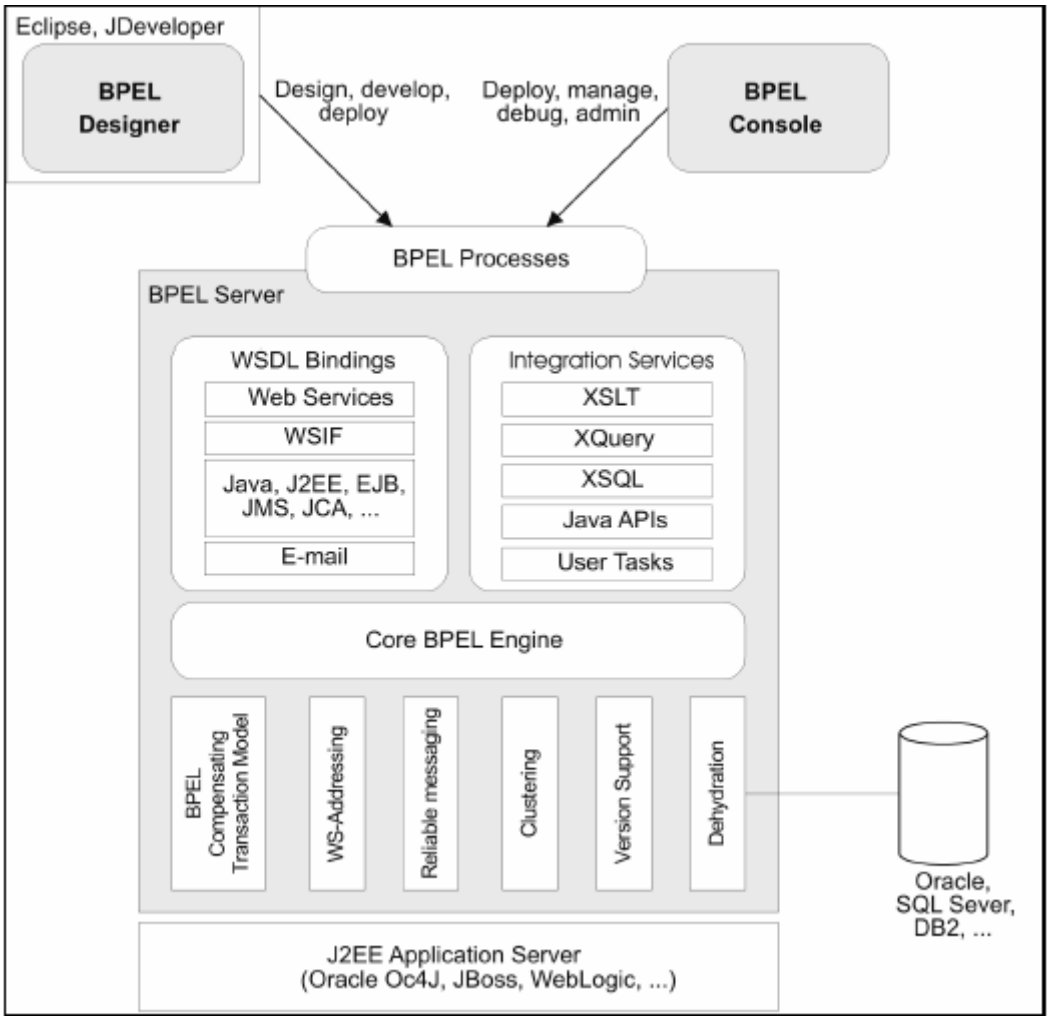

Figura 2.33. Arquitectura servidor BPEL

Como se puede ver en la figura anterior se compone de 4 partes:

- Diseñador BPEL: Para modelar los procesos BPEL.
- Servidor BPEL: Para ejecutar los procesos BPEL.
- Consola BPEL: Para monitorizar los procesos BPEL.
- Base de datos: Para guardar los procesos BPEL.

2.3.8 Seguridad y servicios Web

Con los servicios Web, muchas de las aplicaciones internas a una organización son expuestas a Internet. La necesidad de garantizar la integridad, la confidencialidad y la autenticidad de los datos que fluyen a través de la Web se ha convertido en un requisito esencial. Pero hay muchas dificultades a la hora de manejar datos con estructuras jerárquicas y subgrupos de datos con diferentes requisitos en lo que se refiere a confidencialidad, derechos de acceso o integridad.

Tradicionalmente, los protocolos *Secure Sockets Layer* (SSL) [Freier,1996], *Transport Layer Security* (TLS) [Dierks, Allen, 1999], Redes Privadas Virtuales (VPNs) [VPN, 2006] o el Internet Protocol Security (IPSec) [IPSec, 2006] son las formas más comunes de aportar seguridad.

Sin embargo en la utilización de los servicios Web, si se crea una comunicación punto a punto, se tiene que crear un túnel seguro a través del cual los datos puedan pasar. Con el protocolo *Secure Multipurpose Internet Mail Exchange* (S/MIME) [Ramsdell, 2004], los datos pueden enviarse firmados digitalmente y encriptados. Aunque las tecnologías tradicionales pueden ser usadas; éstas no son suficientes para un entorno de servicios Web.

La seguridad de los servicios Web se puede dividir en dos niveles: el nivel de transporte y el nivel de aplicación.

2.3.8.1 Seguridad en el nivel de transporte

La seguridad en el nivel de transporte consiste en la utilización de características predefinidas en la capa de transporte sobre el protocolo que están utilizando los servicios Web. Esto es posible porque el mensaje SOAP es normalmente encapsulado directamente con un mecanismo de paquetes que aporta la capa de transporte.

Es importante recordar que la seguridad en este nivel es una seguridad punto a punto, esto implica que la comunicación es directa sin intermediarios. Los *routers* en Internet no se pueden considerar intermediarios para las aplicaciones que utilizan servicios Web.

Desde la perspectiva de los servicios Web, la seguridad punto a punto es suficiente cuando el consumidor del servicio le llama directamente, sin tener intermediarios SOAP entre los dos.

La seguridad en este nivel se basa en aportar seguridad al tráfico IP mediante tecnologías como IPSec (*Internet Protocol Security*). IPSec utiliza criptografía e implementa autenticación IP, privacidad e integridad de los datos.

2.3.8.2 Seguridad en el nivel de aplicación

La seguridad en el nivel de transporte no impacta sobre el mensaje SOAP, pero requiere configuración del servidor y del cliente.

Cuando se aplica seguridad en el nivel de aplicación, hay que modificar el mensaje SOAP. El mensaje modificado es enviado por cualquier protocolo, y no se necesita

realizar ninguna configuración sobre el servidor o sobre el software del cliente. En cambio, la seguridad a nivel de aplicación requiere que el servidor y el cliente admitan ambos la misma seguridad para poder intercambiar mensajes.

2.3.8.3 Especificaciones Estándar de Seguridad de Servicios Web

El W3C Consortium [W3C, 2006] es el encargado del desarrollo de las principales tecnologías estándar para la seguridad de los documentos XML y para la extensión de los mensajes SOAP: XML Encryption [W3C, 2001c], XML Digital Signature [W3C, 2002], XML Key Management System [W3C, 2001c].

XML Encryption y XML Signature están preparados para manejar situaciones en las que partes de un mismo documento necesitan un tratamiento diferente, como ocurre en documentos con diferentes secciones cuyo contenido puede ser visto por unos usuarios pero no por otros. En estos casos, la encriptación juega un papel muy importante ya que es lo que va a confirmar la integridad del texto. Por otro lado, las firmas digitales permiten la autenticación del remitente. Otro problema añadido surge cuando diferentes personas firman digitalmente un mismo documento XML o cuando es necesario hacerlo conjuntamente codificando ciertas partes de ese documento.

A continuación se describe cada una de estas tecnologías:

XML Encryption: Es un lenguaje cuya función principal es asegurar la confidencialidad de partes de documentos XML, a través de la encriptación parcial del documento transportado, o de todo el documento. XML Encryption se puede aplicar a cualquier recurso Web, incluyendo contenido que no es XML.

Ejemplo de utilización XML Encryption

El siguiente fragmento de código muestra información sin cifrar de un cliente ficticio y su tarjeta de crédito.

```
<?xml version='1.0'?>
<Metodopago xmlns='http://ejemplo.org/pago'>
  <Nombre>Cliente Ficticio</Nombre>
  <TarjetaCredito Limite='5.000' Moneda='EU'>
    <Numero>0000 0000 0000</Numero>
    <Issuer>Ejemplo de Banco</Issuer>
    <Caducidad>10/06</Caducidad>
  </TajetaCredito>
</Metodopago>
```

El fragmento de código anterior muestra que Cliente Ficticio utiliza una tarjeta de crédito con un límite de 5.000 Euros para realizar su compra.

El número de la tarjeta de crédito es una información que debería ser confidencial por motivos de seguridad, por lo que es importante que esté cifrada.

La forma de realizar el encriptado de esa parte con XML Encryption sería la siguiente:

```
<?xml version='1.0'?>
 <Metodopago xmlns="http://ejemplo.org/pago">
  <Nombre>Cliente Ficticio</Nombre>
  <DatosEncriptados xmlns="http://www.w3.org/2001/04/xmlenc#"
    Tipo="http://www.w3.org/2001/04/xmlenc#Element">
    <DatosClave>
      <DatosClave>A23B45C56</DatosClave>
    </DatosClave>
  </DatosEncriptados>
</Metodopago>
```

Al cifrar el elemento completo de tarjeta de crédito, el elemento en sí mismo se oculta por lo que no es posible saber si el cliente está usando un tarjeta de crédito o simplemente dinero en metálico. El elemento *<DatosClave>* contiene el número encriptado del elemento *TarjetaCredito*.

También es posible cifrar todo el documento.

XML Signature: Asegura la integridad de partes de documentos XML transportados. También proporciona la autenticación de mensajes y/o servicios de autenticación de firma para datos de cualquier tipo, tanto si se encuentra en el XML que incluye la firma o en cualquier otra parte. Puede aplicarse a cualquier contenido digital (objeto de datos), incluyendo XML. Lo que hace principalmente XML Signature es asociar claves con los datos de consulta. XML Signature representa un sistema que, a través de una firma digital, permite ofrecer autenticidad de los datos. Con la firma digital se confirma la identidad del emisor, la autenticidad del mensaje y su integridad, sin olvidar que los mensajes no serán repudiados.

Ejemplo de XML Signature

```
<Signature Id="EjemploXMLSignature"
 xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
     Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
     Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference
```

```
          URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
          <DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
    <KeyInfo>
        <KeyValue>
            <DSAKeyValue>
              <p>...</p><q>...</q><g>...</g><y>...</y>
            </DSAKeyValue>
        </KeyValue>
    </KeyInfo>
</Signature>
```

El elemento <*Signatura*> encapsula la firma digital. Contiene tres sub-elementos: <*SignedInfo*>, <*SignatureValue*> y <*KeyInfo*>. El elemento <*SignedInfo*> contiene información sobre qué es lo que se firma y cómo se firma, es decir, contiene la información necesaria para crear y validar la firma. Este elemento contiene dos algoritmos. Por un lado, está el <*CanonicalizationMethod*> que es el algoritmo de transformación de <*SignedInfo*> antes de realizar la firma digital. Por otro lado, estaría el método de firma (<*SignatureMethod*>), que sería el algoritmo utilizado para calcular el valor de la firma digital. También se incluye en el elemento <*SignedInfo*> las referencias a los objetos que se van a firmar (<*Reference*>), que incluye además <*DigestMethod*> y <*DigestValue*>. La validación de una firma requiere dos procesos que son la validación de la firma y la validación de los resultados de las referencias.

El elemento <*CanonicalizationMethod*> es el encargado de indicar el algoritmo para canonizar el elemento <*SignedInfo*>, que tendrá lugar durante la creación de la firma. El <*SignatureMethod*>, es el encargado de indicar el algoritmo para general la firma a partir de la canonización de <*SignedInfo*>. El resultado obtenido si indicará en el elemento <*SignatureValue*>.

Cada elemento <*Reference*>, incluye una referencia al objeto que se firmará. Al mismo tiempo, incluye el resultado de <*DigestValue*>**,** que es el valor resultante.

El elemento <*SignatureValue*> contiene el resultado de la firma digital que se ha aplicado sobre el elemento <*SignedInfo*>. El resultado de esta firma está codificado y contiene un atributo que es único con el que se identificará la firma en procesos posteriores de validación.

El elemento <*KeyInfo*>, es opcional e indica la clave que ha de utilizarse para validar la firma. El elemento <*KeyValue*>, especifica la clave para validar la firma digital.

Por tanto, y a modo de resumen, se puede decir que el elemento <*SignedInfo*> contiene lo que se firma. Por otro lado, el elemento <*SignatureValue*>, contiene la firma, es decir, contiene el elemento <*SignedInfo*> en forma canonizada, resumida y encriptada con la clave pública del firmante; y por último, <*KeyInfo*>, contiene el certificado de la clave pública del firmante.

XML Key Management: Es un protocolo para distribuir y registrar claves públicas. Lo que hace es ocultar la parte compleja que surge con PKI (Infraestructura de Clave Pública). Está compuesto de dos partes que son: el registro de la clave pública (X-KRSS) y la información de clave pública (X-KISS).

2.4 SERVICIOS WEB Y E-LEARNING

Una vez presentados los conceptos más importantes sobre e-learning, servicios Web y SOA, veremos que relación han tenido estas tecnologías y como su unión puede ayudarnos en la definición de la arquitectura objeto de esta tesis.

Primero se presentará una serie de trabajos (generalmente ponencias de congresos) que relacionan los servicios Web y el e-learning y a continuación nos centraremos en las especificaciones de IMS relacionadas con el tema en cuestión.

2.4.1 Trabajos relacionados

Hay una serie de trabajos que presentan ideas sobre lo que los servicios Web pueden aportar al mundo de la teleformación. De estos trabajos hemos seleccionados un conjunto significativo que nos han servido de base para la definición de la arquitectura.

En el trabajo de Liu et al. [2003] titulado “Una arquitectura implementable para un sistema de e-learning” los autores repasan los estándares y propuestas de arquitecturas de sistemas de e-learning y a continuación presentan una arquitectura funcional y orientada a servicios para la construcción de sistemas de teleformación interoperables. Se presenta también como se integran los servicios Web en las aplicaciones de e-learning y como se puede usar J2EE para la construcción de sus componentes.

Como se puede apreciar en la figura 2.34 el modelo funcional que propone Liu está muy relacionado con el de SCORM, en el que se divide la funcionalidad del sistema entre la asignada al LMS y al LCMS, existiendo intercambio de información entre ambos.

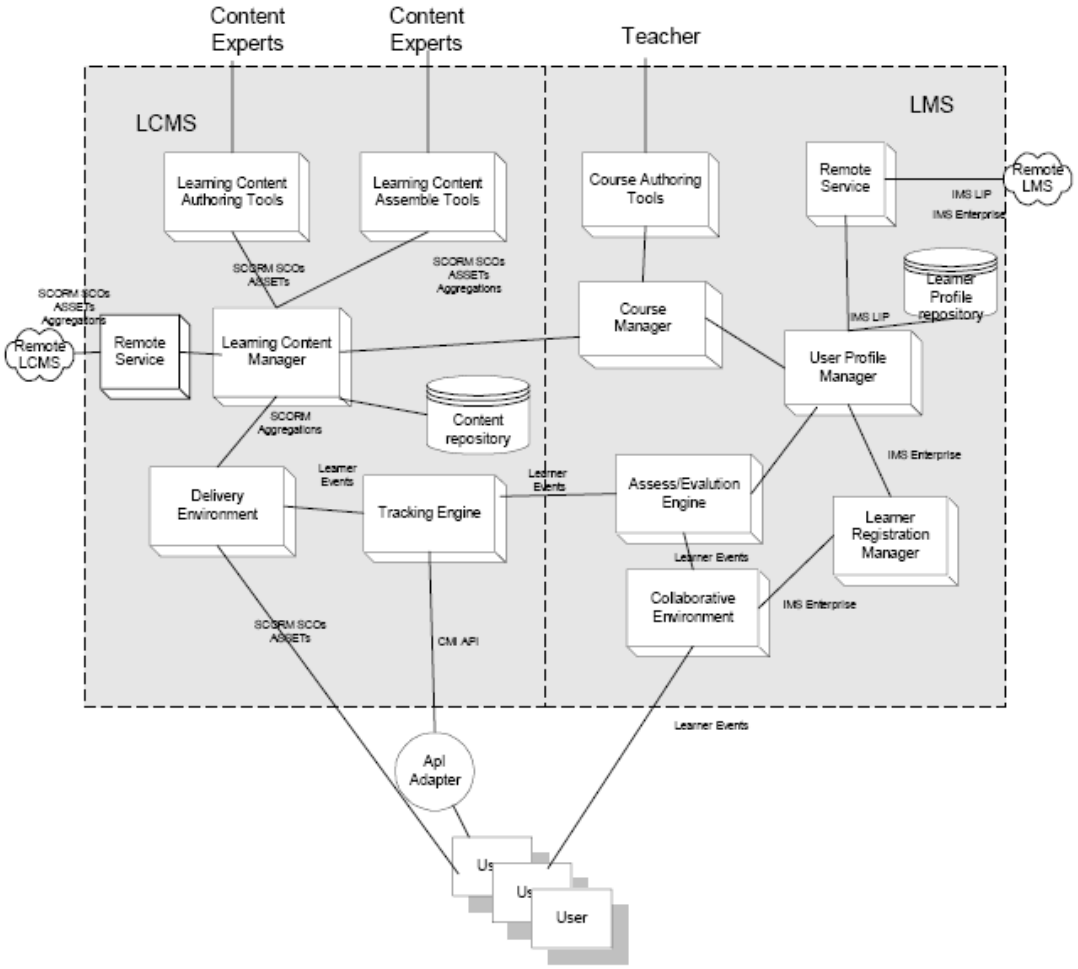

Figura 2.34. Modelo funcional de un sistema de e-learning

Para la implementación de la arquitectura, los autores se basan en los servicios Web por varias razones: la utilización de XML para el intercambio de datos entre diferentes sistemas de e-learning, por la independencia de los servicios Web de la plataforma y el lenguaje de programación y por sus posibilidades de interoperabilidad y extensibilidad y su integración con Internet. La arquitectura, mostrada en la figura 2.35 define como diferentes sistemas de e-learning intercambian mensajes mediante agentes que proporcionan los servicios Web.

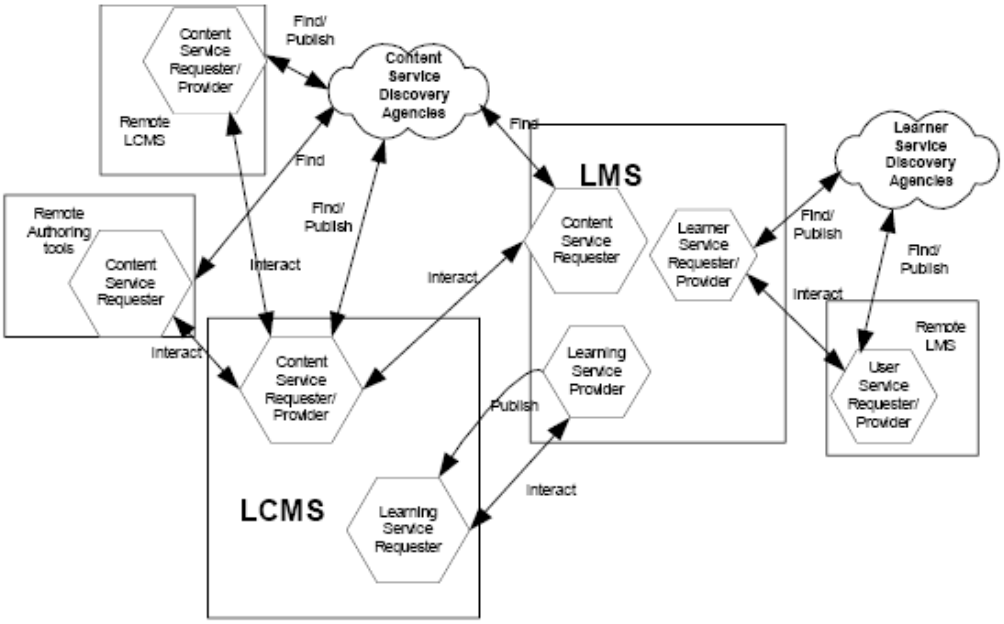

Figura 2.35. Arquitectura orientada a servicios de un sistema de e-learning

El segundo trabajo relacionado es el titulado “Framework orientado a servicios Web para sistemas e-learning dinámicos” de Xu et al. [2003]. El objetivo de este trabajo es proponer un framework basado en servicios Web que proporciona la integración de los componentes y aplicaciones que forman un sistema de e-learning facilitando su interacción. Usando este *framework*, los proveedores de servicios de formación podrán publicar sus objetos de aprendizaje o aplicaciones de forma universal permitiendo la interoperabilidad y la reusabilidad.

Como se aprecia en la figura 2.39, el *framework* se basa en una arquitectura orientada a servicios donde se distingue entre los proveedores de servicios de formación y los consumidores de éstos. Para definir los contenidos docentes y hacerlos reutilizables se basa en SCORM Content Model.

Todos los objetos de aprendizaje, contenidos y aplicaciones se construyen como servicios Web, se describen mediante su WSDL y se publicarán en un registro UDDI.

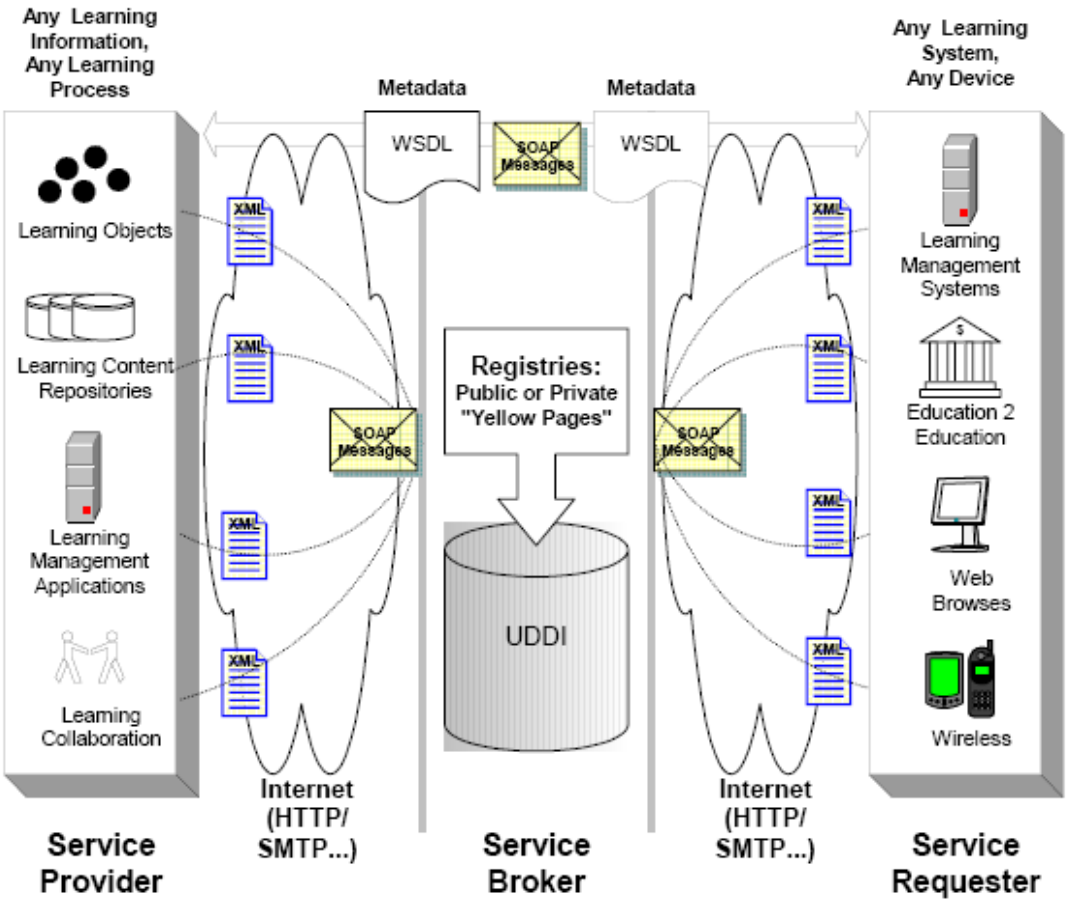

Figura 2.36. Framework orientado a servicios Web para sistemas e-learning dinámicos

El siguiente trabajo relacionado se titula "¿Cómo el paradigma de los servicios Web puede mejorar el e-learning?" de Rodríguez et al. [2003]. Este trabajo es muy similar al anterior en el sentido en el que propone un *framework* orientado a servicios para la construcción de una plataforma intermediaria que comunica a los proveedores de servicios con sus consumidores.

Esta plataforma proporciona una forma común de comunicación y localización de servicios de formación. Los desarrolladores de plataformas de e-learning pueden reutilizar los servicios de este intermediario para construir nuevos sistemas. Se puede apreciar esta idea en la figura 2.37.

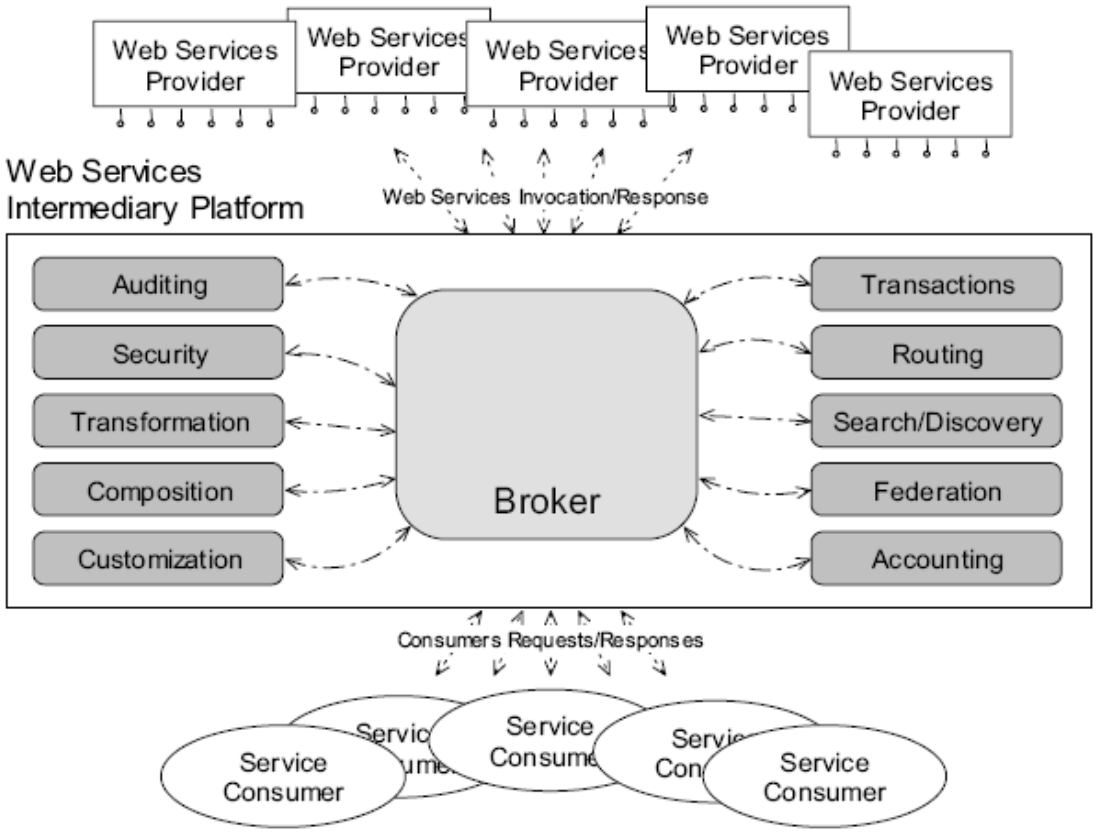

Figura 2.37. Plataforma intermediaria para sistemas e-learning

El último trabajo relacionado se titula “La interoperabilidad de los repositorios de objetos de aprendizaje y los servicios”, de Hatala et al. [2004]. Este trabajo se aleja de los explicados anteriormente en el sentido en el que los anteriores trataban la relación de los servicios Web y los sistemas de e-learning en general; éste se aproxima más a la relación de los servios Web y los repositorios, lo cual está más relacionado con el objetivo de la tesis.

El trabajo se centra en el problema de la interoperabilidad en la construcción de una red de repositorios. Se basa en el proyecto “eduSource” que trata de construir una red de repositorios de objetos de aprendizaje en Canadá con una serie de servicios de formación.

Realiza un estudio muy interesante de los principales esfuerzos en materia de interoperabilidad en el mundo del e-learning centrándose en el estudio de IMS Digital Repository Interoperability (DRI) [IMS, 2003a] en el que se basa para la definición de eduSource. Esta especificación también se utilizara en esta tesis.

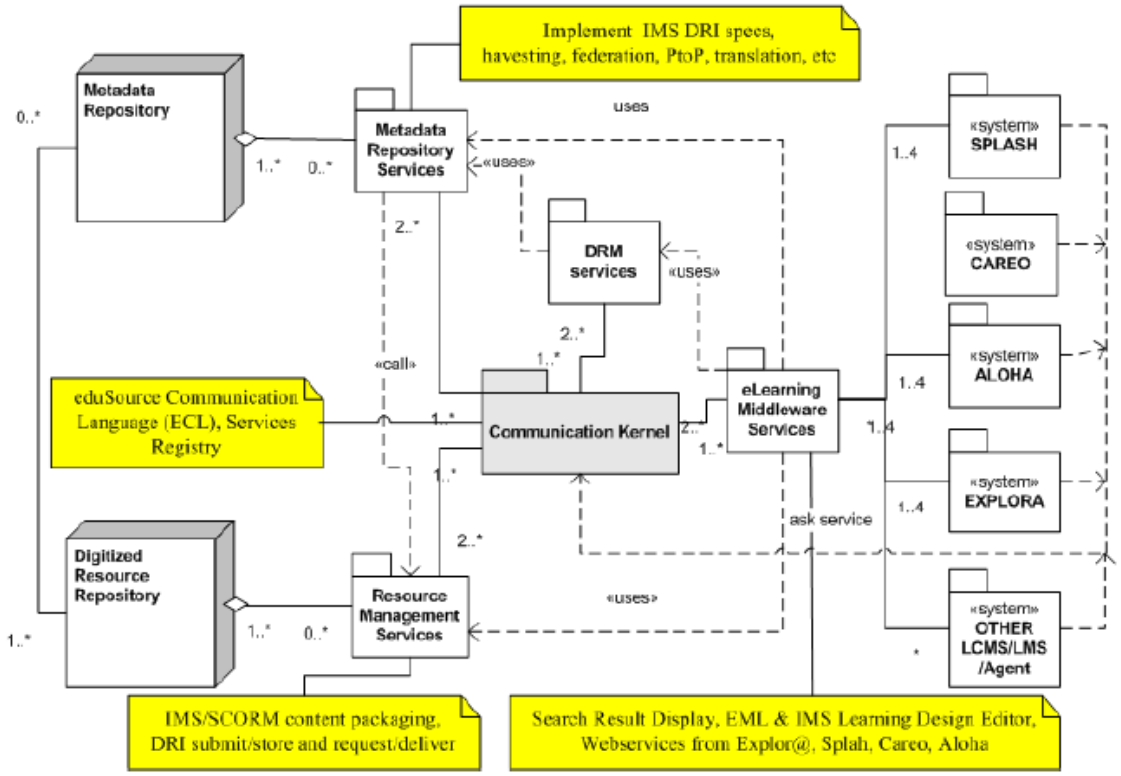

Figura 2.38. Arquitectura funcional de eduSource

Como se puede apreciar en la figura 2.38, la arquitectura funcional de eduSource está basada en una serie de servicios que se comunican mediante mensajes SOAP, estos servicios pueden estar publicados en un registro UDDI. eduSource está formado por una serie de redes heterogéneas consistente en repositorios de instituciones existentes y futuras, redes P2P, repositorios individuales y aplicaciones. Para soportar la comunicación de estos servicios se utiliza lo que los autores llaman ECL (*eduSource Communication Layer*), que está implementado siguiendo la especificación DRI de IMS y que permitirá soportar nuevos servicios en el futuro.

2.4.2 IMS Abstract Framework

Es interesante explicar de forma resumida el IMS Abstract Framework [IMS, 2003b], ya que es el contexto en el que se basa IMS para realizar las especificaciones de todas sus tecnologías de e-learning. Este framework no se ocupa de definir la arquitectura IMS, más bien es un mecanismo para definir un conjunto de servicios para los cuales se definen sus especificaciones de interoperabilidad. Como ya se indicó en un apartado anterior, en los casos donde IMS no puede producir una especificación, trata de adoptar o recomendar una especificación conveniente de otra organización.

El framework de IMS tiene como particularidades:

- Es una representación abstracta del conjunto de servicios que se utilizan para construir un sistema e-learning en su sentido más amplio.
- Está centrado en el soporte de los sistemas de formación distribuidos.
- Es un *framework* que cubre todo el rango de posibles arquitecturas e-learning que se podrían construir a partir de un conjunto de servicios definidos.

Y como principios:

- **Interoperabilidad:** Las especificaciones están basadas en el intercambio de información entre sistemas pero sin tomar decisiones acerca de cómo se tratan los datos en la comunicación.
- **Orientada a servicio:** La interacción entre sistemas se define en términos de los servicios expuestos entre ellos para establecer esa colaboración. Esta colaboración puede realizarse de múltiples formas desde la basada en "*peer-to-peer*" a técnicas cliente - servidor.
- **Basada en componentes:** El conjunto de servicios será ofrecido por componentes que pueden ser combinados para formar un servicio particular. Un componente puede proporcionar todos o una parte de los servicios.
- **Por capas:** El conjunto total de servicios requeridos para hacer un sistema e-learning será modelado como un conjunto de capas y cada capa proporcionará un conjunto claro de servicios definidos.
- **Comportamientos y Modelos de Datos:** Un servicio será definido en términos de sus comportamientos y su modelo de datos. Los comportamientos causarán cambios en el estado del modelo de datos y el estado del modelo de datos sólo cambiará como consecuencia de un comportamiento claramente definido.
- **Múltiples ligaduras:** El modelo de información para una especificación (comportamiento y datos) puede ser apoyado por múltiples ligaduras como por ejemplo Java, XML, servicios Web, etc.

Antes de empezar a describir este *framework*, es interesante introducir la visión que IMS hace de la arquitectura de los sistemas de e-learning en general. Da dos visiones distintas de estos sistemas, una sería la arquitectura lógica y otra sería la física.

La arquitectura lógica está basada en un modelo multicapa o multinivel como el que utilizaremos para describir la que se propone en esta tesis. Este modelo se puede ver en la figura 2.39 y consiste en las siguientes capas:

- Usuarios: Representa al conjunto de usuarios de un sistema de e-learning, como los estudiantes, administradores, profesores, etc. Los usuarios tienen acceso al sistema a través de "agentes de usuario".

- Agentes de usuario: Los agentes que proporcionan los servicios a los usuarios. Los agentes de usuario son los sistemas de e-learning que proporcionan la interfaz para tener acceso y actuar con todos los servicios de e-learning. Los agentes de usuario incluyen sistemas como LMS, LCMS, herramientas de autor y contenido. Los Agentes de Usuario son construidos usando la colección de servicios proporcionados por la capa de servicios.
- Herramientas: Permiten el acceso a los diferentes servicios de una forma conveniente y fácil de usar. Esto incluye la evaluación, tutorización, simulación, etc.
- Servicios de educación. Incluyen todos los servicios y los componentes que tienen la funcionalidad específica de e-learning.
- Servicios de soporte: Servicios comunes tales como la autenticación, el descubrimiento de recursos, etc.
- Repositorios digitales: Para el almacenamiento del material docente.
- Infraestructura de comunicaciones: La interconexión básica y los servicios de transporte de datos que entregan la información punto a punto.

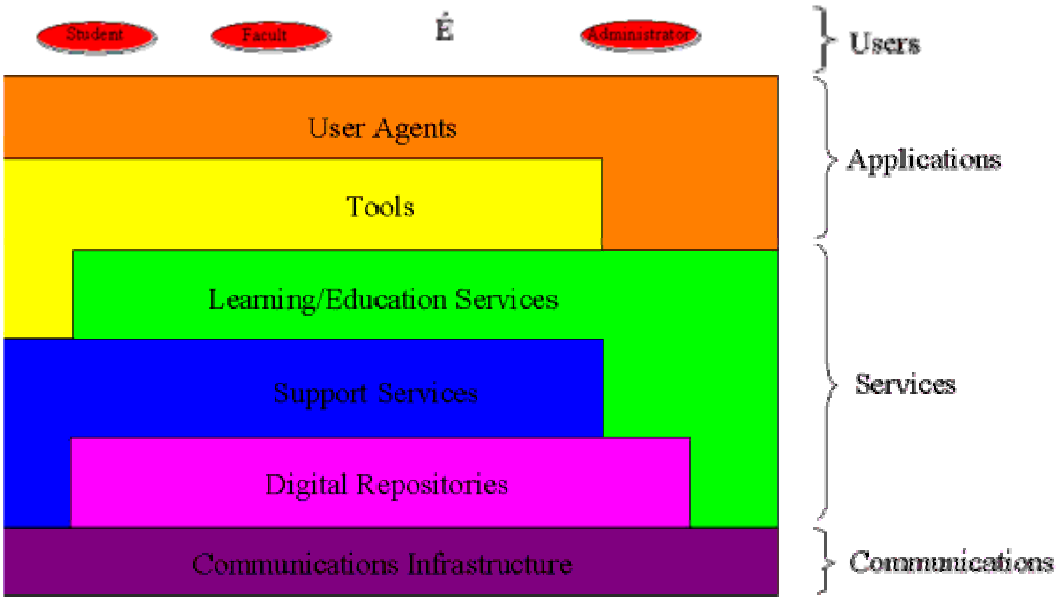

Figura 2.39. Arquitectura lógica de los sistemas de e-learning

La arquitectura física es la que se muestra en la figura 2.40, está compuesta por las siguientes estructuras:

- La red principal: Red primaria que interconecta los sistemas informáticos principales. Estaría representada por Internet. Es una parte de la Infraestructura de Comunicaciones en el modelo lógico.

- La red de acceso: La red que une los dispositivos de entrega para la red principal. Ejemplos típicos son redes de cable, redes inalámbricas, etc. Es una parte de la Infraestructura de Comunicaciones en el modelo lógico.
- Repositorios digitales federados: La serie de los recursos digitales que están disponibles en una variedad de repositorios digitales, bases de datos, servidores Web, etc. Representa la capa de Repositorios Digitales en el modelo lógico.
- Motores de servicios de entrega: Los sistemas responsables de la provisión de servicios de e-learning. Corresponde a los servicios de educación y de soporte en el modelo lógico.
- Dispositivos de entrega: Los dispositivos encargados de entregar el material docente al usuario. Corresponde a las herramientas y a los agentes de usuario en el modelo lógico.

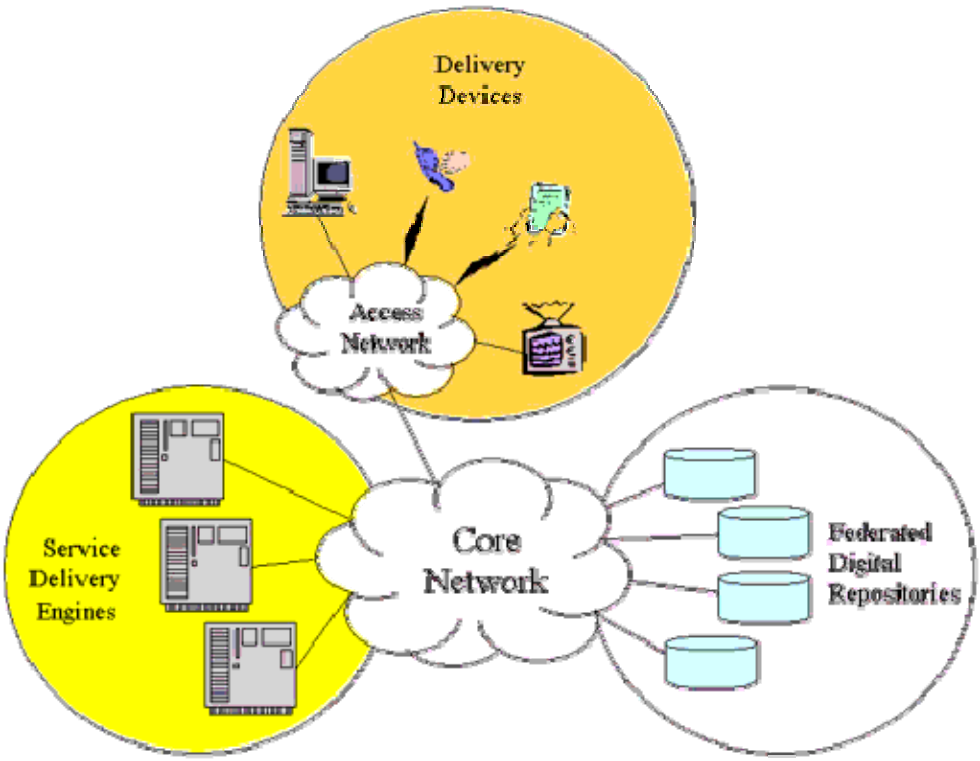

Figura 2.40. Modelo físico de los sistemas de e-learning

Una vez vista la arquitectura general que representa a los sistemas de aprendizaje, pasaremos a describir el *framework* de IMS, que ha sido diseñado para que pueda ser usado en una amplia gama de arquitecturas. Este *framework* se representa como un modelo en capas, como se aprecia en la figura 2.41; IMS justifica la adopción de este tipo de modelo debido a la gran proliferación de su uso en la definición de arquitecturas e-learning.

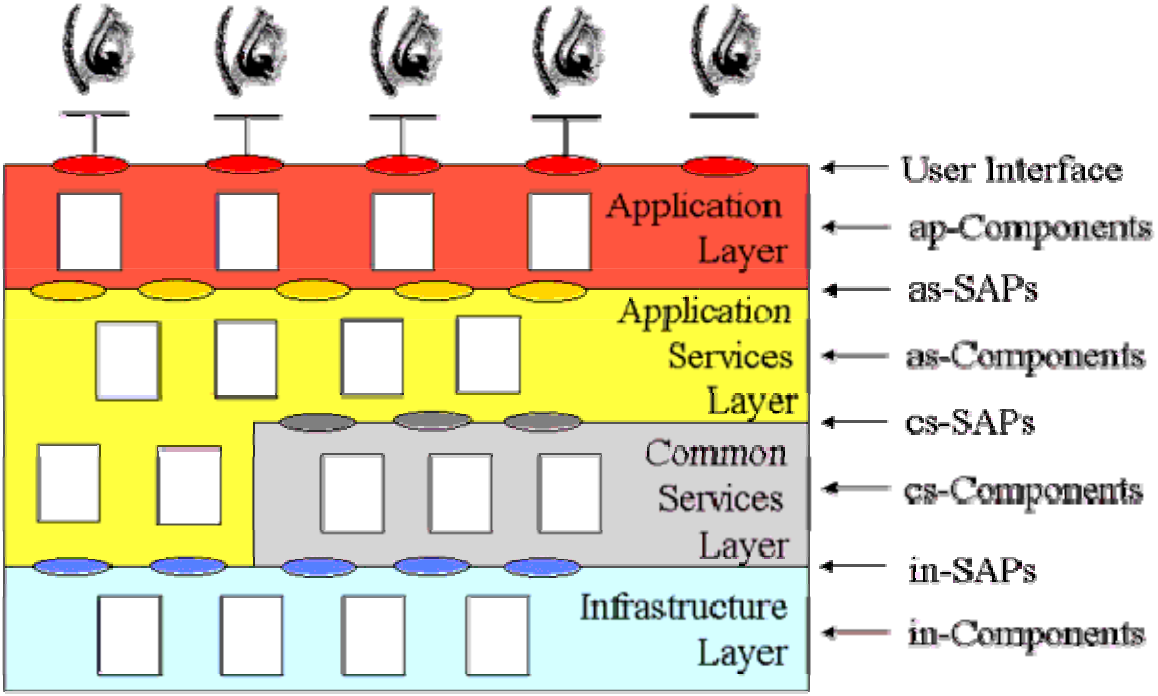

Figura 2.41. Modelo en capas del IMS Abstract Framework

Las principales características de cada una de las capas que definen este framework son las siguientes:

- **Capa de aplicación:** Sistemas, herramientas y aplicaciones que exponen los servicios de aplicación al usuario final proporcionando un conjunto particular de funcionalidades de e-learning.
- **Capa de servicios de aplicación:** Conjunto de entidades que proporcionan los servicios específicos de funcionalidades de e-learning. IMS se centra principalmente en la especificación de estos servicios.
- **Capa de servicios comunes:** Conjunto de entidades que proporcionan servicios generales que serán usados por los servicios de aplicación. Por ejemplo autenticación.
- **Capa de infraestructura:** Servicios encargados del intercambio de información (mensajes, transacciones, etc.).
- **Puntos de acceso a los servicios:** Los puntos de acceso o interfaces correspondientes a cada servicio.
- **Entidades:** Los procesos que se utilizan para representar un servicio particular. La implementación de una entidad con sus puntos de acceso de servicio se denomina componente y su representación abstracta se llama clase.

El acceso a un servicio se realiza mediante su Punto de Acceso al Servicio (denominado en la figura SAP: *Service Access Point*) apropiado. Cada servicio tiene un

SAP único. Un componente puede soportar uno o varios SAP (en una representación orientada a objetos un SAP podría ser soportada por una o varias operaciones donde la clase es la definición del servicio).

Uno de los principios de diseño para el IMS Abstarct Framework es la adopción de la orientación a servicio como forma de describir la funcionalidad de e-learning. El servicio se oculta detrás de un SAP y sólo se puede acceder usando ese SAP. La figura 2.45 muestra una representación esquemática de un servicio.

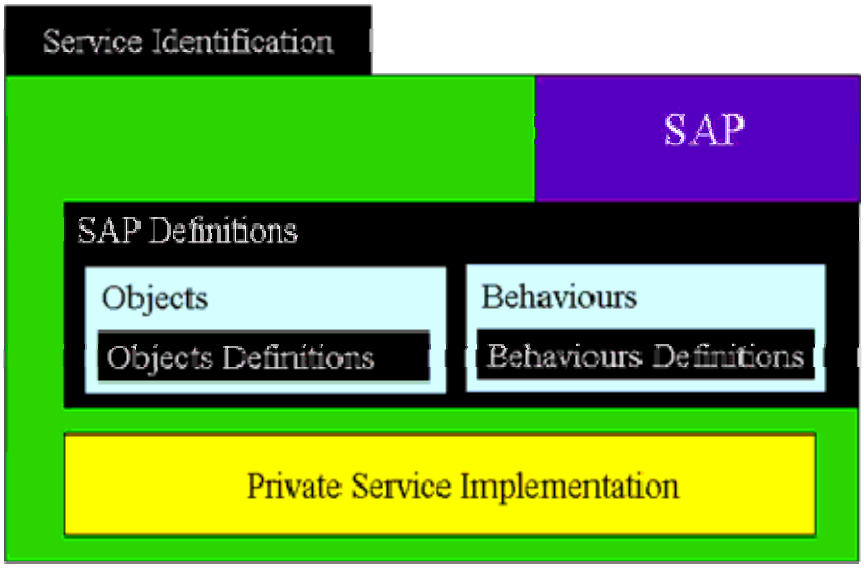

Figura 2.42. Descripción de un servicio en IMS AF

- El servicio tiene un punto de acceso de servicio claramente definido. Cada servicio tiene sólo un SAP. El SAP se define en términos de sus objetos constituyentes y sus comportamientos.
- El SAP puede consistir en uno o varios objetos y cada objeto va a tener, en general, más de una operación. Cada objeto es definido usando una definición de clase y consiste en un conjunto de atributos y operaciones. La operación describe como el estado de los atributos puede ser cambiado. El juego de comportamientos permitidos para cada clase también debe ser definido.
- La implementación de Servicio Privado está fuera del alcance del IMS AF. La única exigencia es que debe proporcionar todas las características apropiadas del servicio y nada más.

En la capa de infraestructura lo más importante es la interoperabilidad en el intercambio y manipulación de datos. En esta capa, las especificaciones de IMS se utilizan para definir las estructuras de datos y los comportamientos permitidos para el intercambio de esas estructuras de datos entre los componentes que forman la capa. Esta interoperabilidad se define en términos de intercambio de documentos XML. El

framework también describe los mecanismos de transporte que se pueden utilizar para intercambiar los documentos XML.

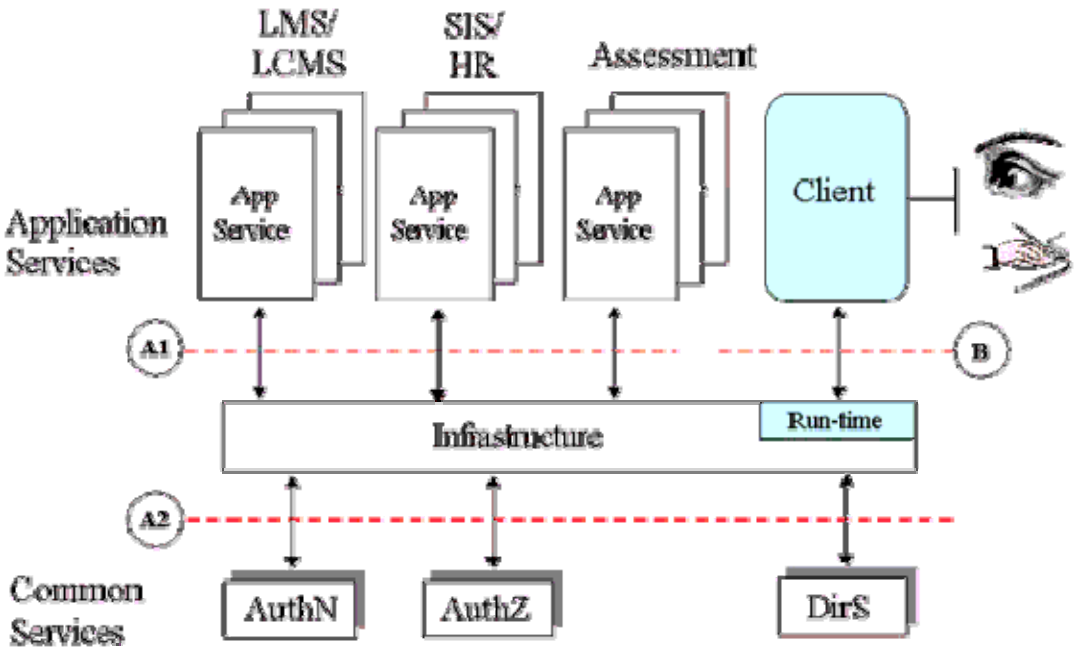

Figura 2.43. Interacción de los servicios en IMS AF

En una implementación distribuida de este *framework*, como en un ambiente de servicios Web, la interacción entre los servicios sería como la mostrada en la figura 2.46. En este framework de interacción hay tres categorías de interfaces que deben ser apoyadas por la capa de Infraestructura, como son:

- El interfaz de Servicios de Aplicación (A1) - este interfaz es usado para proporcionar la interoperabilidad entre servicios comunes de aplicación; por ejemplo, entre sistemas de empresa-a-empresa (B2B).
- El interfaz de Servicios Común (A2) - este interfaz es usado para proveer la interoperabilidad del conjunto de servicios comunes poniéndolos a disposición de los servicios específicos de aplicación; por ejemplo, la autenticación y la autorización
- El interfaz de ejecución (B) - este interfaz es usado para interconectar la ejecución del cliente con el proveedor de servicios remoto.

Hay dos tipos de comportamiento de interacción que tienen que ser especificados para asegurar la interoperabilidad:

- El paso de Mensajes - donde la información es intercambiada entre sistemas que usan alguna forma de paso de mensajes. El contenido y la secuencia de los mensajes definen el comportamiento esperado.

- Ejecución - donde los sistemas finales tienen que usar algún algoritmo predefinido sobre la información que les llega. Los datos determinarán los resultados de los algoritmos pero el comportamiento estará bien definido para todos los resultados posibles.

2.4.3 IMS Enterprise Services

La especificación IMS Enterprise Services [IMS, 2004] es la definición de cómo los sistemas de aprendizaje son capaces de gestionar el intercambio de información concerniente a las personas, grupos y componentes del contexto de aprendizaje. Esta especificación se basa en los siguientes conceptos:

a) *Interoperability*: Intercambio de información entre diferentes sistemas.
b) *Service-oriented*: Intercambio de información desde el punto de vista de los servicios que son suministrados gracias a la colaboración de los sistemas. Esto adquiere la forma de Servicios de Gestión de: personal, grupos y miembros.
c) *Component-based*: Es el conjunto de servicios que serán suministrados para conformar un rango de servicios. Los Servicios de gestión de personal, grupos y miembros pueden combinarse para proporcionar otros servicios a los cuales se añadirá más procesos en futuras versiones.
d) *Layering*: Define el comportamiento y modelo de datos entre las diferentes capas de los Servicios.
e) *Multiple Bindings*: El modelo de información del "*Enterprise Services*" se define empleando el lenguaje UML, lo cual permite contrastar de una manera fiable el modelo de información en una amplia variedad de marcos (entornos). Los marcos de mayor importancia son los lenguajes de descripción de servicios Web (WSDL)
f) *Adoption*: El "*Enterprise Services*" se basa en las especificaciones originales del Enterprise. Aún existiendo múltiples y significativos cambios entre ellos, la base del modelo de datos ha mantenido las estructuras de las personas, grupos o miembros, de la base original.

En lo referente al modelo abstracto de esta especificación, se puede indicar que está formado por una estructura como la que se muestra a través de la figura 2.44.

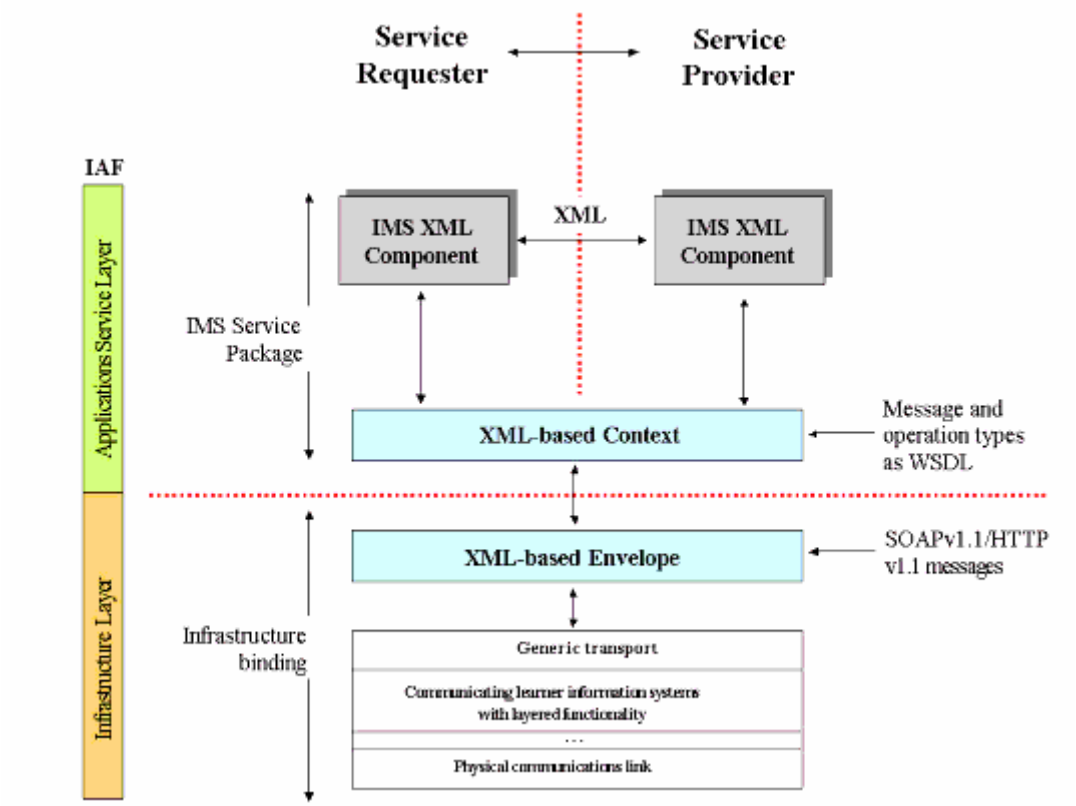

Figura 2.44. Modelo de la Arquitectura Enterpise Services

Como se muestra en la figura 2.44, el alcance del modelo *Enterprise Services* quedaría representado por la parte superior del modelo a partir de la línea de puntos horizontal. Antes de comentar las diferentes partes de la figura, es importante indicar que con este gráfico IMS no pretende hacer una representación de cómo debería estar construido un sistema empresarial. La especificación IMS Enterprise Services contiene un modelo abstracto basado en mensajería, encargado de ser la base para el intercambio de información entre los distintos sistemas. La comunicación entre la capa de infraestructura y la capa de servicios de aplicación se realizaría a través de un sistema de mensajería basado en SOAP, llamado IMS Service Package.

El intercambio de información entre el sistema que realiza la petición y el que proporciona la respuesta, se realizaría, a través de un intercambio de información tipada utilizando para ello XML. Será necesario simplemente indicar los datos a intercambiar entre los dos sistemas, las operaciones que pueden ser invocadas y la secuencia de operaciones permitidas.

IMS también da ciertas especificaciones que deberían cumplir tanto los sistemas invocadores de peticiones, como los que van a procesarlas para retornar la información solicitada.

Solicitadores de servicios.

- Deberían construir los mensajes de petición apropiados según los parámetros especificados en la definición del enlazado creada sobre el modelo de información.

- Debería ser capaz de procesar los mensajes de respuesta correspondientes, según fueron definidos en el enlazado del modelo de información. El documento de enlazado es el responsable de detallar cómo un consumidor de servicios debe establecer la secuencia de mensajes de petición/respuesta. Dicha especificación asume que todo servicio garantizará que nunca se producirán mensajes de respuesta duplicados.

- Debería devolver un código de resultado de la solicitud así como los comentarios necesarios asociados a la misma.

Proveedores de servicios.

- Debería ser capaz de procesar el conjunto de mensajes de petición que pueden ser recibidos, según han sido definidos en la definición del enlazado creada sobre el modelo de información. Si se recibieran un conjunto de datos inválidos en el mensaje de petición, esto no debería suponer un problema para el proveedor de servicios, manteniéndose la integridad de los datos posiblemente modificados antes de producirse el error; es decir, deberían ser servicios transaccionales.

- Debería construir los mensajes de respuesta apropiados según han sido definidos en el enlazado. En este documento de enlazado se detallará cómo un servicio de petición debe mantener la relación atómica de la secuencia de mensajes de petición/respuesta.

- Debería mantener la persistencia de los objetos de datos desde el momento de su creación hasta que son suprimidos.

- Debería implementar los procesos necesarios para darle respuesta a la solicitud recibida.

2.4.4 IMS General Web Services

La base de los Servicios Web Generales [IMS, 2005] promueve la interoperabilidad de las especificaciones de los servicios Web basados en software y plataformas de diferentes vendedores. Se centra en la especificación de un conjunto de servicios Web y en los problemas más comunes encontrados en su implementación. Trata de dirigir la interoperabilidad en la capa de aplicación, en particular, la descripción de los comportamientos expuestos vía servicios Web.

Esta especificación puede ser extendida gracias a la adopción de uno o más perfiles, como los relacionados con *'Addressing'* (transporte neutral del direccionamiento de los servicios Web), *'Attachment'* (envío de documentos no XML a través de mensajería SOAP) o *'Security'* (intercambio de datos dotados de mecanismos seguros).

En principio los enlaces basados en SOAP soportan gran cantidad de modelos de mensajería, ya que el modelo de información de la especificación IMS está definido independientemente de la naturaleza del mensaje (determinado en el enlazado descrito en el fichero WSDL), sin embargo actualmente sólo hay un modelo soportado, y este es el modelo síncrono, en el que el invocador del servicio se quedará bloqueado hasta recibir la respuesta del mismo.

El resto de modelos serán añadidos cuando las necesidades de las aplicaciones lo demanden. Hay tres métodos gracias a los cuales la capacidad funcional puede ser extendida:

- Añadir nuevos mensajes SOAP: El añadir nuevas transacciones de negocio y el uso de nuevos modelos de mensajería requerirá la creación de nuevos mensajes SOAP.

- Extensión de la cabecera SOAP: La actual especificación usa la cabecera SOAP para albergar la información del estado del intercambio de datos producido entre aplicaciones. Sería recomendable que se crearan extensiones propias para dicha cabecera SOAP.

- Extensiones de los datos contenidos en el cuerpo SOAP: El cuerpo SOAP contiene la instancia XML que será usada para representar los parámetros definidos por las operaciones de intercambio en la especificación. De igual forma se necesitaría añadir nuevos parámetros o extender la estructura XML de los actuales.

La especificación IMS GWS promueve la interoperabilidad a través de diferentes plataformas o sistemas software, gracias a la implementación basada en servicios. Según se puede ver en el modelo presentado a continuación:

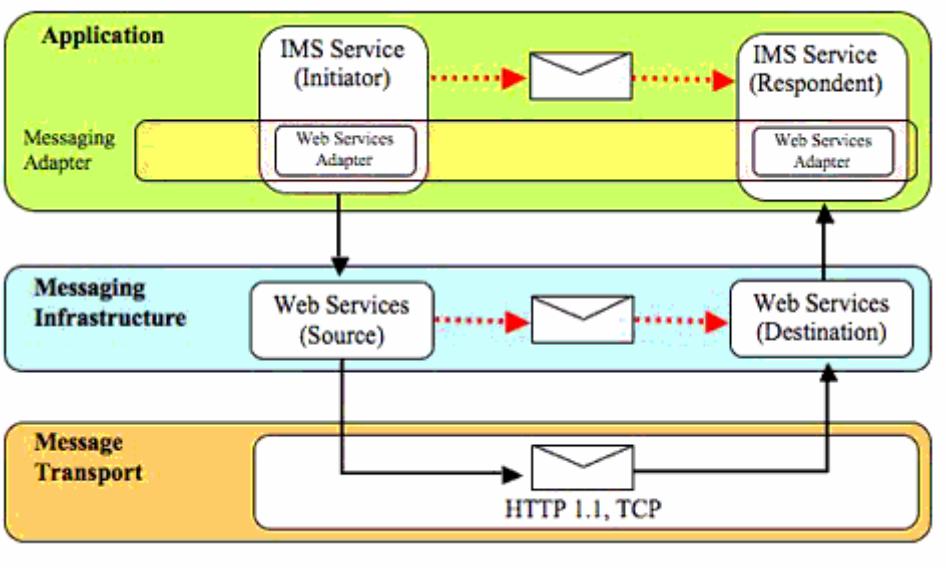

Figura 2.45. Modelo de intercambio de información en IMS GWS

Dicha especificación no está centrada en generar una arquitectura "*plug-and-play*" para garantizar la completa interoperabilidad a través de servicios Web, ya que asume que con la interoperabilidad de los protocolos de los niveles inferiores sería suficiente. Como se puede ver en la imagen mostrada, los niveles inferiores son los primeros responsables de la infraestructura de intercambio de datos a través de mensajería. Una vez que el modelo del sistema de información define las reglas de enlazado, estas serán aplicadas a crear toda la coreografía de intercambio de mensajes. Una de las principales ventajas de utilizar estas reglas de enlazado, es que el adaptador de servicios necesario en ambas partes (emisor/receptor), tiene unas características similares, lo que permitirá generar un adaptador común de servicios Web o un framework, lo cual facilitará en gran medida el trabajo a desempeñar.

Uno de los principales perfiles que se pueden destacar es el denominado "*GWS Attachments Profile*", el cual extiende dicha especificación para soportar el intercambio de información no XML a través de mensajes SOAP, gracias a la cual permitirá intercambiar ficheros *doc*, *zip*, *pdf*, etc. Esta información será incorporada al mensaje SOAP usando MTOM (*Message Transmission Optimization Mechanism*), el cual combina la eficiencia de transporte que utiliza SWA (*SOAP con Attachment*) y la flexibilidad de la codificación Base64. MTOM usa el mecanismo XOP (*XML-binary Optimization Packaging*) para de una manera eficiente colocar contenido no XML dentro de paquetes MIME.

3 PLANTEAMIENTO DEL PROBLEMA

En este capítulo se analiza la evolución de los sistemas de aprendizaje para determinar qué especificaciones deberán tener en un futuro próximo. Se realizará un estudio de los repositorios de objetos de aprendizaje más significativos describiendo sus características y las carencias que presentan a la hora de interoperar con otros sistemas y permitir la publicación y localización de sus objetos de aprendizaje. La propuesta que se presenta en el capítulo 4 de esta tesis pretende precisamente ofrecer una solución a estos problemas.

3.1 INTRODUCCIÓN

En la actualidad existen dos elementos clave para el desarrollo del e-learning: la reutilización de objetos de aprendizaje y la interoperabilidad entre los repositorios de objetos de aprendizaje. Para conseguir la completa reutilización de un objeto de aprendizaje (LO: *Learning Object*) se debe, en primer lugar, desarrollarlo conforme a estándares de e-learning, como los establecidos por IMS, ADL (SCORM) o IEEE (LOM) explicados en el capítulo anterior. De esta forma se asegura que, a través de su empaquetamiento y descripción mediante metadatos, pueda ser integrado en cualquier plataforma de e-learning compatible con estos estándares. En segundo lugar, se ha de publicar en un repositorio que garantice su localización automática por parte de los diferentes tipos de usuarios que puedan verse implicados en un proceso educativo. Pero para que esta reutilización sea completa, estos recursos deben poderse descubrir desde otros repositorios o plataformas de formación distribuidos a través de Internet.

En esta tesis se propone una arquitectura orientada a servicios multicapa basada en servicios Web para implementar estos sistemas de publicación y búsqueda, que permita el acceso universal a objetos y unidades de aprendizaje, almacenados en diferentes repositorios, a través de directorios de servicios de búsqueda publicados en registros

UDDI (*Universal Description, Discovery and Integration*). También incorporará mecanismos de actualización automática de estos servicios y se encargará de la localización de los objetos de aprendizaje independientemente de su ubicación física y de su tecnología de almacenamiento, así como de su filtrado y conversión.

En los siguientes apartados de este capítulo iremos descubriendo cual es la situación actual en cuanto a la interoperabilidad de los actuales sistemas de aprendizaje, para determinar cómo se pueden publicar y descubrir sus contenidos didácticos y como son capaces de intercambiarlos con otros sistemas o repositorios, y de esa forma determinar cuales serán las necesidades y problemáticas detectadas, que trataremos de solventar con el presente trabajo.

3.2 EVOLUCIÓN DE LOS SISTEMAS DE APRENDIZAJE

Los primeros sistemas de aprendizaje eran más cerrados y limitados que los actuales y se ejecutaban sobre ordenadores aislados. Se trataba de cursos de autoestudio y prácticamente lo único que cambiaba con respecto a un libro era el soporte (de papel a soporte magnético). Estos fueron los principios de lo que hoy se conoce como Aprendizaje Asistido por Ordenador (*Computer Based Training* – CBT), hay términos alternativos a este y que vienen a significar lo mismo como son CAI (*computer-assisted instruction*), CBI (*computer based instruction*) y CAL (*computer-assisted learning*).

La principal característica de este tipo de sistemas es su completo aislamiento, ya que estaban estructurados en un único paquete formado por el cliente y el LMS, y cada alumno disponía de una copia completa del sistema (cursos en CD-ROM u otros soportes anteriores). Estos sistemas adolecían de problemas de actualización, interactividad y otros muchos, pero para la época en que se comenzaron a utilizar también presentaban grandes avances y ventajas sobre todo en cuanto a costes y requerimientos hardware para su ejecución. Evidentemente estos sistemas no eran en absoluto distribuidos.

Siguiendo la misma evolución que cualquier otro tipo de sistema informático, se pasó a la distribución del sistema a través de las redes telemáticas. Esta distribución fue beneficiosa y se basó en las primeras redes, con clientes y protocolos específicos y extensión de los sistemas a las redes existentes, ya sean de área local o área extendida propietaria. Estos sistemas estaban formados básicamente por tres componentes activos: por un lado los equipos “clientes” de los usuarios, por otro el servidor con los contenidos y las herramientas de gestión del aprendizaje y por último la red que los comunica, como se muestra en la figura 3.1. Estos primeros sistemas aparecieron cuando las capacidades de transmisión eran reducidas y no era factible enviar por la red cierto material como el video o el audio por limitaciones de ancho de banda. Además la

capacidad de almacenamiento de servidores y la de visualización de clientes, tampoco permitían grandes alardes multimedia dados los requerimientos hardware de estos contenidos.

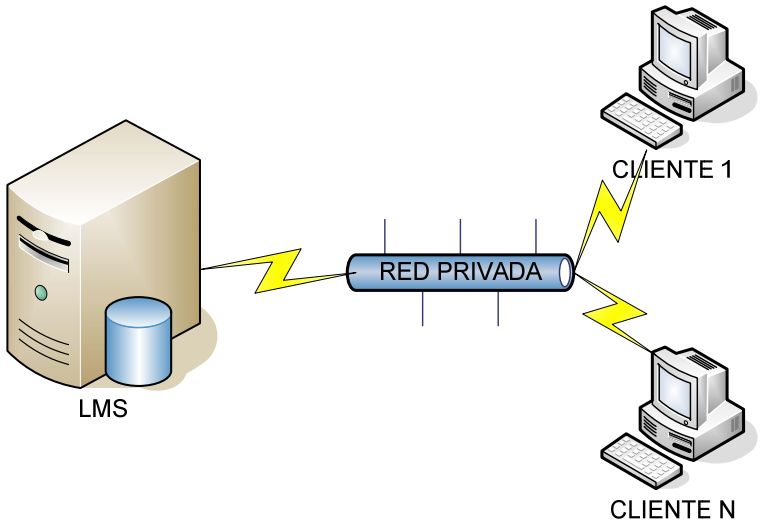

Figura 3.1. Sistema LMS basado en red local

Las redes y los equipos de hardware han seguido evolucionando a gran velocidad. En el momento actual, el ancho de banda que soportan las redes es muy elevado y los equipos tienen grandes capacidades multimedia. Además, con la aparición de Internet se ha producido una autentica revolución en la evolución de este tipo de sistemas, dando paso a protocolos estandarizados, sustituyendo protocolos y piezas de software propietarias, y su total integración con navegadores y servidores Web.

Sin embargo, a pesar de esta evolución, se siguen utilizando las mismas estructuras, el soporte de los contenidos es fijo y rígido (ya sea en el CD-ROM, el servidor específico o el servidor Web) y la interacción de los clientes se limita al uso de los contenidos del servidor. Como se aprecia en la figura 3.2, se continúa teniendo un esquema similar a los anteriores, donde el único componente nuevo es Internet; esto significa un cambio de plataforma y no de concepto.

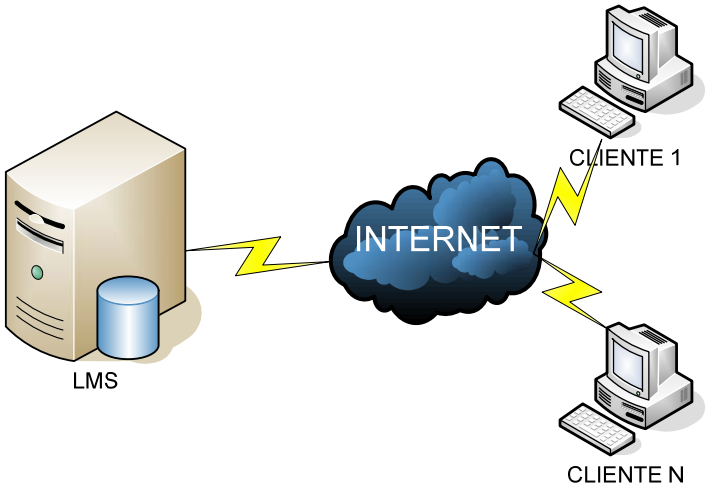

Figura 3.2. ***Sistema LMS basado en Internet***

Es en este punto actual de la evolución donde planteamos nuestro objetivo, que no es otro que el siguiente paso en la evolución de los sistemas de aprendizaje. El objetivo que planteamos es abrir los sistemas para conseguir que se comuniquen entre ellos. En esta comunicación se producirá un intercambio de contenidos donde un usuario del sistema será capaz de localizar objetos de aprendizaje fuera de su LMS o LCMS, y donde un LMS o LCMS será capaz de publicar sus objetos de aprendizaje para que sean utilizados en otros, independientemente del formato de metadatos utilizado. La figura 3.3 representa esta interconectividad entre sistemas.

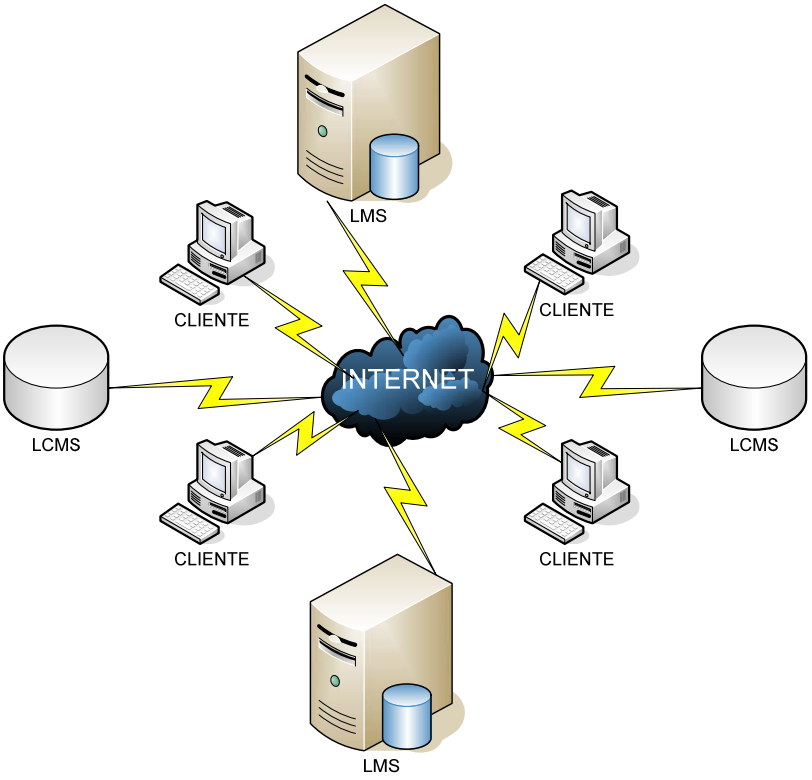

Figura 3.3. ***Sistemas que interactúan sobre la red***

Pero, desde un punto de vista arquitectural, la cuestión es cómo conseguir ésta comunicabilidad. Obviamente se necesita un nexo de unión entre todos los participantes del sistema este será el "**Sistema de Reutilización de Objetos de Aprendizaje"** al que se llamará **SROA**, que se encargará de distribuir los contenidos docentes entre los distintos participantes. Esta forma de trabajo se representa en la figura 3.4.

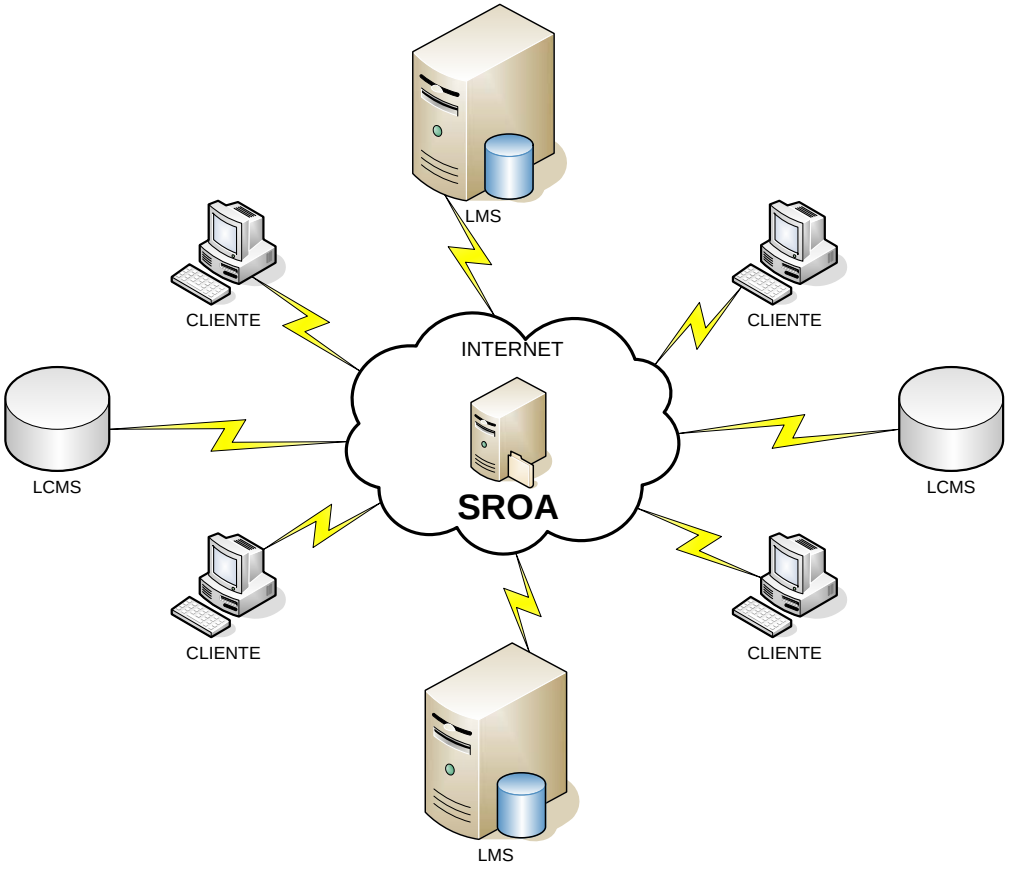

Figura 3.4. LMS, LCMS y clientes interactuando con SROA

La idea que planteamos podría servir para diversas iniciativas en la distribución de contenidos educativos, tanto de carácter libre o gratuito, como comercial. Desde la perspectiva del material libre, podría servir sobre todo en el ámbito universitario, donde se crearían nodos de compartición de conocimiento fomentando el intercambio enriquecedor de la cultura que es una de sus misiones. Desde la perspectiva comercial se podría ver como un nuevo servicio de pago para la compra de material didáctico de alta calidad para la confección de nuevos cursos.

3.3 REPOSITORIOS DE OBJETOS DE APRENDIZAJE

Actualmente existen diferentes iniciativas en la dirección marcada en el apartado anterior; se trata de repositorios de objetos de aprendizaje que tratan de compartir sus recursos. Como se comentó en el capítulo anterior, la finalidad de éstos es permitir el almacenamiento de contenidos docentes alejados de las herramientas que los utilizan para ofrecerlos al usuario. Estos repositorios tienen la misión de servir de punto de encuentro de los creadores de contenidos donde intercambiar los contenidos creados y así hacer que sean reutilizables.

Generalmente estos sistemas suelen ser gratuitos, tanto en su uso como en el precio de los contenidos que almacenan; por lo tanto, no se puede garantizar la calidad del material docente que almacenan ni en sus contenidos ni en los metadatos que los describen. Por otro lado, estos repositorios no llegan a ser útiles si no utilizan metadatos para la clasificación de sus objetos de aprendizaje [Gutiérrez y Otón, 2004], y si los utilizan suelen ceñirse a un tipo determinado de especificación para la confección de los metadatos, haciendo la búsqueda y reutilización muy compleja si se cambia esta especificación. Cierto es, también, que las especificaciones existentes, no ayudan demasiado a esta tarea por su falta de completitud y especificidad para muchos de los metadatos que deben ser rellenados para anotar cada objeto docente.

Sin embargo, este tipo de sistemas representa una pieza clave a la hora de reutilizar el material didáctico, ya que se pueden convertir en fuentes de información donde los docentes busquen y publiquen sus trabajos, pero deben ir acompañados de una metodología de creación de ese material de forma que sigan los estándares. También sería muy recomendable acompañar a estos sistemas de algún mecanismo de pago, de forma que se puedan comprar contenidos de calidad contrastada.

A continuación se analizarán algunos de los repositorios que publican sus contenidos en Internet con objeto de detectar sus propiedades y carencias. Un buen punto de entrada es la página Web http://elearning.utsa.edu/guides/LO-repositories.htm donde se publica un resumen de los principales repositorios con una pequeña descripción de sus características (figura 3.5).

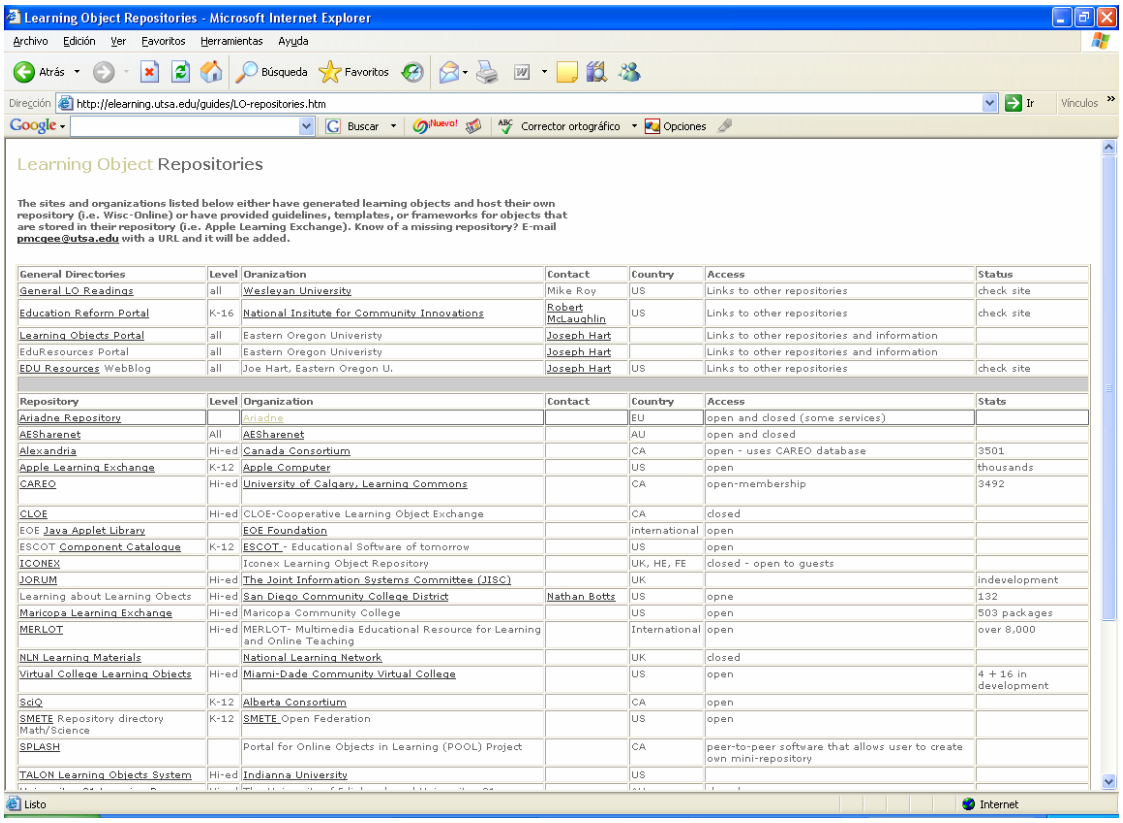

Figura 3.5. Repositorios de LO en Internet

3.3.1 MERLOT

MERLOT (*Multimedia Educational Resource for Learning and Online Teaching*) [MERLOT, 2006], es un repositorio libre y gratuito diseñado principalmente para estudiantes de educación superior o universitarios, donde puedan dejar y encontrar recursos de aprendizaje "*on-line*" y evaluarlos. En él se puede inspeccionar su catálogo de contenidos relacionados con campos como negocios, arte, humanidades, ciencia y tecnología, ciencias sociales. También se pueden realizar búsquedas sobre el mismo; para ello, los contenidos suelen llevar una pequeña descripción mediante la utilización de algunos metadatos como la descripción, autor, audiencia, idioma, coste, derechos de autor, etc. Estos metadatos se basan en el estándar LOM [IEEE, 2002].

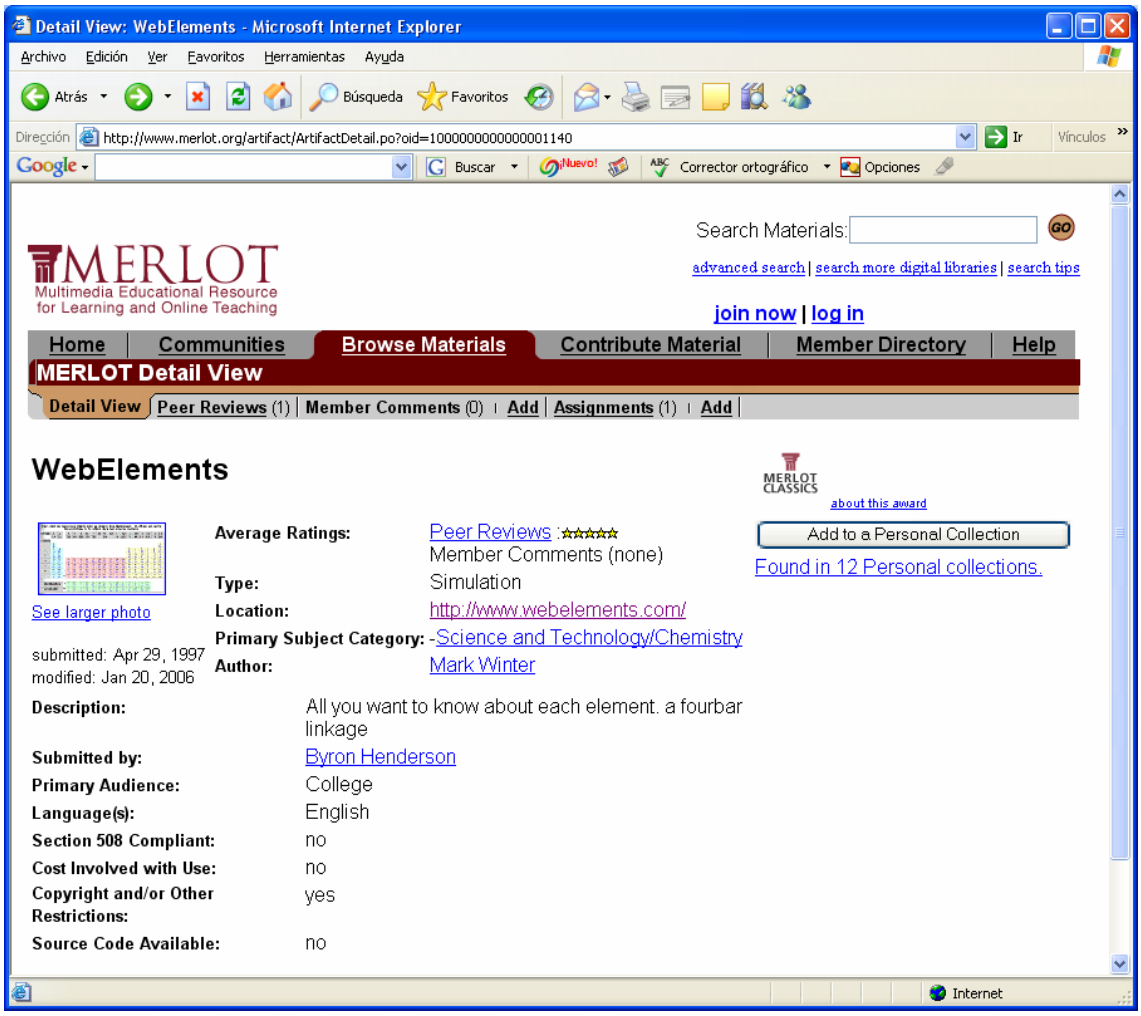

Figura 3.6. Repositorio MERLOT

El contenido físico generalmente es un enlace al contenido alojado en la Web del autor. El principal problema es que no suele ser fácil integrar ese contenido en un curso nuevo que un usuario quiera elaborar, ya que no da la posibilidad de descargarlo en un formato empaquetado estándar como SCORM, en el que se tendrán los contenidos y los metadatos agrupados. Esto es un gran inconveniente, ya que para integrarlo en otro repositorio se debería empaquetar de forma manual con una herramienta como Reload [Reload, 2006] y rellenar los campos de metadatos con la información proporcionada por MERLOT campo a campo (aquellos que proporcione). Además de este inconveniente, como se señala en [Pagés et al., 2003] la completitud en los campos de los metadatos de los objetos de aprendizaje no es lo suficientemente buena para su correcta descripción.

3.3.2 CAREO

CAREO (*Campus Alberta Repository of Educational Objects*) [CAREO, 2006] es un proyecto cuyo objetivo es alcanzar la realización de una colección de materiales de aprendizaje multidisciplinares, basados en Web, que estén a disposición de los profesores en todo Canadá e incluso fuera de las fronteras del país (figura 3.7). Es un proyecto en el que participan las universidades de Alberta, Calgary y Athabasca en cooperación con BELLE ("*Broadband Enabled Lifelong Learning Environment*") y CANAIRE ("*Canadian Network for the Advancement of Research in Industry and Education*"). El repositorio de objetos educativos CAREO es un prototipo en desarrollo. Como ocurría con el anterior, también podemos inspeccionar su catálogo o realizar búsquedas sobre el material que posee.

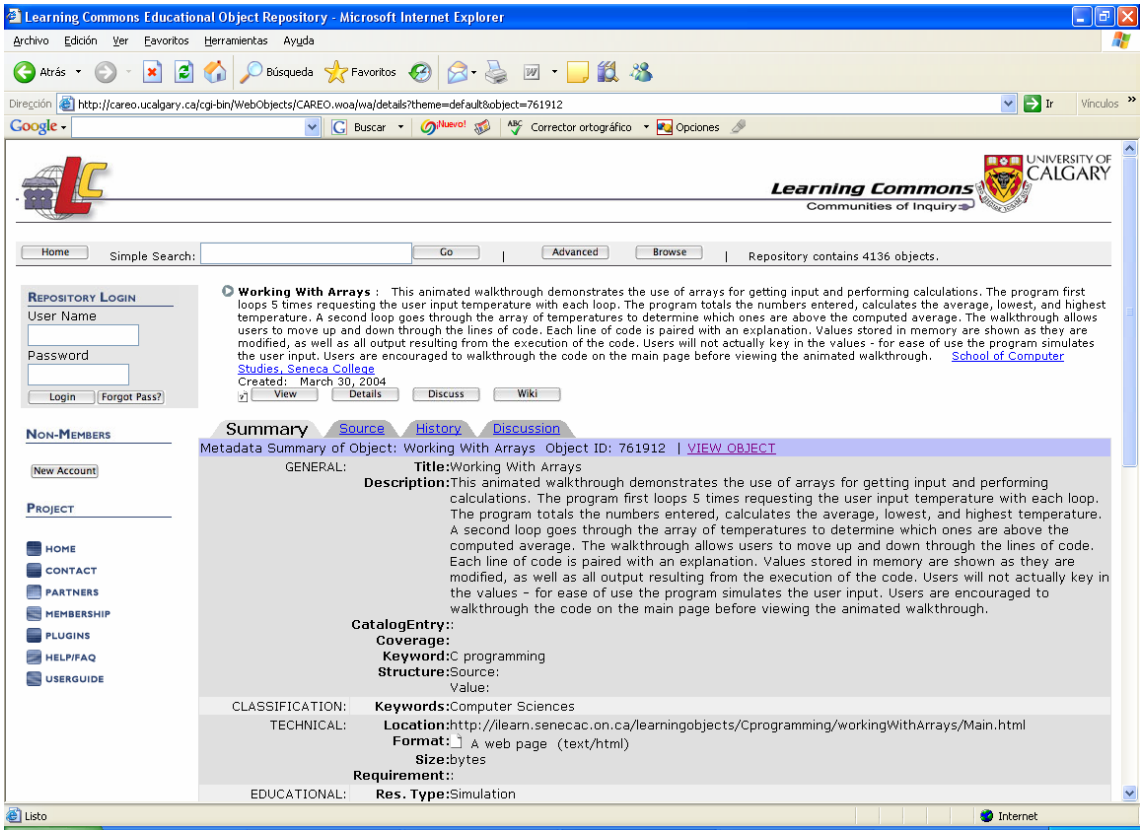

Figura 3.7. Repositorio CAREO

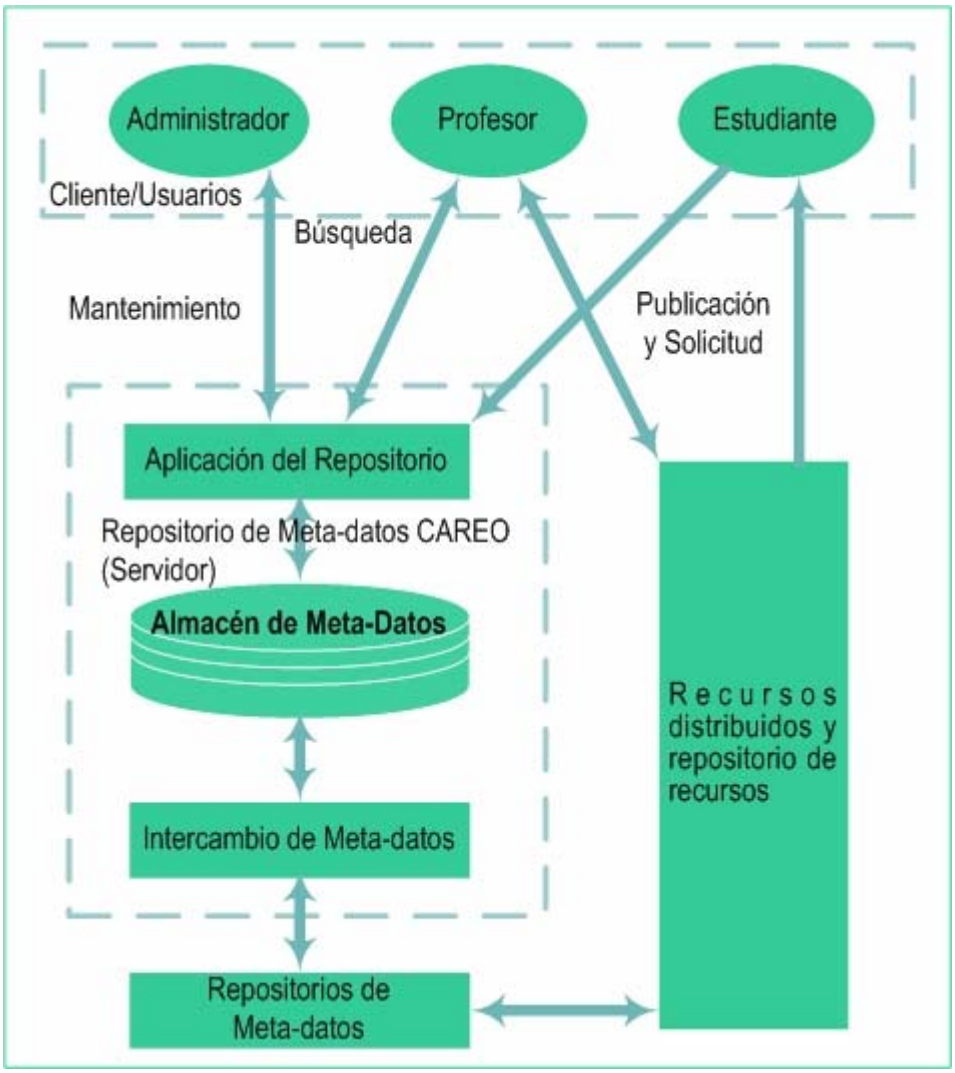

Figura 3.8. Arquitectura conceptual del repositorio CAREO

La arquitectura conceptual del repositorio CAREO (figura 3.8) se basa en una estructura de tres capas, un modelo cliente-servidor y está formada por tres componentes principales:

1. La aplicación del repositorio: esta aplicación realiza para los usuarios y administradores la búsqueda y acceso en el almacén de meta-datos (en la capa de base de datos). Presenta a los usuarios una interfaz HTML para la búsqueda y acceso a los registros de meta-datos que describen y enlazan los Objetos de Aprendizaje. Para los administradores, la aplicación presenta una interfaz HTML para el mantenimiento de funciones (proporcionando control de calidad de los registros de meta-datos y gestión de funciones particulares del repositorio).

2. Los clientes o usuarios: que pueden subdividirse en tres grupos principales: estudiantes, profesores y administradores. Los usuarios acceden a los

metadatos a través del repositorio CAREO, pero accederán a los objetos educativos distribuidos a través de un servicio Web común. Exceptuando la conectividad en Internet, la única tecnología requerida por este grupo será un navegador Web que cumpla los estándares generales.

3. El almacén de metadatos: que esta localizado en el mismo servidor en el que se encuentra la aplicación del repositorio. Está estructurado de acuerdo con el protocolo de metadatos de objetos de aprendizaje CanCore que está basado en el LOM.

El objetivo de CAREO es que los recursos u objetos de aprendizaje sean distribuidos por toda la red a través de servidores. Un recurso creado por un profesor, por ejemplo, será colocado en el servidor Web de la institución para la que esté trabajando dicho profesor. Los metadatos que describen dicho recurso serán enviados por el profesor directamente al repositorio CAREO, o bien CAREO los recogerá de otro repositorio o almacén de repositorios. CAREO también recogerá los metadatos de los repositorios que contengan ambos elementos: metadatos y recursos, empleando para ello los protocolos aceptados.

Finalmente el componente para el intercambio de metadatos del repositorio CAREO recopilará la colección de registros disponibles en otros repositorios, o almacenes de metadatos. Este componente de intercambio de metadatos será capaz de emprender las funciones básicas de mapear e interpretar, para asegurar que los metadatos recopilados se ciñen al protocolo CanCore. Este componente además proporcionará mecanismos para la creación de metadatos en el repositorio CAREO, estos metadatos serán accesibles por otros repositorios de metadatos que empleen los mismos protocolos que CAREO emplea para la recopilación de metadatos.

La diferencia más apreciable con MERLOT es que, en este caso, los metadatos que posee cada objeto de aprendizaje están mucho más detallados, y muchos de los objetos que posee los podemos descargar en un fichero comprimido, pero en el cual no se encuentran sus metadatos. Por lo tanto, adolece del mismo problema que MERLOT, ya que para poder integrar alguno de los objetos de aprendizaje en otro repositorio se necesitaría, por un lado descargar el contenido, y por otro copiar sus metadatos adaptándolos al estándar que se utilice y ensamblarlo todo.

3.3.3 ARIADNE - KNOWLEDGE POOL SYSTEM (KPS)

Es el repositorio empleado en un proyecto de la Unión Europea destinado a la creación de metodologías e instrumentos para la producción y gestión de material didáctico electrónico y los sistemas de e-learning, llamado ARIADNE (*Alliance of*

Remote Instructional Authoring and Distribution Network for Europe) [ARIADNE, 2006]. Se trata de una red europea de recursos educativos distribuidos, alrededor de la cual se han creado una serie de herramientas que ayudan a la compartición y reutilización del material educativo (figura 3.9).

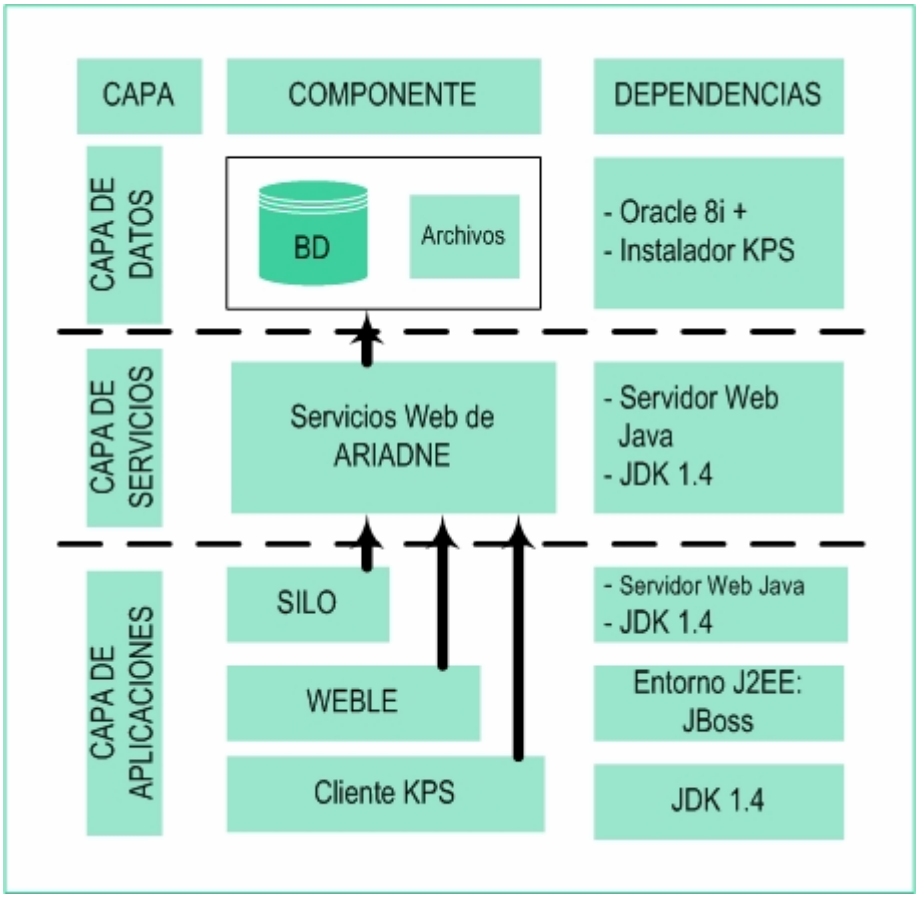

Figura 3.9. Arquitectura ARIADNE

- El núcleo central del proyecto es la distribución de los datos contenidos en un repositorio con componentes digitales reutilizables para la educación, llamado KPS (*Knowledge Pool System*).

- El gestor de KPS forma parte de la capa de datos, en la cual se almacenan los objetos de aprendizaje y los metadatos que los describen.

- El acceso al KPS para disponer de los datos, así como la gestión de cursos, se hace mediante las herramientas SILO (*Search and Index Learning Objects*) y WEBLE (*Web-Based Learning Environment*).

- KPS Client: Es la herramienta de acceso al KPS.

La principal ventaja que presenta ARIADNE es la posibilidad de hacer búsquedas en otros repositorios externos, una de las características que integramos en el sistema propuesto en la tesis. Estas búsquedas, llamadas "búsquedas federadas", permiten realizarlas de forma transparente al usuario en los repositorios de ARIADNE y MERLOT. Con esto se consigue una reutilización mayor de los objetos de aprendizaje ya que las búsquedas no solo se realizan en un solo repositorio (figura 3.10).

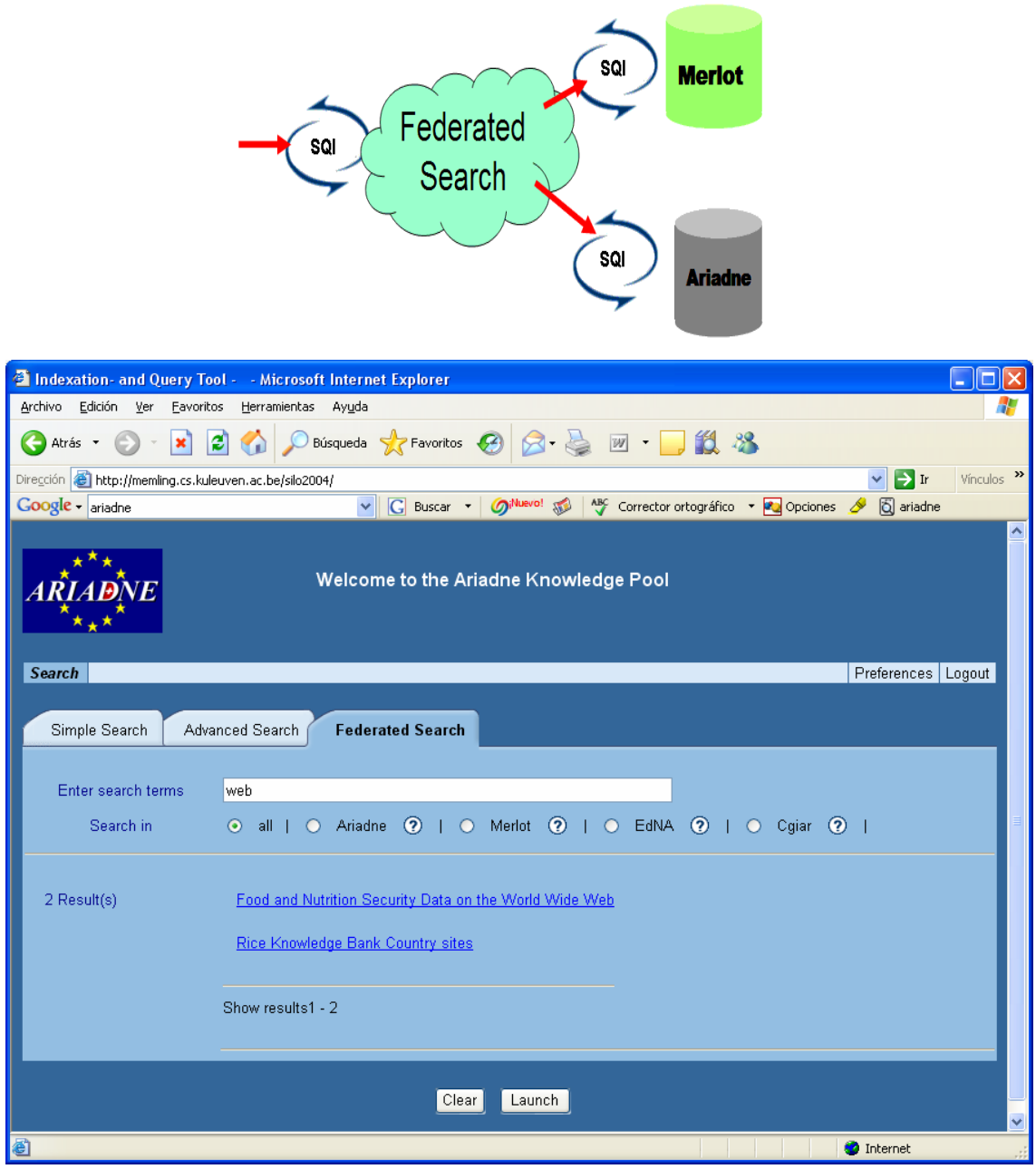

Figura 3.10. Búsquedas federadas en ARIADNE

Para realizar las búsquedas federadas se utiliza un lenguaje de consulta llamado SQI (*Simple Query Interface*) [SQI, 2006], que es un lenguaje de consulta específico de repositorios de objetos de aprendizaje para permitir la interoperabilidad entre ellos. Especifica un API de programación donde se definen un conjunto de métodos comunes

para la realización de consultas y la recuperación de resultados que un repositorio debe implementar. De esta forma se asegura la interoperabilidad entre repositorios, ya que "hablan" un mismo lenguaje entre ellos, pero que luego se traduce en llamadas a métodos locales propietarios de cada repositorio. En el sistema propuesto utilizamos una idea similar a la planteada con SQI e incluso compatible con él.

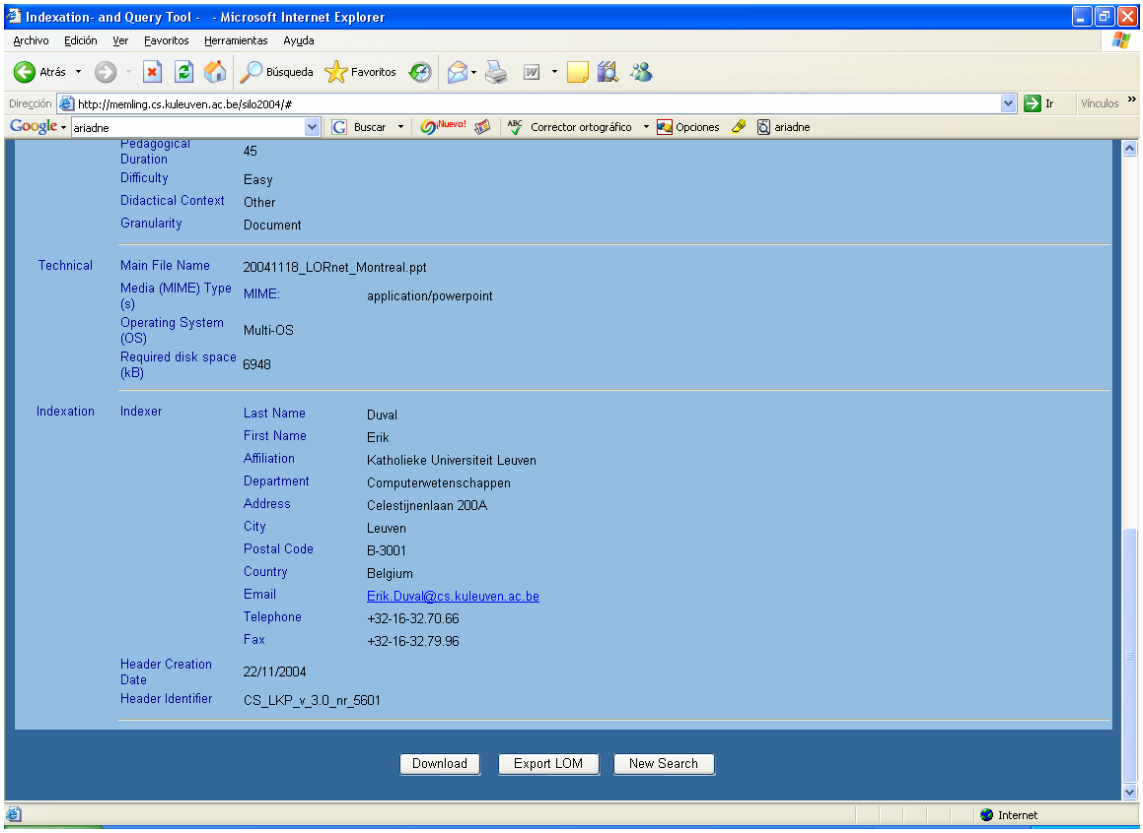

Figura 3.11. Metadatos de un LO en ARIADNE

Con respecto a la descripción de los objetos de aprendizaje que almacena, al igual que ocurría en CAREO, la completitud de sus metadatos es bastante correcta y la principal ventaja que presenta con respecto a este es que no solo permite la descarga del contenido sino que además permite exportar sus metadatos en LOM. Con esto se consigue que la reutilización de los objetos docentes sea muy grande, el único inconveniente es que el usuario final tenga que realizar una transformación de LOM (figura 3.11) al tipo de metadatos que utilice su repositorio.

3.3.4 GLOBE y CORDRA

GLOBE (*Global Learning Objects Brokered Exchange*) [GLOBE, 2006] es un consorcio internacional formado por Australia, Canadá, Europa, Japón y EEUU, cuyo objetivo es hacer que los recursos educativos sean compartidos por educadores y estudiantes de todo el mundo. El consorcio proporciona una red distribuida de objetos de aprendizaje estandarizados.

El consorcio GLOBE está formado por ARIADNE, Education Network Australia (*EdNA Online*), eduSourceCanada, MERLOT y NIME (*National Institute of Multimedia Education*). Estas organizaciones colaboran para poder tener un acceso ubicuo a recursos educativos de calidad.

Los primeros pasos que el consorcio está llevando a cabo es el desarrollo de casos de uso, especificaciones, tecnologías y reglas de negocio que permitan la búsqueda de objetos de aprendizaje a través de los repositorios que los miembros de la alianza han desarrollado. Como se podía apreciar en ARIADNE se trata de expandir la búsqueda federada a más de un repositorio.

CORDRA (*Content Object Repository Discovery and Registration/Resolution Architecture*) [CORDRA, 2006] es un modelo abierto y basado en estándares para diseñar e implementar sistemas software destinados al descubrimiento, compartición y reutilización de material docente a través de repositorios interoperables. Este modelo hace de puente entre los materiales docentes y los repositorios. Trata de identificar y especificar (no desarrollar) las tecnologías apropiadas y los estándares de interoperabilidad para combinarlas en un modelo de referencia.

Por lo tanto, CORDRA es un modelo formal que puede ser usado para el diseño de repositorios federados y su interoperabilidad.

Las actividades de CORDRA se coordinan entre ADL (*Advanced Distributed Learning Initiative*), CNRI (*Corporation for National Research Initiatives*) y LSAL (*Learning Systems Architecture Lab*).

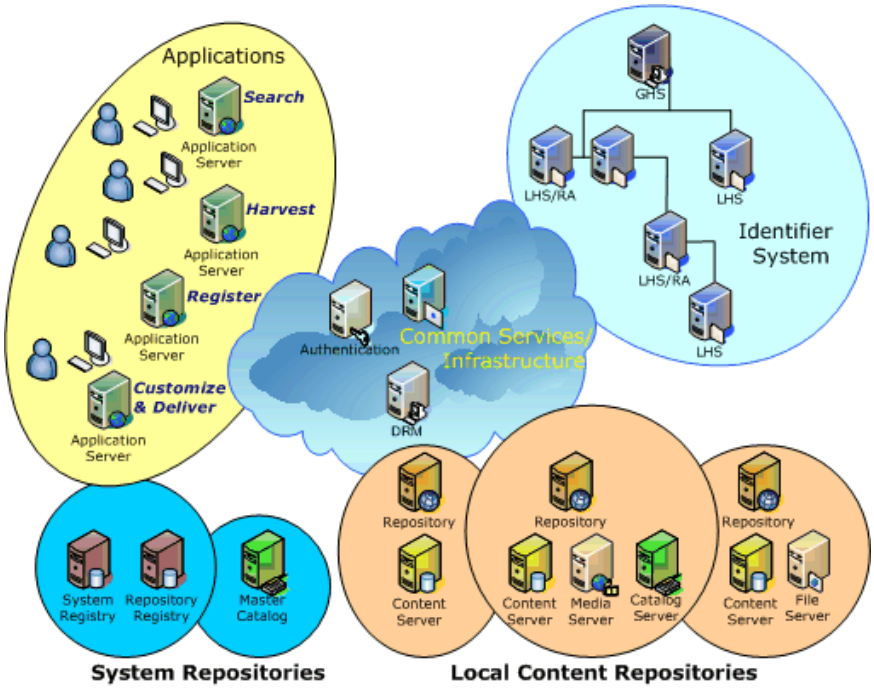

Figura 3.12. El modelo CORDRA

Los componentes del modelo CORDRA se pueden resumir en los siguientes:

- *Local Content Repositories*: Representan los repositorios locales donde se encuentra el material docente y su metainformación asociada.

- *System Repositories*: Integrado por los sistemas de CORDRA entre los que se encuentran el registro de repositorios y sistemas y el catálogo maestro.

- *Identifier System*: Infraestructura para la identificación de los distintos objetos de aprendizaje registrados a los que se puede acceder.

- *Common Services Infrastructure*: Servicios del núcleo del sistema y servicios administrativos usados para la implementación del modelo, por ejemplo, autentificación, derechos de autor, reglas de procesamiento, etc.

- *Applications*: Aplicaciones e interfaces usados para el manejo de los objetos de aprendizaje por los usuarios finales tales como búsqueda, registro, entrega, etc.

El proceso general de uso podría ser el siguiente: Un repositorio que contiene material docente con metainformación y unos métodos de gestión propios que quiere pertenecer al sistema federado CORDRA debe inscribirse en el registro de repositorios para permitir su descubrimiento, acceso y administración. Los registros federados

mantienen un catálogo maestro con los metadatos de todos los objetos de aprendizaje que contienen. Mediante este catálogo se pueden realizar las búsquedas de los objetos de aprendizaje y se mantiene un índice de los mismos para ayudar a realizar dichas búsquedas. El sistema de identificación permite la identificación, registro y definición de los objetos de aprendizaje. Mediante un conjunto de servicios comunes se da acceso a toda la funcionalidad del sistema. Por último, mediante una interfaz se proporcionan todos estos servicios a los usuarios que pueden acceder o publicar los objetos docentes.

Aunque las propuestas de GLOBE y CORDRA son las mejores soluciones a los problemas de la publicación y descubrimiento de los objetos de aprendizaje contenidos en un repositorio, en realidad se tratan sólo de modelos propuestos en los cuales no se entra en detalle en cuanto a cómo construir sistemas que se adapten al mismo.

En el caso de GLOBE aún no se ha definido ninguna propuesta detallada de como está definido su modelo, tan solo algunos casos de uso del mismo. La única referencia clara es la de ARIADNE que sí detalla algo más sus búsquedas federadas e incluso publica las herramientas necesarias para montar un sistema similar.

CORDRA es el modelo más detallado y definido en el que se explica con claridad el diseño a seguir, pero no como llevarlo a cabo. ADL está implementando parte del modelo y construyendo un registro de repositorios llamado ADL-Registry [ADL-R, 2006] que permita la publicación y búsqueda de objetos de aprendizaje.

Como conclusión a este apartado, comentaremos algunas de las principales diferencias que se han encontrado entre este modelo, y el que se va a presentar en esta tesis. Según CORDRA, su arquitectura contaría con un gran catálogo, en el que se indexaría el contenido de todos los repositorios. Es decir, se indexarían todos los objetos de aprendizaje, de todos y cada uno de los repositorios sobre los que se realizarían las futuras búsquedas. A priori, esta tarea parece bastante pesada para una arquitectura, ya que se plantean algunas preguntas: ¿cada cuánto tiempo habrá que reorganizar estos índices?, ¿qué tiempo puede suponer el realizar una búsqueda a través de un índice que crece exponencialmente?, ¿qué ocurre cuando se añaden nuevos objetos de aprendizaje o se modifican?, ¿es necesario reorganizar estos índices cada vez que se añadan nuevos objetos o nuevos repositorios? Esta tarea puede acabar siendo un gran cuello de botella para las búsquedas del sistema.

Sin embargo, la solución planteada en esta tesis es mucho más eficiente en este aspecto, ya que será un servicio Web asociado al repositorio el que se encargue de buscar objetos de aprendizaje en ese repositorio. El utilizar indexaciones para realizar las búsquedas puede resultar muy eficiente, pero en nuestro caso, se hace de forma interna en cada repositorio, y así evitamos la necesidad de tener que indexar todos los objetos de aprendizaje de todos los repositorios. La arquitectura presentada, no cuenta

con ninguna estructura en la que se almacenen todos los contenidos docentes de todos los repositorios que se van a analizar, ya que la cantidad de datos con la que se tendría que trabajar sería completamente excesiva.

Otra de las diferencias que se pueden observar es la utilización de estándares; es evidente que para que un objeto de aprendizaje sea reutilizable, debe crearse conforme a la definición de algún estándar, y por lo tanto el contenido del objeto debe estar descrito mediante metadatos. Pero una de las principales características con las que cuenta nuestra arquitectura, es que no está ligada a ningún estándar, por lo que es completamente válida, independientemente del estándar escogido. Sin embargo el modelo CORDRA está muy ligado a ADL, y con ello a SCORM, por lo tanto aquí se plantea otra duda, ¿qué ocurre con aquellos repositorios en los que se utilicen otros estándares? ¿Y si el usuario utiliza una plataforma (LMS) incompatible con SCORM? Desde el punto de vista del autor, una herramienta que busque una reutilización universal de objetos de aprendizaje, no debería estar ligada a un estándar determinado; más aún, cuando existen en el mercado varios de ellos, y no existe una unidad en cuanto a su utilización y total compatibilidad. Por ello, encontramos en este apartado, otro punto a favor de la arquitectura presentada.

Aún así, es pronto para poder evaluar completamente este modelo, ya que como se ha comentado anteriormente, CORDRA no cuenta todavía con ningún prototipo con el que poder evaluar su contenido teórico. Así que hasta que esto no ocurra, no podremos comparar completamente las dos arquitecturas. Este es otro de los puntos a favor que encontramos, ya que el trabajo de investigación llevado a cabo en esta tesis, además de generar una propuesta de arquitectura, se ha materializado en un prototipo completamente funcional, que pone de manifiesto todas las expectativas obtenidas durante el desarrollo teórico de la tesis.

3.4 JUSTIFICACIÓN DEL OBJETIVO DE LA TESIS

Sin duda, el mayor problema que aborda la industria del e-learning en la actualidad, aún sin resolver en aspectos muy fundamentales, es la ausencia de unas metodologías técnicas, documentales y psicopedagógicas comunes y aceptadas que garanticen los objetivos de accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales docentes. Por otro lado, el rápido avance de las infraestructuras tecnológicas y la falta de compatibilidad entre tecnologías e-learning existentes es una de las principales barreras para su desarrollo, y hace de la adopción de estándares el instrumento crucial para su correcta y rápida evolución. Por estos motivos, actualmente se está produciendo una convergencia hacia estándares comunes e intercambiables y la definición de modelos que permiten la interoperabilidad de gestores de contenidos docentes.

Ya que el requisito indispensable de los estándares es garantizar la interoperabilidad entre los componentes de diferentes vendedores [Henderson, 2003], es decir, que distintos sistemas o plataformas puedan intercambiarse información y trabajar conjuntamente, es necesario que las plataformas de construcción de componentes e-learning y las herramientas de autoría se basen en ellos. Al estar los estándares en evolución, no es extraño encontrar productos que no siguen estrictamente un estándar de una organización en concreto, sino que han sido desarrollados aplicando diferentes recomendaciones de los estándares más comunes en cada una de sus partes, adaptando de esta manera los estándares a las necesidades del producto.

Queda claro que para que un objeto de aprendizaje sea reutilizable debe crearse conforme a la definición de algún estándar y, por lo tanto, el contenido del objeto debe estar descrito mediante metadatos. Sin embargo, los estándares no ofrecen ninguna pauta sobre cómo puede ser descubierto un objeto de aprendizaje. Por lo tanto, los metadatos son el primer paso para hacer que un objeto de aprendizaje sea reutilizado, pero faltan otros muy importantes como el de su descubrimiento e intercambio entre diversas plataformas de formación.

Resumiendo lo anterior, podemos afirmar que el material educativo debe ser:

- Desarrollado conforme a estándares.
- Descubierto de forma universal.
- Fácil de encontrar.
- Independiente del formato de metadatos.
- Reusable.
- Independiente de la plataforma que lo almacena.
- Integrable en otros sistemas de formación.

Como se indicó al principio de este capítulo, en el apartado de evolución de los sistemas de aprendizaje, éstos han evolucionado hacia la comunicación a través de Internet, ya sea para comunicarse con sus usuarios o para interoperar con otros sistemas. La principal característica de los nuevos entornos de aprendizaje virtuales es que utilizan la Web como única plataforma de distribución; esto implica que disponen de una serie de capacidades hasta ahora inexistentes para las aplicaciones de enseñanza asistida por ordenador, que permiten superar ciertas deficiencias, tales como los elevados costes de producción y tiempos de desarrollo. Sin embargo, se aprecia una reutilización prácticamente nula tanto de los materiales docentes como de los componentes software que integran una herramienta de teleformación.

Estos problemas los resume Koper [2000] en los siguientes puntos:

- El aumento en la heterogeneidad de los productos y en la interacción entre personas y sistemas, únicamente entre personas y únicamente entre sistemas.

- El espectacular aumento de información disponible y su dispersión en distintos sistemas y aplicaciones, lo que implica la necesidad de poner en comunicación distintos productos software y plataformas.

- La organización de procesos de aprendizaje distribuidos, motivada por la dispersión geográfica de los usuarios de los cursos.

Y es para solucionar estos inconvenientes por lo que surgen diferentes iniciativas que pretenden unificar, tanto la forma de crear contenidos, como la forma de implantar plataformas y repositorios educativos, y de comunicar esos contenidos. Ofrecer una solución a estos inconvenientes es exactamente el objetivo de esta tesis.

Para solucionar los problemas planteados necesitamos establecer la arquitectura de un sistema software capaz de asumir las funcionalidades derivadas de esta problemática, que se puede resumir en la interoperabilidad de sistemas de e-learning (su integración) y en la reutilización de los objetos de aprendizaje que contienen.

Teniendo en cuenta la arquitectura de un sistema e-learning desarrollado mediante la integración de un conjunto de componentes independientes, debería tener las siguientes características [Cisco, 2001]:

- **Abierta.** El objetivo es crear aplicaciones e-learning interoperables y conectables entre sí de forma sencilla ("*plug-and-play*"), es decir, que herramientas comerciales de fabricantes distintos puedan ensamblarse en un único sistema global. Para ello es necesario que el marco de definición de la arquitectura del sistema sea conforme a un modelo estándar.

- **Escalable.** Independientemente del tamaño inicial con que se conciba el sistema, la arquitectura debe estar definida de tal forma que permita su crecimiento. Por ejemplo, al ir aumentando los repositorios de objetos educativos las aplicaciones encargadas de gestionarlos deben tener capacidad suficiente para no sobrecargarse.

- **Global.** Permitir la diversidad lingüística y cultural. Este es uno de los objetivos más difíciles de conseguir, puesto que la gran mayoría de las aplicaciones están destinadas para una audiencia anglosajona: actualmente existen varios esfuerzos de estandarización en marcha cuyo objetivo es

presentar un mismo contenido (incluso un mismo entorno de aprendizaje) en diferentes lenguas en función del usuario a quien esté destinado.

- **Integrada.** No sólo entre los componentes del propio sistema, sino entre otras aplicaciones no directamente relacionadas con el aprendizaje (por ejemplo, de gestión de recursos humanos, financieras, de gestión del conocimiento, etc.). El objetivo es conseguir la interoperabilidad entre todas ellas.

- **Flexible.** Es importante la capacidad de implementar nuevas soluciones sin tener que efectuar grandes cambios en la arquitectura del sistema.

Por lo tanto, el objetivo propuesto en esta tesis es el de definir una arquitectura orientada a servicios de un sistema capaz de resolver los problemas planteados anteriormente e implementar un prototipo real, para demostrar con su funcionamiento que la arquitectura planteada se puede llevar a la práctica.

4 PROPUESTA ARQUITECTURAL

Después de mostrar el estado del arte de los sistemas de formación basados en Internet y de los servicios Web y SOA, pasaremos a especificar una arquitectura software para la construcción de sistemas que ayuden a resolver los problemas planteados en el capítulo anterior que se pueden resumir en la reutilización de los objetos de aprendizaje y la interoperabilidad de los sistemas de formación. Se analizará la funcionalidad básica que debe implementar la arquitectura, así como los actores implicados en la misma, y sus relaciones, para finalmente proporcionar un conjunto de capas que compondrán la arquitectura y se describirán sus componentes.

En el siguiente capítulo se presentará un prototipo software desarrollado para validar la arquitectura propuesta.

4.1 INTRODUCCIÓN

Antes de realizar una propuesta de arquitectura, es importante tener claro lo que se entiende precisamente por "arquitectura de un sistema software". Para ello utilizaremos una metáfora, comparando la arquitectura del software con la arquitectura de edificios. Un edificio es habitualmente una sola unidad desde la perspectiva del cliente. El arquitecto del edificio puede encontrar útil hacer una maqueta a escala del edificio, junto con los dibujos del edificio visto desde distintas perspectivas. Como un edificio, un sistema software es una única entidad, pero al arquitecto de software y a los desarrolladores les resulta útil presentar el sistema desde diferentes perspectivas para comprender mejor el diseño. Estas perspectivas son vistas del modelo del sistema. Todas las vistas juntas representan la arquitectura.

La arquitectura software abarca decisiones importantes sobre:

- La organización del sistema software.
- Los elementos estructurales que compondrán el sistema y sus interfaces, junto con sus comportamientos, tal y como se especifican en las colaboraciones entre estos elementos.
- La composición de los elementos estructurales y del comportamiento en subsistemas progresivamente más grandes.
- El estilo de la arquitectura que guía esta organización: los elementos y sus interfaces, sus colaboraciones y su composición.

Sin embargo, la arquitectura software está afectada no sólo por la estructura y el comportamiento, sino también por el uso, la funcionalidad, el rendimiento, la flexibilidad, la reutilización, la facilidad de comprensión, las restricciones y compromisos económicos y tecnológicos, y la estética.

Un sistema grande y complejo requiere una arquitectura para que los desarrolladores puedan progresar hasta tener una visión común. Un sistema software es difícil de abarcar visualmente porque no existe en un mundo de tres dimensiones. Suele utilizar tecnología poco probada o una mezcla de tecnologías nuevas. Tampoco es raro que el sistema lleve a sus últimos límites la tecnología existente. Además, debe ser construido para acomodar gran cantidad de componentes que sufrirán cambios futuros. A medida que los sistemas se hacen más complejos, “los problemas de diseño van más allá de los algoritmos y las estructuras de datos para su computación”.

Se necesita una arquitectura de sistema para:

- Comprender el sistema
- Organizar el desarrollo
- Fomentar la reutilización
- Hacer evolucionar el sistema
- La utilización del sistema con un rendimiento óptimo

Si hay algo de lo que podemos estar seguros, es que cualquier sistema de un tamaño considerable evolucionará. Evolucionará incluso aunque aún esté en desarrollo. Más tarde, cuando esté en uso, el entorno cambiante provocará futuras evoluciones. Hasta que esto ocurra, el sistema debe ser fácil de modificar, esto quiere decir que los desarrolladores deberán ser capaces de modificar partes del diseño o implementación sin tener que preocuparse por los efectos inesperados que puedan tener repercusión en el sistema. En la mayoría de los casos, deberían ser capaces de implementar nuevas funcionalidades en el sistema sin tener que pensar en un impacto dramático en el diseño

e implementación existentes. En otras palabras, el sistema debe ser en sí mismo flexible o tolerante a los cambios. Las arquitecturas del sistema pobres, suelen degradarse con el paso del tiempo y necesitan ser "parcheadas" hasta que al final no es posible actualizarlas con un coste razonable.

4.2 DEFINICIÓN DE ARQUITECTURA

No existe una definición universalmente aceptada, del concepto de "Arquitectura". Así, por ejemplo para Bass et al. [1997], la arquitectura de un sistema es la estructura del sistema que comprende sus componentes, las propiedades de estos componentes y las relaciones entre ellos.

Es decir, según estos autores:

- La arquitectura define componentes, pero define lo necesario para su interrelación no sus características propias.
- Un mismo sistema puede definirse con más de una arquitectura y no se especifica que tipo de componentes son los que forman el sistema, por ejemplo en los sistemas software no se especifica si los componentes son objetos, procesos, librerías, etc.
- Todos los sistemas tienen una arquitectura ya que todos están formados por componentes y tienen relaciones entre ellos.
- El funcionamiento de cada componente es parte de la arquitectura desde el punto de vista del resto de los componentes.

Para otros autores, como Jazayeri et al. [2000], la arquitectura de un sistema es un conjunto de conceptos y decisiones de diseño sobre la estructura del sistema, que deben ser tomadas en cuenta antes de su realización para satisfacer de una manera efectiva los requisitos de funcionamiento y calidad implícitos en el sistema.

Shaw et al. [1996] refiriéndose a los sistemas software clasifican las vistas de estos sistemas en:

- Modelo estructural: Todos los sistemas software están compuestos de componentes y relaciones entre los componentes, más algunos otros aspectos que incluyen: configuración, estilo, requisitos, semántica, análisis y propiedades.

- Modelo de entorno de trabajo: Es similar al modelo estructural pero enfatiza la coherencia del sistema completo en oposición a la definición de los componentes singulares.

- Modelo dinámico: Se refiere a la calidad del sistema, entendiendo por dinámico la posibilidad de cambios en la configuración del sistema.

- Modelo de procesos: Centra la atención en la construcción de la arquitectura y los pasos o procesos involucrados en esa construcción.

Para Booch et al. [1999], la arquitectura es el conjunto de decisiones significativas acerca de la organización de un sistema, la selección de sus elementos estructurales y las interfaces que componen el sistema, junto con la colaboración entre los elementos estructurales y la descomposición en subsistemas según el funcionamiento de estos elementos. Por tanto, la arquitectura se compone de elementos que forman el sistema, sus interfaces, la colaboración entre ellos y su composición.

Para Boehm y Port [1998], una arquitectura de sistemas comprende:

- Un conjunto de componentes del sistema, conexiones y restricciones.

- Un conjunto de requisitos del sistema.

- Un razonamiento que demuestre que los componentes, sus conexiones y restricciones que definen el sistema, una vez implementados, satisfacen el conjunto de requisitos del sistema.

La Architecture Tradeoff Analysis [ATA, 2003] proporcionó las siguientes definiciones desde distintos puntos de vista:

- Desde un punto de vista técnico, una arquitectura es el conjunto mínimo de reglas que gobiernan la interdependencia de las partes o elementos que juntos se usan para formar un sistema.

- Desde un punto de vista operativo, una arquitectura es una descripción, a veces gramatical, que define los elementos y las necesidades de intercambio de información entre ellos en cuanto a tipo de información y frecuencia de intercambio.

- Desde un punto de vista de sistemas, una arquitectura es una descripción, a menudo gramatical, de la solución utilizada para satisfacer los requisitos del sistema y los parámetros de medida de cumplimiento de requisitos.

Soni et al. [1995], también han definido una arquitectura desde distintos puntos de vista, según estos autores:

- Conceptualmente, una arquitectura describe un sistema en función de los principales elementos del diseño y las relaciones entre ellos.

- Modularmente, una arquitectura establece la descomposición funcional del sistema y sus distintos niveles.

- Desde un punto de vista ejecutivo, una arquitectura debe describir la capacidad y estructura dinámica del sistema.

- La codificación de una arquitectura detalla como están organizados los elementos que forma el entorno de desarrollo del sistema.

IEEE en su glosario estándar sobre Ingeniería del Software define el término arquitectura como "la estructura organizacional de un sistema o componente" y el término "diseño arquitectural" como "el resultado del proceso de definición de una colección de componentes hardware y software y sus interfaces para establecer el *framework* de desarrollo de un sistema" [IEEE, 1990].

También IEEE, en su estándar 1471, ofrece otra definición más detallada de arquitectura como "la organización fundamental de un sistema expresado en sus componentes, relaciones con los demás componentes y con el entorno, y los principios que guían su diseño y evolución" [IEEE, 2000]. En este estándar se indica que la descripción de la arquitectura de un sistema debe:

- Especificar los participantes en el sistema.

- Identificar los objetivos del sistema en términos de:

    - Funcionalidad, planteando la cuestión de qué necesita hacer el sistema.
    - Rendimiento, considerando cómo se comportará el sistema ante situaciones extremas de carga de trabajo.
    - Seguridad, teniendo en cuenta si el sistema puede proteger adecuadamente la información del usuario.
    - Viabilidad, planteándose si realmente se puede implementar el sistema.

- Organizarse en una o más vistas de la misma arquitectura.

- Ofrecer alternativas para que los arquitectos puedan tomar decisiones.

4.2.1 Objetivos de una arquitectura

Según Joe Batman, del *Software Engineering Institute* [Batman, 1999], los objetivos que debe cumplir una arquitectura efectiva son los siguientes:

- La arquitectura se debe usar como un plan prescriptivo de la construcción del sistema, estructurando su resultado.

- Mediante la arquitectura debe ser fácil identificar los modelos estructurales del dominio de aplicación, intentando proporcionar la solución más simple.

- La arquitectura debe identificar las partes del sistema, ordenándolo mediante su descomposición.

- La arquitectura debe aislar los componentes del sistema, presentando las instrucciones y líneas de acción necesarias para llegar al detalle de la descomposición.

- La arquitectura debe identificar las relaciones con sistemas externos.

- El entorno de desarrollo del sistema debe estar soportado por la arquitectura y ésta debe facilitar su conocimiento, en cuanto a servicios y utilidades.

- La arquitectura debe soportar los incrementos adicionales de capacidad del sistema.

- La arquitectura debe utilizar estándares para uniformizar problemas y soluciones en cuanto a nomenclatura, estilo, planificación, procedimientos, etc.

- Ya que las excepciones son inevitables, la arquitectura debe anticipar lo necesario para cubrirlas.

- La arquitectura debe proporcionar documentación adecuada.

- Para el uso efectivo de la arquitectura debe realizarse entrenamientos y dar soporte a los cambios culturales que conlleva.

- El control de calidad debe estar integrado en la arquitectura.

- La arquitectura debe proporcionar propiedades estructurales y de calidad que permitan un proceso de prueba eficiente.

Allamaraju [1998], cuando realiza un estudio de la arquitectura Paradox, afirma que un arquitecto para cumplir su cometido debe, en una primera fase, basándose en los requisitos del sistema diseñar un esbozo de la arquitectura y establecer los objetivos que debe cumplir para satisfacer estos requisitos, a continuación, establecer los principios que debe seguir para cumplir los objetivos definidos e integrar esos principios en el diseño de la arquitectura y, por último, evaluar y redefinir de acuerdo a los resultados la arquitectura para todos los casos de uso, midiendo parámetros de facilidad de adaptación al tamaño de los sistemas, prestaciones, integración, etc.

4.2.2 Características de una arquitectura

Vistos los objetivos a los que debe tender una arquitectura, veamos que características debe cumplir para poder lograr estos objetivos.

Según el Grupo de Ingeniería del Software de la Universidad de Málaga [UMA, 2006] es muy importante la componibilidad, es decir, si sus componentes presentan comportamientos compatibles y pueden ser combinados de forma segura para formar un sistema. Además se asegura la reutilización si se comprueba que un cierto componente puede ser utilizado en un nuevo sistema donde se necesita un comportamiento similar, incidiendo en la compatibilidad de los componentes. Estas características conducirán a una arquitectura a poseer propiedades relativas a consistencia y operatividad.

Allamaraju [1998] aprecia como esenciales las siguientes características:

- Una arquitectura debe representar un punto de vista elevado de los sistemas que revele su estructura, pero esconder todos los detalles de implementación.

- Una arquitectura debe representar todos los posibles casos de uso. Esto garantiza que la estructura representada por la arquitectura responda a todos los requerimientos funcionales.

Según Batman [1999], las características que debe cumplir una arquitectura efectiva son los siguientes:

- Debe ser reutilizable y exportable, para lo cual debe cubrir los requerimientos del dominio entero.

- Debe ser compacta, es decir debe ser lo suficientemente flexible para adaptarse sin comprometer la integridad conceptual.

- Debe tener un alto nivel de abstracción.

- Debe estar bien documentada, de forma clara y no ambigua.

Para el SEI (*The Software Engineering Institute*), que ha trabajado en la ATA [2003] con el objetivo de establecer técnicas válidas para representar arquitecturas y promover su conocimiento, los atributos de calidad que debe tener una arquitectura son [Barbacci et al., 1997]:

- Rendimiento
- Posibilidad de Modificación
- Disponibilidad
- Seguridad
- Interoperabilidad

Garlan et al. [1995], creen que las principales características de la configuración de una arquitectura se dividen en tres categorías generales:

- Cualidades de la descripción de la configuración: especificaciones comprensibles, modularidad, refinamiento y trazabilidad y heterogeneidad.

- Cualidades del sistema descrito: heterogeneidad, escalas diversas, evolución y dinamismo.

- Propiedades del sistema descrito: dinamismo, restricciones y propiedades no funcionales.

Veamos la definición de cada una de estas características:

- Especificaciones comprensibles: Uno de los papeles principales de una arquitectura es servir de vía de comunicación entre los distintos componentes de un proyecto y facilitar el entendimiento del sistema a un nivel alto de abstracción.

- Modularidad: La modularidad o descomposición jerárquica permite describir los sistemas según diferentes niveles de detalle; por ejemplo, se pueden representar estructuras complejas como un único componente del sistema.

- Refinamiento y trazabilidad. La arquitectura debe poder ser revisada y corregida en distintas fases de refinamiento, y debe ser posible seguir los cambios realizados en cada una de esas fases.

- Heterogeneidad: Un objetivo de cualquier arquitectura es facilitar el desarrollo de sistemas a gran escala, posiblemente integrados por componentes y conectores de tipos distintos y soportados por entornos diferentes.

- Escalas diversas: La arquitectura intenta proporcionar a sus usuarios herramientas válidas para tratar sistemas de complejidad y tamaño diversos.

- Evolución: La arquitectura debe evolucionar y así poder reflejar los cambios de los sistemas, ya que sus componentes y conectores se incrementan, se modifican y cambian sus relaciones.

- Dinamismo: El término evolución refleja los cambios externos de una arquitectura (y sus sistemas resultantes); en cambio, dinamismo se refiere a la modificación de una arquitectura y sus sistemas resultantes mientras éstos se están ejecutando; son sistemas que cambian dinámicamente y la arquitectura que los soporta debe estar basada en una evolución *run-time*. De este tipo son los sistemas de control de tráfico y de redes telefónicas.

- Restricciones: Son dependencias de la configuración completa debido a restricciones en los componentes o conectores individuales. La seguridad y prestaciones de un sistema completo siempre depende de la seguridad y prestaciones de sus componentes.

- Propiedades no funcionales: Son propiedades referentes al sistema completo y no a sus componentes ni conectores. Se refieren a ayudas a la dirección del proyecto, análisis, restricciones de la dirección, etc.

Por su parte, OMG [2001], ha establecido que las características necesarias para que una arquitectura soporte la interoperabilidad e integración durante todo el ciclo de vida de un sistema, separando especificaciones de realización, son:

- Posibilidad de utilización con distintas tecnologías.
- Portabilidad a través de múltiples plataformas.
- Integración basada en un modelo de relaciones entre diferentes dominios.

4.2.3 Factores que influyen en la arquitectura

Existen diferentes tipos de factores que influyen en la arquitectura, como puede apreciarse en la figura 4.1. En primer lugar se encuentran los casos de uso, un caso de uso es una secuencia típica de acciones en un sistema, desde el punto de vista del usuario, que muestra cómo el sistema interacciona con el exterior y que se obtiene como

resultado del uso del mismo [Jacobson et al., 2000]. También son de ayuda en el diseño de una arquitectura la experiencia de trabajos anteriores y las estructuras que podamos identificar como los patrones de la arquitectura.

Figura 4.1. Factores que influyen en la arquitectura

Si el sistema proporciona los casos de uso correctos, casos de uso de alto rendimiento, calidad y facilidad de utilización; los usuarios pueden emplearlo para llevar a cabo sus objetivos. Pero, la cuestión es: ¿cómo se puede conseguir ésto? La respuesta es construir una arquitectura que nos permita implementar los casos de uso propuestos de una forma económica, ahora y en el futuro.

Sin embargo, la arquitectura no sólo se ve condicionada por los casos de uso arquitectónicamente significativos, sino también por los siguientes factores (figura 4.1):

- Sobre qué productos software del sistema se quiere desarrollar, como sistemas operativos o sistemas de gestión de bases de datos concretos.

- Qué productos *middleware* (capa intermedia) se quieren utilizar. Por ejemplo, si se tiene que seleccionar un mecanismo para la conversión y envío de mensajes a objetos en entornos heterogéneos, o un marco de trabajo independiente de la plataforma, es decir, un subsistema "prefabricado".

- Qué sistemas heredados se quiere utilizar en el sistema.

- A qué estándares y políticas corporativas hay que adaptarse.

- Qué requisitos no funcionales generales se deben satisfacer, como requisitos de disponibilidad, tiempo de recuperación o uso de memoria.

- Cuales son las necesidades de distribución, que especifican cómo distribuir el sistema, quizá a través de una arquitectura cliente/servidor.

4.2.4 Utilización de patrones en la arquitectura

Las ideas del arquitecto Christopher Alexander [1977] sobre cómo los "lenguajes de patrones" se utilizan para sistematizar principios y prácticas importantes en el diseño de edificios y comunidades, han inspirado a muchos desarrolladores, como ya comentamos en el capítulo 2. Los patrones de las arquitecturas se utilizan de una forma parecida pero se centran en estructuras e interacciones de grano más grueso, entre subsistemas e incluso entre sistemas. Existen muchos patrones de arquitectura [Buschmann, 1996], pero vamos a tratar los más interesantes en el ámbito de las arquitecturas orientadas a objetos, por ser en las que se basa la propuesta de esta tesis.

El patrón *Broker* es un mecanismo genérico para la gestión de objetos distribuidos. Permite que los objetos hagan llamadas a otros objetos remotos a través de un gestor que redirige la llamada al nodo y al proceso que guardan al objeto deseado. Esta redirección se hace de manera transparente, lo cual quiere decir que el llamante no necesita saber si el objeto llamado es remoto.

El patrón *Layers* es aplicable a muchos tipos de sistemas. Este patrón define cómo organizar el modelo de diseño en capas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no "son conscientes" de ningún detalle o interfaz de las superiores. Además, ayuda a identificar qué puede reutilizarse, y proporciona una estructura que ayuda a tomar decisiones sobre qué partes conseguir de terceros y qué partes construir.

Hay otros patrones que ayudan a comprender el hardware de los sistemas a construir y que ayudan a diseñar un sistema software sobre él, como por ejemplo *Client/Server*, *Three-Tier* y *Peer-to-Peer*. Estos patrones definen una estructura para el modelo de despliegue y sugieren cómo se deben asignar los componentes a los nodos.

Como patrón arquitectónico también se podría considerar la estructuración de una arquitectura en capas. Una capa es un conjunto de subsistemas que comparten el mismo grado de generalidad y de volatilidad en las interfaces: las capas inferiores son de aplicación general a varias aplicaciones y deben poseer interfaces más estables, mientras que las capas más altas son más dependientes de la aplicación y pueden tener interfaces menos estables.

Un sistema con una arquitectura en capas sitúa a los subsistemas de aplicación individuales en la capa más alta. Estos se construyen a partir de subsistemas en las capas más bajas, como son los marcos de trabajo (*frameworks*) y las bibliotecas de clases. Lo podemos observar en la figura 4.2.

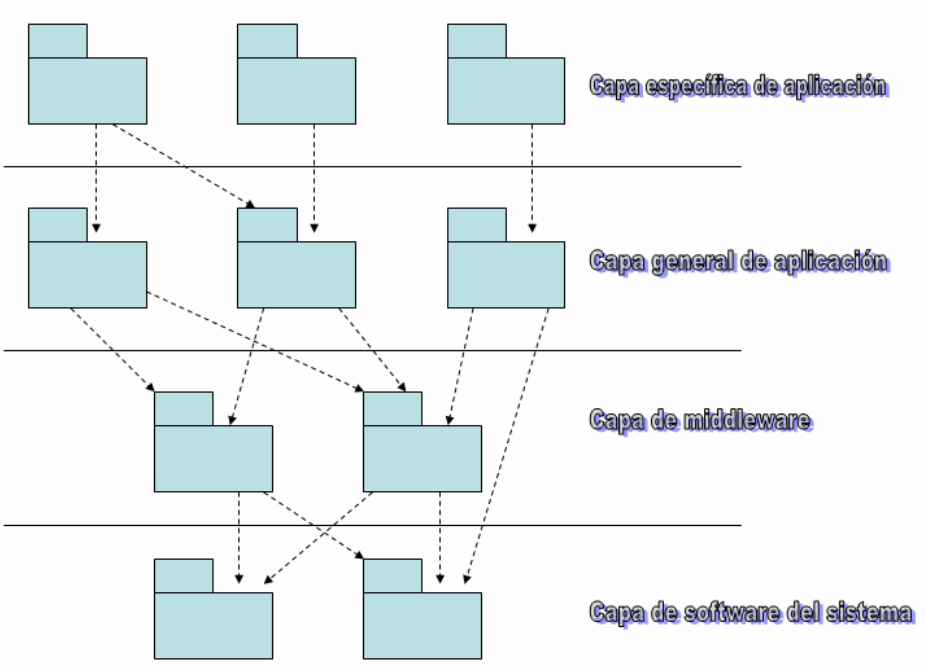

Figura 4.2. Arquitectura en capas

La capa general de aplicación contiene los subsistemas que no son específicos de una sola aplicación, sino que pueden ser reutilizados por muchas aplicaciones diferentes dentro del mismo dominio o negocio. La arquitectura de las dos capas inferiores puede establecerse sin considerar los requisitos o casos de uso debido a que no son dependientes del negocio. La arquitectura de las dos capas superiores se crea a partir de los casos de uso significativos para la arquitectura (estas capas son dependientes del negocio).

4.2.5 Conclusión

Tras esta breve introducción a las arquitecturas, y antes de plantear la propuesta arquitectural que se presenta en esta tesis es conveniente tratar de sintetizar las múltiples visiones hasta aquí descritas con la idea de presentar la arquitectura del sistema en forma multicapa o multinivel con los componentes que forman cada nivel, incluyendo:

1. Los **Niveles** o **capas** en los que se estructura la arquitectura del sistema
2. Los **Componentes** ó elementos estructurales

3. Las **Propiedades** visibles de estos componentes y su **Funcionalidad**
4. Las **Relaciones** y colaboración entre ellos

Con respecto a los objetivos de una arquitectura podemos resaltar lo siguiente:

1. La arquitectura debe establecer los objetivos que ha de cumplir para satisfacer los requisitos del sistema e integrarlos en el diseño de la misma.
2. La arquitectura debe identificar las partes del sistema, ordenándolo mediante su descomposición.
3. La arquitectura debe identificar las relaciones con sistemas externos.
4. La arquitectura debe utilizar estándares para uniformizar problemas y soluciones en cuanto a nomenclatura, estilo, planificación y procedimientos.
5. La arquitectura debe soportar los incrementos adicionales de capacidad del sistema.

Con respecto a sus características podemos resaltar:

1. La arquitectura debe representar todos los posibles casos de uso.
2. Debe ser reutilizable y exportable.
3. Debe tener un alto nivel de abstracción.
4. Debe permitir la interoperabilidad (posibilidad de utilización con distintas tecnologías, portabilidad a través de múltiples plataformas).
5. Debe ser heterogénea, facilitando la integración de componentes de distinto tipo.
6. Debe ser evolutiva: La arquitectura debe evolucionar y así poder reflejar los cambios de los sistemas.

4.3 FUNCIONALIDAD SOPORTADA POR LA ARQUITECTURA

Como se ha indicado en el apartado anterior, la descripción de una arquitectura debe hacerse desde diferentes puntos de vista [IEEE, 2000]. En este apartado se adoptará un enfoque funcional, para describir los requisitos funcionales generales que debe ofrecer un Sistema de Reutilización de Objetos de Aprendizaje (SROA) que se base en esta arquitectura; así como los actores que participarán en estos sistemas haciendo uso de la funcionalidad anterior. Por su importancia en los sistemas de e-learning, se dedicará un apartado independiente (4.4) para justificar la decisión de adoptar inicialmente un número limitado de metadatos para describir los objetos de aprendizaje gestionados por los sistemas basados en la arquitectura propuesta. Finalmente en los dos últimos apartados (4.5 y 4.6), se realizará una descripción de la arquitectura desde un punto de vista organizacional y de servicios, presentando las capas en que se ha estructurado la arquitectura de SROA.

Para describir la funcionalidad soportada por la arquitectura, en primer lugar realizaremos una presentación de los requisitos generales que debe satisfacer un SROA construido siguiendo la arquitectura propuesta. Después detallaremos cada requisito en un caso de uso donde se explicará con más detalle los pasos a seguir para completar con éxito la funcionalidad requerida. A continuación presentaremos el modelo del dominio del problema y por último los diagramas de interacción más importantes.

4.3.1 Especificación de requisitos generales

Como se ha observado en el apartado 4.2, una de las tareas que primero se deben realizar a la hora de definir la arquitectura de un sistema como SROA es establecer la especificación de sus requisitos. Mediante estos requisitos podremos conocer los procesos que se llevan a cabo en el dominio del problema y establecer los objetivos que debe cumplir la arquitectura.

A continuación se muestra una tabla con el resumen de requisitos funcionales generales, que tiene que cumplir la arquitectura:

REQUISITOS	DESCRIPCIÓN
R1	Realizar la búsqueda de contenidos
R2	Realizar la catalogación de contenidos
R3	Proporcionar los contenidos solicitados
R4	Realizar el intercambio de contenidos entre repositorios
R5	Realizar la publicación de contenidos
R6	Definir un conjunto de metadatos común, que se adapte a los estándares existentes como LOM, SCORM o IMS
R7	Realizar la conversión de metadatos para adaptarlos a la especificación del solicitante de contenidos
R8	Realizar un catálogo de los repositorios que forman parte de SROA
R9	Facilitar la integración con plataformas de aprendizaje
R10	Realizar el registro de usuarios
R11	Realizar la autenticación de usuarios
R12	Gestionar los derechos de los contenidos

R13	Registrar la trazabilidad de las acciones que se realizan en SROA

Tabla 4.1. Especificación de requisitos generales

4.3.2 Actores

Antes de describir cada uno de los requisitos en forma de casos de uso necesitamos detallar que actores utilizarán el sistema. Partiendo de la especificación de IMS DRI (*Digital Repository Interoperability*) [IMS, 2003a], explicada en el capítulo 2, se recuerda que existían roles predeterminados que podían tomar los usuarios de los repositorios digitales y que, por tanto, serán los mismos que interactúen con SROA; estos roles son: Bibliotecario, Contribuyente, Prestatarios, Usuarios Casuales, Administrador y Agentes Software.

El rol de **bibliotecario** lo tendrá el propio SROA que será el encargado de la catalogación y gestión de los repositorios y su contenido docente; por lo tanto, representa un rol interno de SROA.

El rol de **contribuyente** se da en SROA pero de una manera especial. Tal y como se explica en [IMS, 2003a] son aquellas personas que introducen los recursos (objetos de aprendizaje) en el repositorio y se encargan de la generación de metainformación. En nuestro caso es aquel que registra su repositorio en SROA y por tanto, de forma indirecta, da acceso a su contenido.

El rol de **prestatarios** también se da en SROA, ya que son aquellos que adquieren objetos de aprendizaje de los repositorios registrados en SROA de manera regular, y que suelen personalizar la interfaz con la que interactúan con el repositorio; así como preservar y dejar constancia de las búsquedas realizadas en las sesiones con el repositorio.

Los **usuarios casuales** también tienen cabida en SROA, ya que representan usuarios invitados que pueden tener permisos para buscar, explorar o descargar objetos del repositorio pero sin tener su espacio personalizado propio. Generalmente estos usuarios no tienen que estar registrados como usuarios habituales.

El rol de **administrador** estará representado en SROA, ya que tiene la responsabilidad de gestionar los usuarios del repositorio y de establecer la configuración de SROA.

El rol de **agente software** también se da, ya que representa cualquier sistema que puede consultar el repositorio y descargar contenidos de SROA.

4.3.3 Casos de uso

Para describir cada uno de los casos de uso correspondientes a los requisitos funcionales presentados anteriormente vamos a utilizar la siguiente plantilla, basada en la utilizada por IMS en DRI [IMS, 2003a] para describir sus casos de uso:

Funcionalidad	Nombre del caso de uso.
Objetivo	Descripción informal de los objetivos.
Actores	Actores que intervienen: principales y secundarios.
Precondiciones	Condiciones que deben cumplirse para que pueda llevarse a cabo.
Pasos (o Flujos Básicos)	Secuencia de pasos necesarios para que se desarrolle con éxito. Debemos mostrar las interacciones de los actores y las acciones de SROA.
Extensiones (o Flujos Alternativos)	Puntos de extensión.

Utilizando la plantilla anterior, pasamos a documentar cada una de las funcionalidades:

Funcionalidad	***Realizar la búsqueda de contenidos.***
Objetivo	Buscar contenidos educativos a través de diversos repositorios distribuidos.
Actores	Prestatarios, usuarios casuales, agentes software y bibliotecario.
Precondiciones	Detallar la metainformación de los contenidos a buscar.
Pasos	1. Especificar el estándar utilizado por el cliente (SCORM, IMS, etc.). 2. Rellenar un formulario con los campos educativos más significativos. 3. Creación de un documento XML con los campos definidos. 4. Realizar la búsqueda en todos los repositorios registrados en SROA (búsqueda federada). 5. Catalogar los contenidos resultado de la búsqueda. 6. Convertir el estándar de metadatos de los contenidos, si son distintos los del solicitante, de los recuperados de los repositorios. 7. Presentar los resultados de la búsqueda al solicitante.
Extensiones	Si se produce un error en el proceso de búsqueda se le comunicará al usuario. Si no se encuentra ningún contenido que coincida con los parámetros de búsqueda se le comunicará al usuario.

Funcionalidad	***Realizar la catalogación de contenidos.***
Objetivo	Catalogar los contenidos educativos de diversos repositorios distribuidos.
Actores	Bibliotecario.
Precondiciones	Haber realizado una búsqueda en la que existan resultados.
Pasos	1. Realizar la búsqueda en todos los repositorios registrados en SROA. 2. Descartar objetos de aprendizaje duplicados. 3. Descartar objetos de aprendizaje con bajo índice de coincidencia. 4. Clasificar los objetos de aprendizaje por índice de coincidencia.
Extensiones	Si se produce un error en el proceso de catalogación se le comunicará al usuario.

Funcionalidad	***Proporcionar los contenidos solicitados.***
Objetivo	Poder descargar los contenidos educativos de diversos repositorios distribuidos.
Actores	Prestatarios, usuarios casuales, agentes software y bibliotecario.
Precondiciones	Haber realizado una búsqueda en la que existan resultados.
Pasos	1. Realizar la búsqueda en todos los repositorios registrados en SROA. 2. Catalogar los contenidos resultado de la búsqueda. 3. Presentar al cliente los resultados de la búsqueda. 4. Descargar los objetos de aprendizaje.
Extensiones	Si se produce un error en el proceso de descarga se le comunicará al usuario.

Funcionalidad	***Realizar el intercambio de contenidos entre repositorios.***
Objetivo	Poder descargar los contenidos educativos de diversos repositorios distribuidos e integrarlos en un repositorio exterior.
Actores	Prestatarios, usuarios casuales, agentes software y bibliotecario.
Precondiciones	Haber realizado una búsqueda en la que existan resultados y haber descargado los contenidos.
Pasos	1. Realizar la búsqueda en todos los repositorios registrados en SROA. 2. Catalogar los contenidos resultado de la búsqueda. 3. Presentar al cliente los resultados de la búsqueda. 4. Descargar los objetos de aprendizaje. 5. Integrar los objetos de aprendizaje en un repositorio externo.
Extensiones	Ninguna.

Funcionalidad	***Realizar la publicación de contenidos.***
Objetivo	Poder registrar en SROA un repositorio.
Actores	Contribuyente y bibliotecario.
Precondiciones	Ser el propietario de un repositorio con contenidos educativos.
Pasos	1. Realizar un servicio Web que de acceso al repositorio. 2. Publicarlo en un registro UDDI para poder descubrirlo. 3. Registrar en SROA el repositorio, especificando los datos del servicio y de su localización.
Extensiones	Si se produce un error en el proceso de registro del repositorio se le comunicará al usuario.

Funcionalidad	***Definir un conjunto de metadatos común, que se adapte a los estándares existentes como LOM, SCORM o IMS.***
Objetivo	Definir un conjunto de campos de metainformación común a la gran mayoría de especificaciones adaptable y evolutivo que facilite la búsqueda de contenidos y su conversión de un estándar a otro.
Actores	Administrador.
Precondiciones	Ninguna.
Pasos	1. Determinar los campos de metainformación más utilizados e importantes de las diversas especificaciones. 2. Clasificarlos por categoría. 3. Crear un fichero XML con las categorías y campos seleccionados. 4. Permitir la modificación del fichero para futuras ampliaciones o especificaciones.
Extensiones	Ninguna.

Funcionalidad	***Realizar la conversión de metadatos para adaptarlos a la especificación del solicitante de contenidos.***
Objetivo	Permitir la conversión de los metadatos de un objeto de aprendizaje del estándar utilizado en el repositorio origen al necesitado por el cliente.
Actores	Bibliotecario.
Precondiciones	Ninguna.
Pasos	1. Determinar la especificación de metadatos que usa el cliente. 2. Determinar la especificación de metadatos de cada uno de los objetos de aprendizaje resultado de la búsqueda. 3. Si alguna especificación de los metadatos de los objetos de aprendizaje recuperados no coincide con la utilizada por el cliente, realizar la conversión.
Extensiones	Si se produce un error en el proceso de conversión se le comunicará al usuario.

Funcionalidad	***Realizar un catálogo de los repositorios que forman parte de SROA.***
Objetivo	Catalogar en SROA los repositorios registrados.
Actores	Bibliotecario.
Precondiciones	El repositorio debe estar registrado en SROA.
Pasos	1. Cuando se registra un nuevo repositorio en SROA se deben completar todos los datos necesarios para describir el repositorio, su servicio asociado y el UDDI donde está publicado, información utilizada posteriormente para su localización y clasificación. 2. Catalogar los servicios que dan acceso a los repositorios registrados en SROA. 3. Mantener actualizado el estado de los servicios (activados o desactivados).
Extensiones	En caso de fallo en la localización del servicio cambiar el estado del servicio.

Funcionalidad	***Facilitar la integración con plataformas de aprendizaje.***
Objetivo	Dar acceso a SROA desde una plataforma de aprendizaje.
Actores	Agentes software.
Precondiciones	Ser el administrador de una plataforma de aprendizaje.
Pasos	1. Conocer la URL de SROA. 2. Dar acceso a SROA desde la plataforma de aprendizaje. 3. Realizar búsquedas de contenidos o registrar el repositorio de la plataforma.
Extensiones	Ninguna.

Funcionalidad	***Realizar el registro de usuarios.***
Objetivo	Registro de los usuarios de SROA.
Actores	Prestatarios, contribuyente, agentes software y administrador.
Precondiciones	Los nombres de usuario deben ser únicos en SROA
Pasos	1. Pedir el nombre de usuario y la contraseña al cliente para su registro en SROA. 2. Comprobar el nombre de usuario para evitar duplicados. 3. Activar la cuenta de usuario con el perfil correspondiente. 4. Realizar acciones de administración básica: Altas, bajas y modificaciones.
Extensiones	Si ya existe un nombre de usuario en SROA igual al que se quiere dar de alta se le avisará del error.

Funcionalidad	***Realizar la autenticación de usuarios.***
Objetivo	Autenticación de los usuarios de SROA.
Actores	Prestatarios, contribuyente, agentes software y administrador.

Precondiciones	Para autenticarse el cliente tiene que estar registrado.
Pasos	1. Pedir el nombre de usuario y la contraseña al cliente. 2. Comprobar el nombre de usuario y la contraseña en SROA. 3. Cargar el perfil del usuario.
Extensiones	Si no existe un nombre de usuario en SROA igual al que se ha introducido se le avisará del error. Si la contraseña no es correcta para el nombre de usuario se le avisará del error. Se permitirá el acceso restringido a usuarios invitados.

Funcionalidad	***Gestionar los derechos de los contenidos***
Objetivo	Determinar que contenidos están sujetos a derechos de autor y si es necesario el pago por su uso.
Actores	Prestatarios, contribuyente, agentes software y bibliotecario.
Precondiciones	Los contenidos deben tener información sobre derechos.
Pasos	1. El contribuyente ha de determinar que contenidos de su repositorio están sujetos a derechos de autor. Esto se determinará con los metadatos correspondientes. 2. El bibliotecario cuando recupera los objetos de aprendizaje de los repositorios determina cuales están sujetos a derechos y cual es su coste. 3. Cuando se presenta la clasificación final de los resultados de una búsqueda se indicarán estos datos y si es necesario se cobrará por los contenidos.
Extensiones	Si un usuario tiene que pagar por el uso del contenido no se permitirá la descarga del mismo hasta que no se hayan activado los mecanismos de pago determinados.

Funcionalidad	***Registrar la trazabilidad de las acciones que se realizan en SROA***
Objetivo	Guardar un archivo de "*log*" con todas las acciones que se desencadenan cuando un cliente realiza una acción en SROA
Actores	Todos.
Precondiciones	Realizar alguna acción en SROA
Pasos	1. Si no existe el fichero de *log* se debe crear. 2. Si existe el fichero de *log* ir añadiendo anotaciones. 3. Ir anotando los resultados de todas las operaciones que se van realizando en SROA. 4. Realizar acciones de administración básica: Búsqueda, borrado, etc.
Extensiones	Ninguna.

En la figura 4.3 se representa el diagrama UML de los casos de uso generales y todos los actores implicados. Las figuras 4.4 y 4.5 muestran en detalle los dos casos de uso más importantes, ya que como se señala en [Larman, 2001] lo importante de los casos de uso no son los diagramas sino la explicación textual de los mismos.

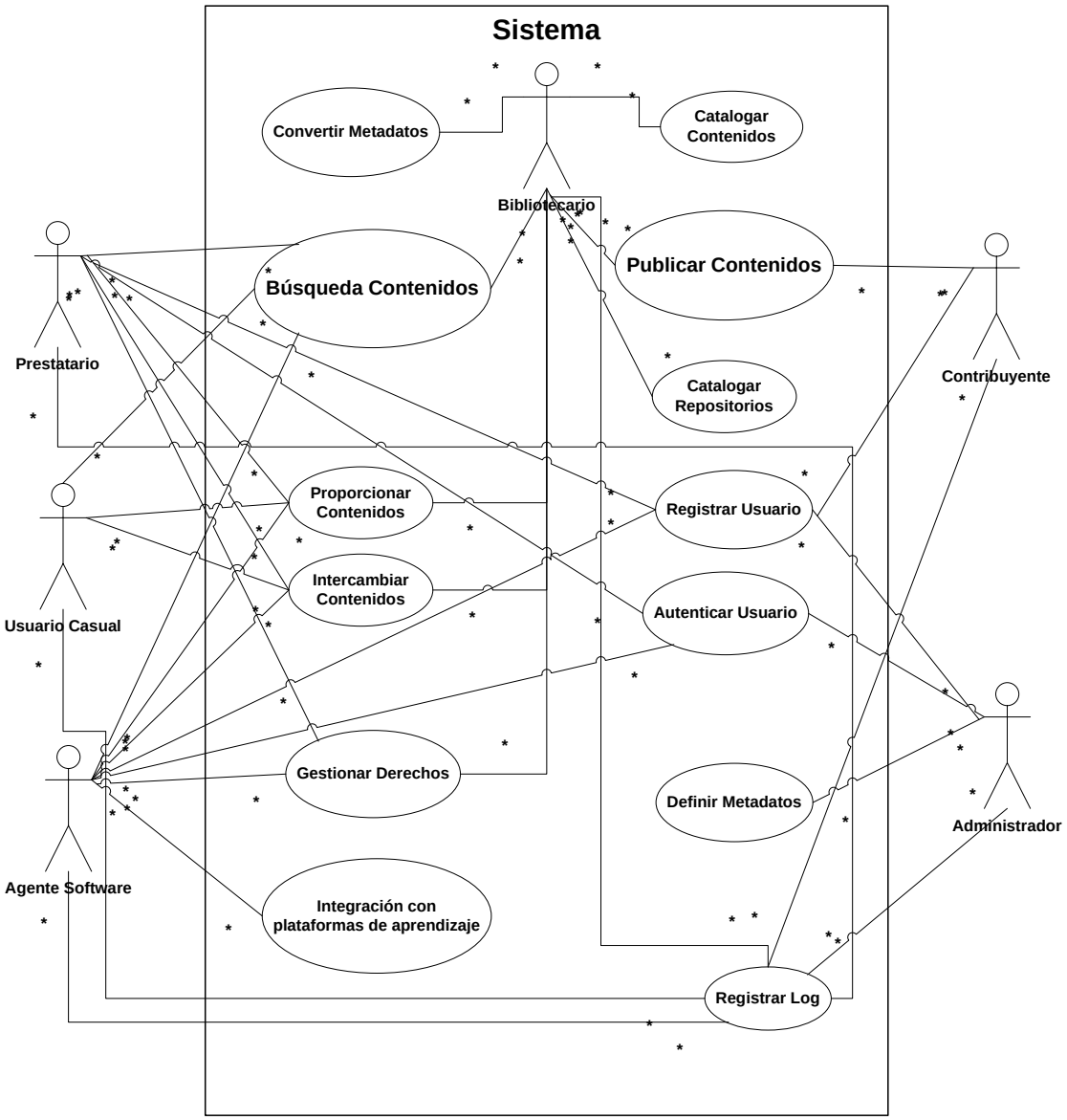

Figura 4.3. Diagrama de casos de uso generales

El primer caso de uso más importante (figura 4.4) es el de la búsqueda de contenidos en SROA, correspondiente al requisito R1. Esta acción desencadena una búsqueda federada en diversos repositorios distribuidos a través de un servicio asociado a cada uno de los repositorios que previamente se han registrado en SROA.

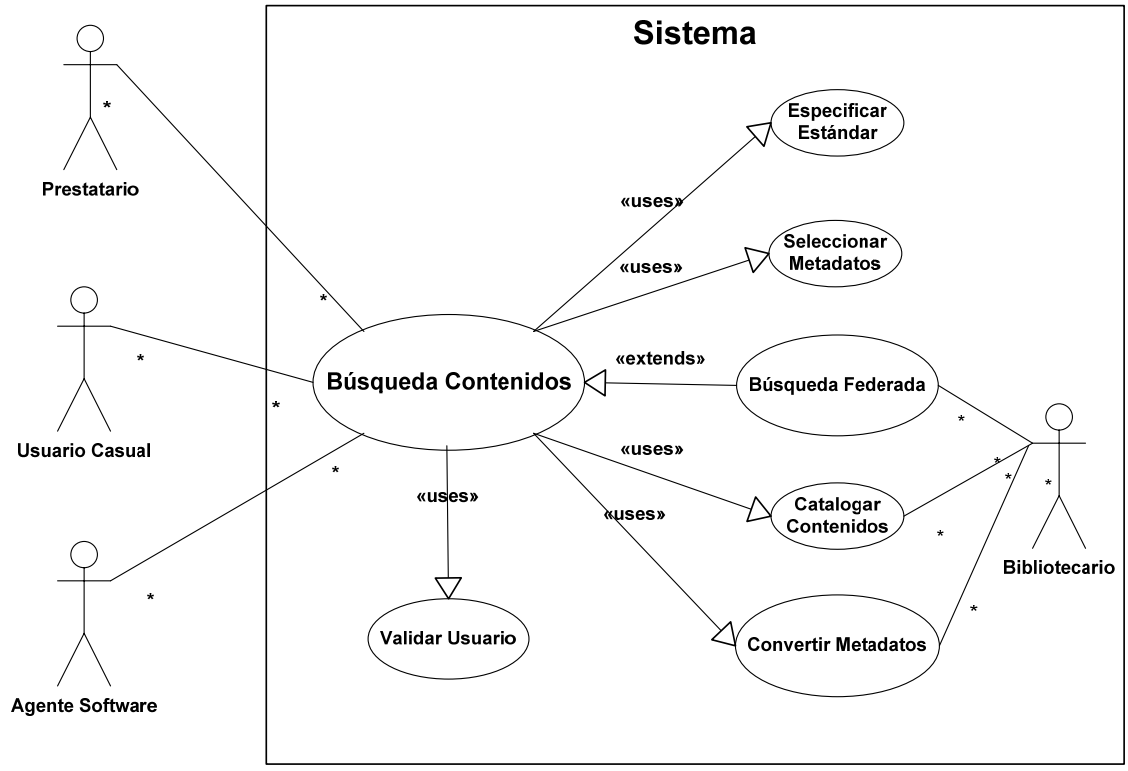

Figura 4.4. Caso de uso: Búsqueda de contenidos

El siguiente caso de uso (figura 4.5) es el que describe cómo se pueden publicar contenidos en SROA, correspondiente al requisito R5. Para ello lo que se necesita es que el Contribuyente registre su repositorio en SROA, donde quedará catalogado. Los pasos previos para poder hacerlo, es haber desarrollado un servicio Web que de acceso al mismo y haberlo publicado en un registro UDDI para poder localizarlo y utilizarlo.

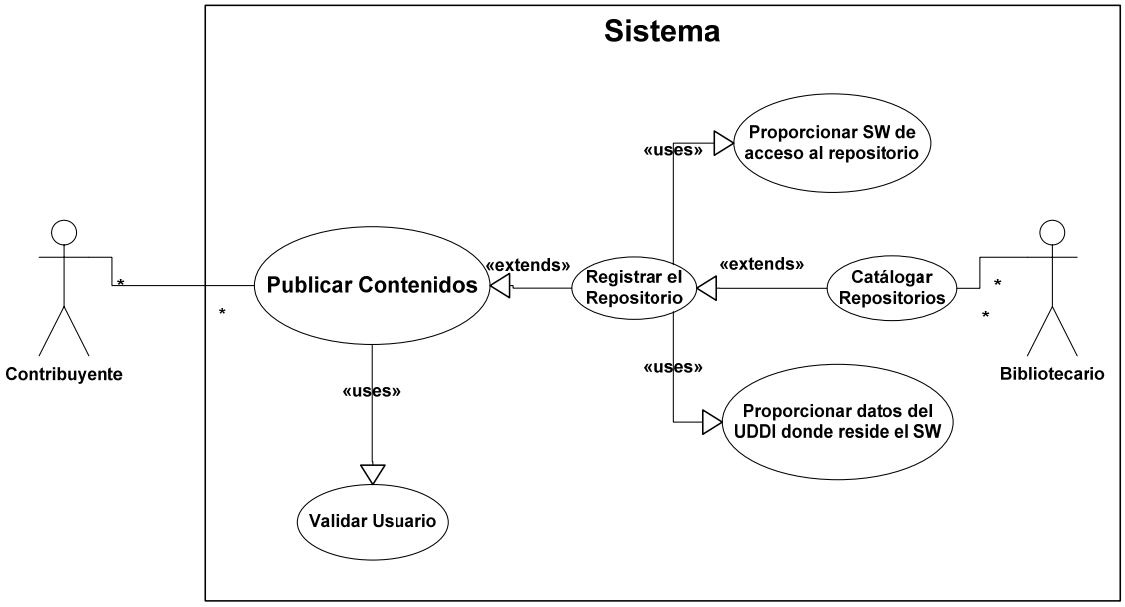

Figura 4.5. Caso de uso: Publicación de contenidos

4.3.4 Modelo del Dominio

El modelo del dominio del problema muestra los conceptos más importantes representados en forma de clases del dominio. En nuestro caso el problema a resolver consiste en la reutilización de objetos de aprendizaje que residen en repositorios distribuidos haciendo interoperables estos repositorios.

Como se aprecia en la figura 4.6 se ha considerado que el Sistema de Reutilización de Objetos de Aprendizaje (SROA) está compuesto por un conjunto de repositorios que relaciona y un conjunto de usuarios del sistema. Estos repositorios contienen objetos de aprendizaje que a su vez están compuestos por contenidos docentes y metadatos que describen estos contenidos utilizando una especificación determinada como LOM, SCORM o IMS.

Las operaciones principales que se pueden llevar a cabo en SROA son las de realizar búsquedas y publicar contenidos. La operación de búsqueda desencadena una serie de acciones detalladas en los casos de uso, cuya principal característica es la de realizar una búsqueda federada en diversos repositorios. La operación de publicación de contenidos en SROA consiste en registrar y catalogar un repositorio y, por lo tanto, dar acceso a los objetos de aprendizaje que contiene.

Como se puede apreciar en el diagrama cada repositorio tendrá sus propias operaciones de búsqueda y publicación. Por lo tanto cuando SROA realiza una búsqueda en un repositorio realmente se está llamando a la operación de búsqueda interna del repositorio en cuestión.

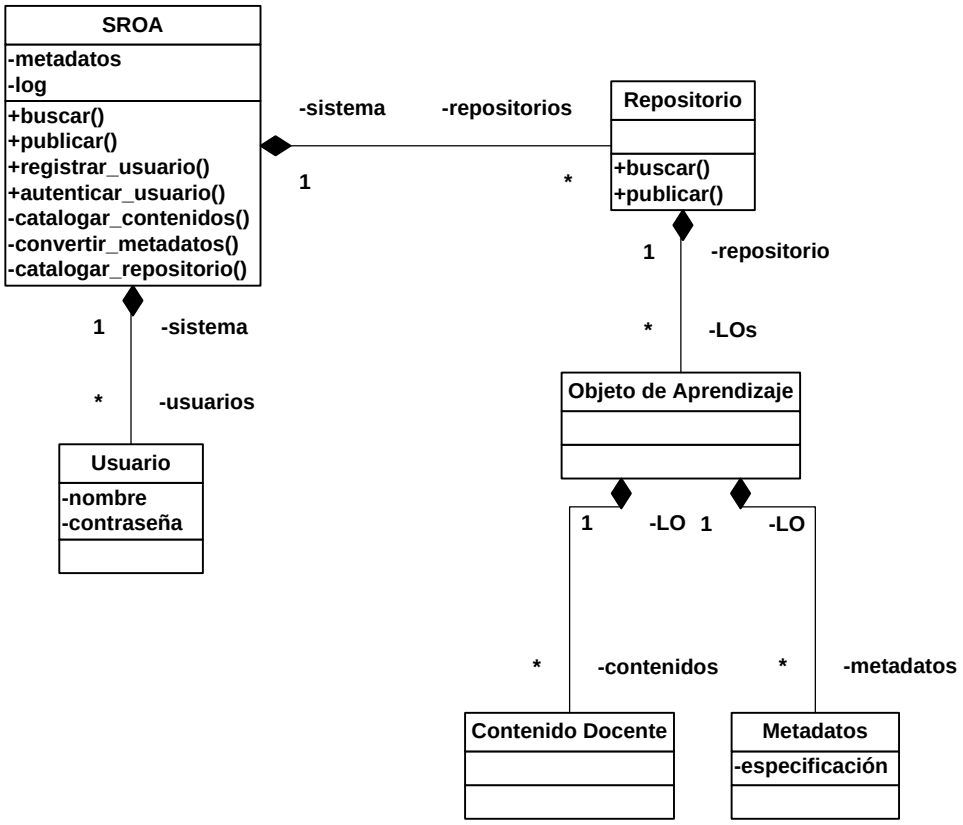

Figura 4.6. Modelo del dominio del problema

4.3.5 Diagramas de interacción

Para completar el estudio de la funcionalidad que ha de ser soportada por la arquitectura propuesta, presentaremos los diagramas de interacción (a través de diagramas de secuencia) que corresponden con las acciones más importantes en SROA que son la de búsqueda y publicación de contenidos.

En la figura 4.7 se muestra el diagrama de secuencia para la búsqueda de contenidos. En este caso el usuario lo primero que realiza es la autenticación en SROA (se supone que es un usuario registrado) y, a continuación, le manda a SROA un mensaje para que active la operación de búsqueda con la información sobre los metadatos que le interesan y la especificación que utiliza. SROA redirige la búsqueda hacia los repositorios distribuidos que le devolverán los objetos de aprendizaje coincidentes con los metadatos seleccionados. Una vez que SROA ha recibido los objetos de aprendizaje realizará su catalogación y conversión en la especificación de metadatos esperada por el usuario, y por último le devolverá los objetos de aprendizaje resultado de la búsqueda.

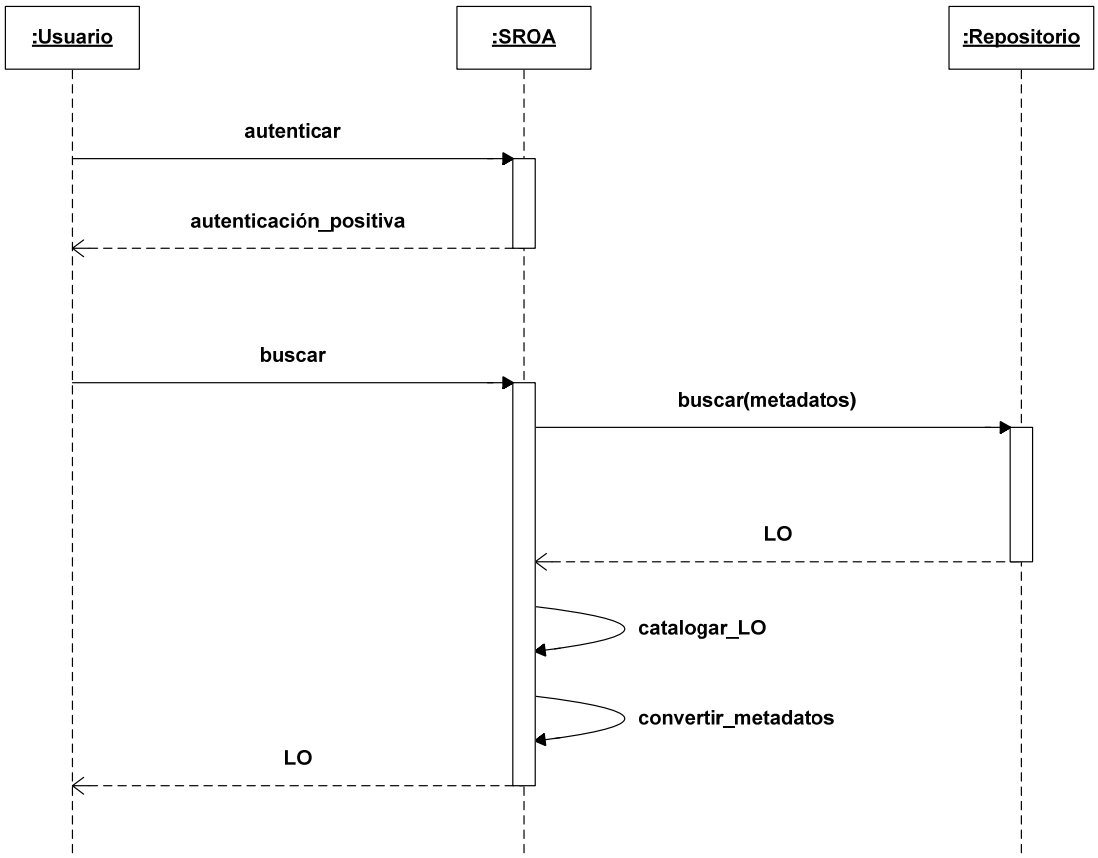

Figura 4.7. Diagrama de Secuencia: Búsqueda de contenidos

En la figura 4.8 se muestra el diagrama de secuencia para la publicación de contenidos. Como se comentó en los casos de uso, la publicación de contenidos conlleva el registro en SROA de un repositorio para dar acceso al mismo. Lo primero que debe hacer el usuario es realizar la autenticación en SROA (se supone que es un usuario registrado), y a continuación enviar a SROA un mensaje para que active la operación de publicación con la información sobre el repositorio que quiere publicar. SROA realizará una comprobación de localización positiva del repositorio y a continuación lo catalogará dentro del mismo, con todos sus datos, dando una respuesta positiva al usuario si todo ha sido correcto.

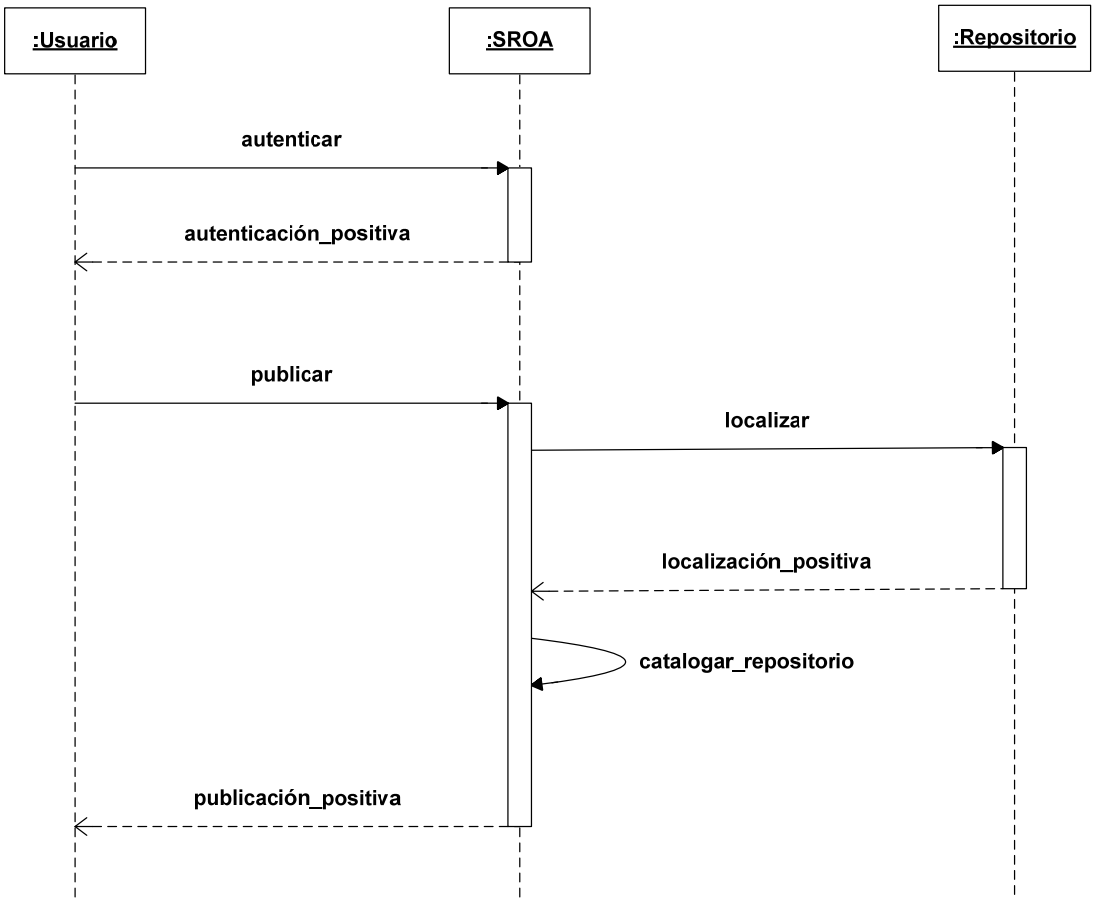

Figura 4.8. Diagrama de Secuencia: Publicación de contenidos

4.4 CAMPOS EDUCATIVOS SELECCIONADOS

En la actualidad existen una gran variedad de estándares educativos que se podrían haber utilizado para el desarrollo de la arquitectura que se está presentando. Sin embargo, el objetivo no era que fuera compatible con todos los estándares que se están utilizando actualmente, sino totalmente adaptable a cualquier estándar actual o que pudiera aparecer en el futuro.

En este caso, la arquitectura que se presenta, es una arquitectura "plana", es decir, una arquitectura que no está ligada a ningún estándar para la descripción de objetos de aprendizaje. Pero que, sin embargo, puede operar con cualquier conjunto de campos definidos por el usuario; es más, el usuario decidirá en todo momento, como veremos a continuación, el conjunto de campos educativos con los que desea realizar la búsqueda, pudiendo modificarlos antes de cada solicitud de búsqueda, para que, de esta manera, pueda obtener diferentes resultados. Esta característica aporta a los sistemas basados en

la arquitectura una gran ventaja, ya que como se ha visto en el análisis de los distintos repositorios existentes actualmente, cada uno estaba ligado a alguna especificación de metadatos.

Lógicamente, aunque se presente una arquitectura totalmente independiente de cualquier estándar, se ha definido una serie de campos educativos que podríamos considerar "por defecto", con los que poder empezar a trabajar. Estos campos educativos se han obtenido del estándar LOM [IEEE, 2002], ya que es el estándar que mayor índice de aceptación tiene hoy en día, y en el que están basados la gran mayoría de especificaciones.

Cuando se concibió esta arquitectura, se pensó que el conjunto de campos educativos con el que se iba a trabajar, no debería ser muy extenso, ya que ésto dificultaría las futuras búsquedas. La gran mayoría de creadores de objetos de aprendizaje (empresas, usuarios, organizaciones, etc.) no tiene tendencia a describir completamente sus objetos de aprendizaje, sobre todo si el conjunto de campos educativos es muy extenso. Este es uno de los factores principales por los que se decidió que los campos educativos con los que se iba a trabajar, tendrían que ser solamente los más utilizados y representativos.

El conjunto de campos educativos de LOM es demasiado extenso para poder utilizarlos todos, así que se ha realizado una selección de los más significativos. Esta selección se ha basado en el estudio realizado por Friesen [2004] miembro del comité JTC1 SC36 de la ISO (comentado en el capítulo 2). Para ello, se han estudiado cuáles son los campos que aparecen con mayor frecuencia en las descripciones de objetos de aprendizaje, con esto sabremos cuáles son los más representativos para los creadores de estos objetos.

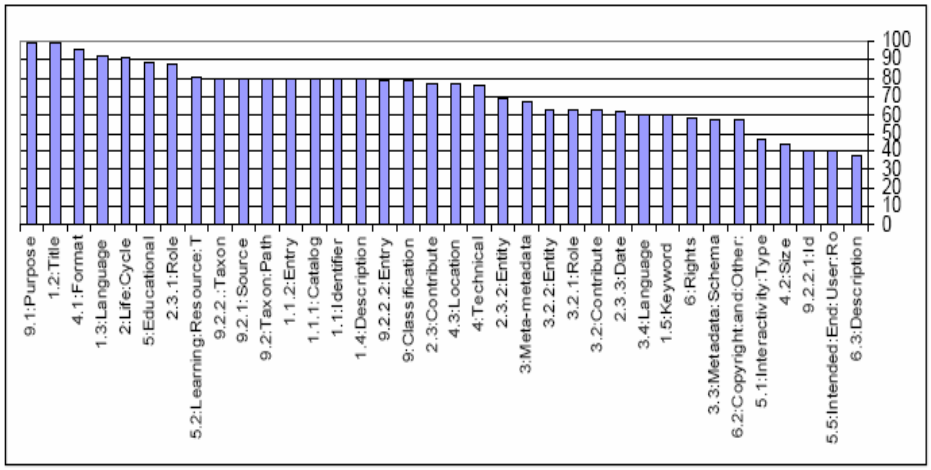

Figura 4.9. Elementos LOM más utilizados

En la figura 4.9 se muestra el resultado del estudio. En ella aparecen los campos que más se utilizan actualmente en la descripción de la metainformación de los objetos de

aprendizaje. Los dígitos que identifican a los elementos LOM en el diagrama, hacen referencia las categorías de metadatos establecidas en LOM de forma jerárquica.

Si analizamos el gráfico podemos comprobar que los campos de la primera de las categorías que conforman LOM, la categoría *General*, se utilizan aproximadamente un 80%, el conjunto de los elementos que conforman la categoría *Life Cycle*, se utilizan aproximadamente entre un 60% y un 80%; sin embargo, en el gráfico no aparece ningún elemento de la categoría *Annotation*.

A partir de este estudio, se han podido obtener ciertas referencias sobre el índice de importancia que tiene cada uno de estos campos. Después se han analizado los porcentajes de utilización de cada uno de estos campos, ordenados por sus categorías, para poder ver realmente el porcentaje de utilización específico de cada uno de ellos y quedarnos con los más representativos. Aún así, aunque se seleccione una serie de atributos educativos de un estándar que se considera de referencia en el ámbito e-leaning, como es LOM, la arquitectura presentada, como ya se ha indicado, no estará ligada a ningún estándar ni conjunto de campos educativos en concreto, sino que será totalmente adaptable a diferentes estándares existentes hoy en día o en el futuro.

Después de repasar los resultados del estudio, y sabiendo que si se seleccionaba un número elevado de campos educativos los usuarios verían más dificultades a la hora de encontrar coincidencias (ya que si los autores de estos objetos educativos no describen completamente sus objetos, obviamente, será más probable que el usuario encuentre más problemas a la hora de realizar las búsquedas), se decidió contar con solamente trece de los campos educativos que establece LOM. No es un número muy elevado de campos, por tanto, los usuarios no "tardarían" mucho tiempo en describir sus objetos y, por lo tanto, las operaciones de búsqueda tendrán un mayor índice de probabilidades de éxito; pero, además, es un número lo suficientemente elevado de campos como para poder describir perfectamente el objeto.

A continuación, se mostrarán cuales han sido los elementos seleccionados. Para ello, se utilizará un fichero XSD, en el que se mostrarán los diferentes campos educativos de SROA, con sus respectivas restricciones de valores en aquellos casos en los que interese. A partir de este fichero XSD, SROA generará el formulario dinámico de recogida de datos del usuario a la hora de realizar una búsqueda, y permitirá saber cuales son los campos a buscar en cada uno de los objetos de aprendizaje de los diferentes repositorios. Cuando se complete este formulario con la información de los campos educativos, se generará un fichero XML cuya extensión se ha denominada XEL (*eXtensible E-learning Language*) y que servirá de base para las búsquedas.

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="general">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title"/>
      <xs:element name="language" type="language_restriction" />
      <xs:element name="description" />
      <xs:element name="keyword" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="technical">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="format" type="format_restriction" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="educational">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="learningresourcetype"
                type="learningresourcetype_restriction" />
      <xs:element name="difficulty" type="difficulty_restriction" />
      <xs:element name="description" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="rights">
  <xs:complexType >
    <xs:sequence>
      <xs:element name="cost" type="yes_no_restriction" />
      <xs:element name="copyrightandotherrestrictions"
                type="yes_no_restriction" />
      <xs:element name="description" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="annotation">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="person" />
      <xs:element name="date" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Restrictions -->

<xs:simpleType name="String_length_restriction" >
  <xs:restriction base="xs:string">
```

```
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="yes_no_restriction" >
  <xs:restriction base="xs:string">
    <xs:enumeration  value="yes"/>
    <xs:enumeration  value="no"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="difficulty_restriction" >
  <xs:restriction base="xs:string">
    <xs:enumeration  value="very easy"/>
    <xs:enumeration  value="easy"/>
    <xs:enumeration  value="medium"/>
    <xs:enumeration  value="difficult"/>
    <xs:enumeration  value="very difficult"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="learningresourcetype_restriction" >
  <xs:restriction base="xs:string">
    <xs:enumeration  value="Diagram"/>
    ……
    <xs:enumeration  value="Table"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="format_restriction" >
  <xs:restriction base="xs:string">
    <xs:enumeration  value="application/excel"/>
    ……
    <xs:enumeration  value="video/x-mpeg"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="language_restriction" >
  <xs:restriction base="xs:string">
    <xs:enumeration  value="AD"/>
    ……
    <xs:enumeration  value="ZW"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>
```

Listado 4.1. Fichero XSD de campos educativos

Como se puede observar en el listado 4.1, se han seleccionado las principales categorías. No sólo se han tenido en cuenta los elementos más utilizados, sino también el sentido común, para generar un conjunto de campos educativos lo más equilibrado posible, que permita identificar un objeto de aprendizaje correctamente, pero sin que esto suponga contar con un conjunto de campos educativos excesivo.

A continuación, describiremos el significado de cada uno de estos campos educativos asociados a un objeto de aprendizaje:

- *General*: Esta categoría agrupa la información general que describe al objeto de aprendizaje en su conjunto.

  - *Title*: El nombre asignado al objeto de aprendizaje.

  - *Language*: El idioma o idiomas humanos predominantes en el objeto de aprendizaje para la comunicación con el usuario. Una herramienta de indexado o catalogación podría proporcionar un valor por defecto. Si el objeto de aprendizaje no tuviese contenido escrito en ningún idioma (como en el caso de un cuadro, por ejemplo), entonces el valor apropiado para este elemento sería "ninguno". Este elemento de datos se refiere al idioma del objeto educativo.

  - *Description*: Una descripción textual del contenido de el objeto de aprendizaje. Esta descripción no tiene porqué estar en el idioma y términos adecuados para los usuarios del objeto educativo descrito. La descripción debería estar en el idioma y términos apropiados para aquellos que deciden si el objeto educativo descrito es apropiado y relevante para los usuarios.

  - *Keyword*: Una palabra clave o frase que describe el tema principal del objeto de aprendizaje. Este elemento de datos no debiera ser utilizado para aquellas características que pueden ser descritas con otros elementos.

- *Technical*: Esta categoría describe los requisitos y características técnicas del objeto de aprendizaje.

  - *Format*: El(los) tipo(s) de datos de (todos los componentes) el objeto de aprendizaje. Este elemento de datos debe ser utilizado para identificar el software necesario para acceder al objeto educativo.

- *Educational*: Esta categoría describe las características educativas o pedagógicas fundamentales del objeto de aprendizaje. Esta es la información pedagógica esencial para aquellos involucrados en la consecución de una experiencia educativa de calidad. Entre los destinatarios de esta categoría se encuentran a profesores, administradores, autores y estudiantes.

  - *Learningresourcetype*: El tipo específico de recurso educativo. El tipo predominante debe aparecer en primer lugar. Los términos del vocabulario han sido definidos a partir del OED (*Oxford English Dictionary*) [OED, 1989] y de su utilización práctica en comunidades educativas.

- *Difficulty*: Este elemento describe lo difícil que resulta, para los destinatarios típicos, trabajar con y utilizar el objeto de aprendizaje.

- *Description*: Comentarios sobre cómo debe utilizarse el objeto de aprendizaje.

- *Annotation*: Esta categoría proporciona comentarios sobre la utilización pedagógica del objeto de aprendizaje, e información sobre quién creó el comentario y cuando fue creado. Esta categoría permite a los educadores compartir sus valoraciones sobre el objeto de aprendizaje, recomendaciones para su utilización, etc.

  - *Person*: La entidad (persona u organización) que creó la anotación.

  - *Date*: La fecha en la que se creó la anotación.

4.5 ORGANIZACIÓN EN CAPAS DE LA ARQUITECTURA PROPUESTA

En este apartado se expondrá la arquitectura propuesta que resolverá los problemas planteados en el capítulo anterior y dará soporte a la funcionalidad descrita en los apartados previos de este capítulo. Para su construcción se han seguido principalmente las especificaciones de IMS que se presentaron en el segundo capítulo, denominadas "*Digital Repository Interoperability*" (DRI) [IMS, 2003a] y "*Abstract Framework*" [IMS, 2003b].

Como se ha indicado anteriormente, un repositorio o almacén digital de recursos educativos es una colección de recursos (objetos de aprendizaje) que son accesibles a través de una red de comunicaciones. No se necesita un conocimiento previo de la estructura de la colección, la cual puede contener los propios recursos o únicamente los metadatos que los describen, junto con una referencia para su localización. Como se explica en "*IMS Digital Repository Interoperability*", el objetivo de un repositorio es favorecer la reutilización de recursos educativos, facilitando el acceso a los recursos almacenados desde:

1. Plataformas de gestión del aprendizaje (LMS: *Learning Management System*).
2. Sistemas de gestión de contenidos educativos (LCMS: *Learning Content Management System*).
3. Portales de búsqueda de contenidos (por ejemplo, sistemas de búsqueda de bibliotecas digitales, buscadores Web, etc.).

4. Cualquier aplicación o agente software desarrollado para acceder a este tipo de información.

En la figura 4.10 se muestra el modelo general de referencia de IMS [2003a] para el acceso a diferentes repositorios por parte de varios tipos de usuarios, como los creadores de los recursos educativos ("*Creator*") que se almacenarán en ellos, los alumnos que los utilizarán ("*Learner*"), o, en general, cualquier persona que precise la búsqueda de información en los repositorios ("*Infoseek*"). Según este diagrama, la búsqueda en un repositorio puede realizarse mediante los mecanismos de acceso ofrecidos por los propios repositorios, basados en diferentes tecnologías y lenguajes de consulta (por ejemplo, SQL, Z39.50, XML-XQuery, etc.), o a través de sistemas intermediarios.

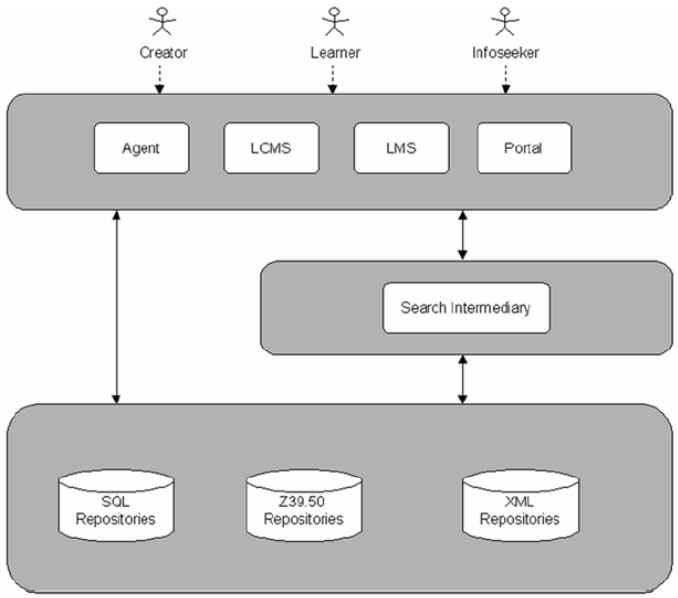

Figura 4.10. Modelo de acceso a repositorios de IMS

Por lo tanto, la arquitectura que se presenta en la tesis serviría para la construcción de un sistema que tomaría el papel del sistema intermediario del modelo anterior (figura 4.10) y daría acceso a diferentes repositorios independientemente de la tecnología utilizada para su construcción. Para su construcción se han utilizado los servicios Web y la arquitectura SOA, ya que constituyen una tecnología muy adecuada para la implementación de repositorios que gestionen objetos de aprendizaje, ubicados en diferentes almacenes de recursos didácticos, ya que permite ofrecer, a través de una única interfaz, un acceso transparente a objetos distribuidos en repositorios basados en diferentes tecnologías de almacenamiento y de metadatos.

Siguiendo el ejemplo de la especificación de "*IMS Abstract Framework*" [IMS, 2003b] y el patrón de diseño "*Layers*" [Buschmann, 1996], se propone una arquitectura multinivel o multicapa basada en servicios Web y SOA para la implementación de este tipo de sistemas, considerando las cuatro capas generales representadas en la figura 4.11.

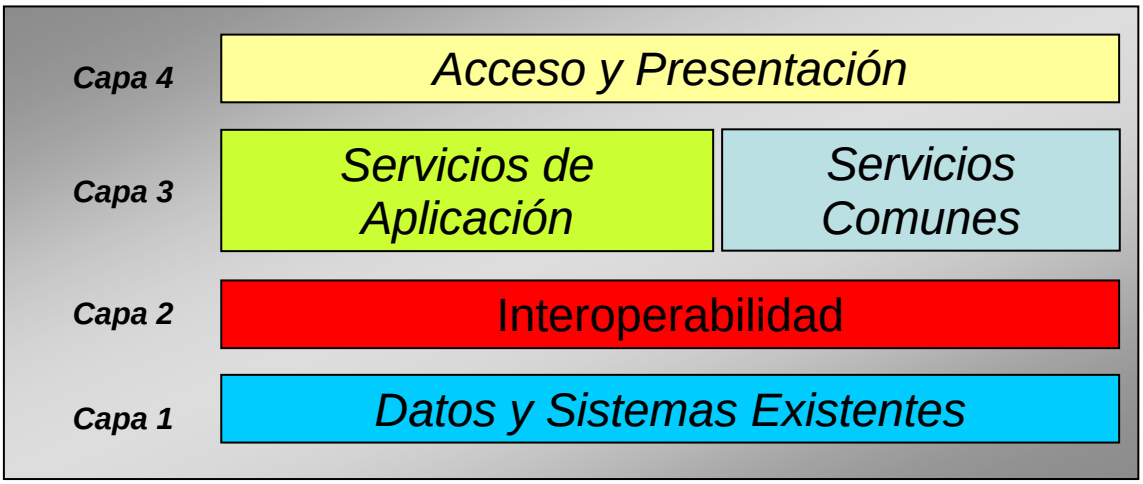

Figura 4.11. Capas generales de la arquitectura

La arquitectura se descompone en las siguientes capas:

- **Capa de Interfaz (o Acceso y Presentación)**: Esta es la parte de un sistema a través de la cual los usuarios pueden interactuar con el sistema, y también existen los componentes necesarios para que otros sistemas puedan interactuar con dicho sistema, y las herramientas de usuario necesarias para que éste pueda desarrollar las funciones propuestas.

- **Capa de Servicios**: En esta capa es donde se encuentran los servicios de aplicación que proporcionan los servicios específicos de funcionalidades e-learning y los servicios comunes usados por los servicios de aplicación.

- **Capa de Interoperabilidad**: Esta capa se puede descomponer a su vez en dos capas más:

  - **Capa de Integración**: En esta capa el sistema tiene los mecanismos y los flujos de información necesarios para recibir las peticiones tanto de los usuarios como de otros sistemas, procesarlas y actuar en la manera más adecuada. Esta capa se puede asemejar a lo que se llama ESB (*Enterprise Service Bus*) [ESB, 2006], y se basará en BPEL [2006], término que ya describimos en el capítulo anterior.

  - **Capa de Directorio de Servicios**: Al igual que los servicios de directorio tradicionales permiten almacenar información acerca de recursos en la red, en esta capa es donde se podrán buscar los servicios necesarios para el acceso a los repositorios distribuidos.

- **Capa de datos y sistemas existentes**: En esta capa se encuentran los sistemas externos a los cuales se quiere acceder. En nuestro caso, se trata de los repositorios distribuidos donde se encuentran los objetos de aprendizaje.

En la figura 4.12, se representa la arquitectura de una forma más detallada. La arquitectura definida es multicapa (o multinivel), compuesta por 4 capas, cada una de las cuales tiene un papel determinante para el correcto funcionamiento del repositorio distribuido universal que conjuntamente consiguen implementar.

En los siguientes apartados se describirán con detalle los servicios más importantes que forman cada capa.

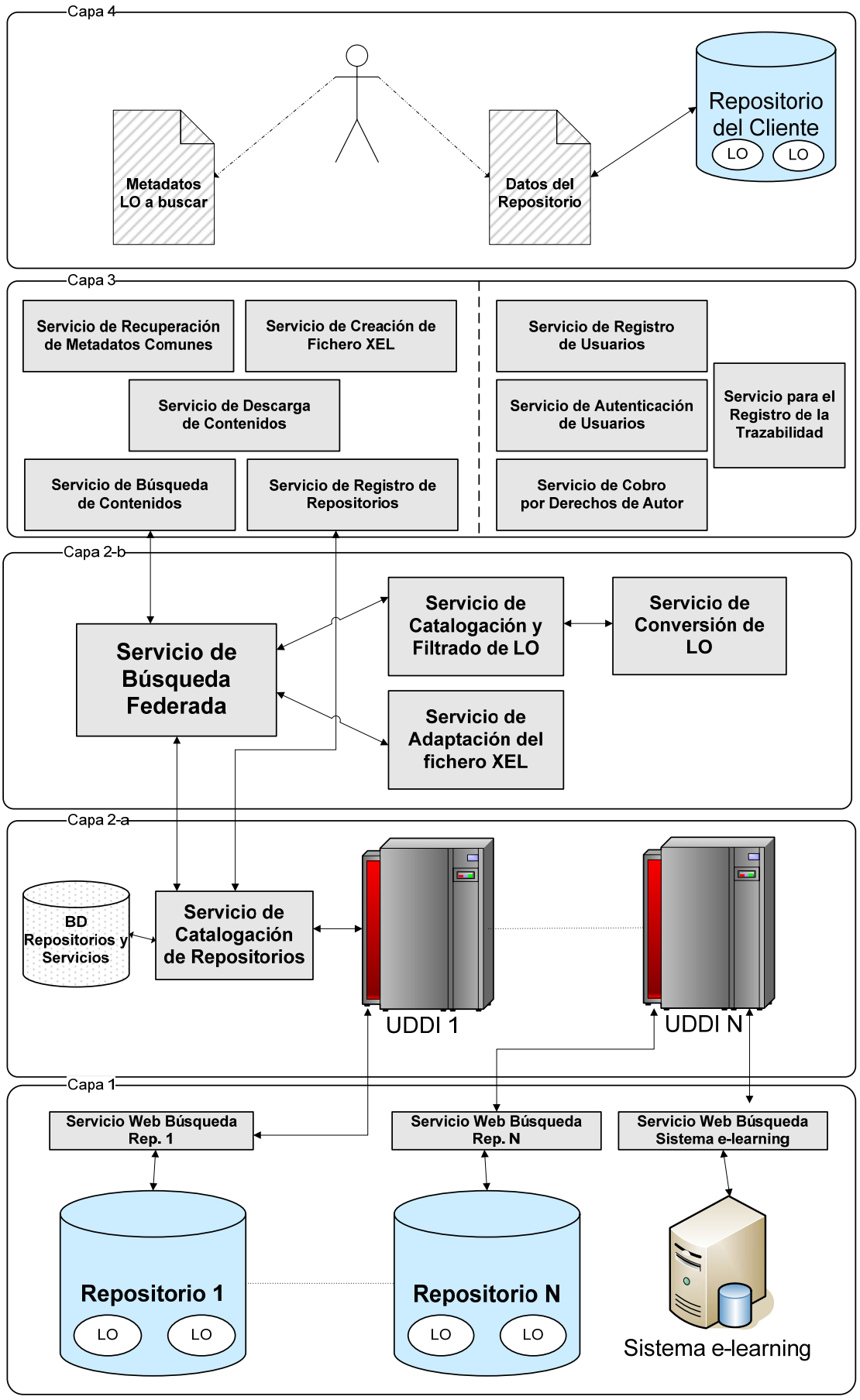

Figura 4.12. Arquitectura propuesta

4.5.1 Capa 1: Datos y Sistemas Existentes

El objetivo de la arquitectura propuesta es hacer transparente el acceso a múltiples repositorios de objetos de aprendizaje (LO), por lo que el nivel más bajo estaría compuesto precisamente por todos estos repositorios que contienen los objetos.

La orientación a servicios se pone ya de manifiesto en este nivel para ocultar a las capas superiores los detalles de implementación de cada repositorio (como determina IMS), existiendo, al menos, un servicio Web asociado a cada repositorio, encargado de gestionar el acceso al mismo. Este servicio debería recibir un fichero XEL con la información de metadatos proporcionada por el usuario y conforme al estándar de metadatos utilizado en los LO de su repositorio, y así poder realizar las búsquedas pertinentes. Para que dicho fichero XEL esté adaptado a la especificación utilizada por cada repositorio, SROA contará con un servicio específico de adaptación, que será utilizado por el servicio de búsqueda federada, como se detallará posteriormente. Este servicio asociado al repositorio, por lo tanto, realizará llamadas a métodos de búsqueda internos del repositorio y cuyo resultado serán los LO que contengan metadatos similares a los recibidos a través del fichero XEL.

Se debe puntualizar que los LO recuperados, en principio, tendrán el formato de metadatos estándar con el que trabaje el repositorio (IMS, SCORM, etc.), correspondiendo a las capas superiores de la arquitectura la función de conversión necesaria para su adaptación al formato deseado por el usuario. No obstante, en algunos casos, cuando los repositorios se hayan desarrollado siguiendo las recomendaciones de la especificación IMS DRI, existirá la posibilidad de que la conversión indicada se realice en esta capa, por parte del propio repositorio, como sugiere DRI [IMS, 2003a].

El servicio Web asociado a cada repositorio, se debería implementar de forma que recibe un fichero XEL, con los campos y valores establecidos por el usuario; el servicio, a partir de esta información, se encargará de buscar en el repositorio asociado, todos aquellos objetos de aprendizaje cuyos metadatos coincidan con los del fichero. Como ya hemos comentado, SROA sería capaz de adaptarse a cualquier conjunto de campos educativos, por lo tanto, será el usuario el que nos indique qué tipo de campos desea que se busquen en los ficheros de metainformación de los objetos de aprendizaje, a través del fichero XSD anteriormente comentado.

En este fichero XSD, el usuario especificará el conjunto de campos, agrupados en categorías, pudiendo incorporarle determinadas restricciones a cada uno de estos campos, y SROA le mostrará (como ya veremos en la capa superior de la arquitectura) un formulario totalmente dinámico, que contendrá dichos campos, para que este último, incorpore las características que deberían tener los objetos de aprendizaje que desea obtener. Por lo tanto, de esta manera, si un usuario cualquiera desea buscar en los

diferentes repositorios especificados, objetos de aprendizaje cuya metainformación haya sido descrita utilizando otros estándares, simplemente modificando este fichero, obtendrá un nuevo formulario y unos nuevos resultados a su petición, sin que este cambio afecte a ni una sola línea de código del SROA desarrollado.

Una de las ventajas que tiene esta arquitectura, es que es totalmente adaptable a cualquier estándar actual o futuro, y por lo tanto, totalmente compatible con cualquier repositorio de objetos de aprendizaje. Además, como ya se ha indicado, está basada en servicios Web, lo que la hace totalmente accesible desde cualquier plataforma, lenguaje de programación, estándares o protocolos utilizados, etc., por lo tanto, se trata de una arquitectura de sistema de una gran flexibilidad.

El fichero XEL obtenido, será simplemente un fichero XML en el que se detallan los campos educativos indicados por el usuario, así como los distintos valores que se recogen del fichero de metainformación del objeto de aprendizaje. En caso de no encontrarse un campo, o una categoría de campos completa dentro del fichero de metainformación, simplemente el sistema lo rellenará vacío.

En el listado 4.2 se muestra un ejemplo de fichero XEL obtenido durante uno de los procesos de búsqueda que más adelante analizaremos. Este fichero XEL, es el resultante de procesar uno de los objetos de aprendizaje de un repositorio dado.

```
<?xml version="1.0" encoding="UTF-8"?>
<educational_content>
  <general>
    <title/>
    <language>es</language>
    <description>Java, orígenes.</description>
    <keyword>Programación</keyword>
    <keyword>Java</keyword>
    <keyword>Objetos</keyword>
  </general>
  <technical>
    <format>application/pdf</format>
  </technical>
  <educational>
    <learningresourcetype>LOMv1.0 Diagram
    </learningresourcetype>
    <difficulty>LOMv1.0 easy</difficulty>
    <description>PDF</description>
  </educational>
  <rights>
    <cost>LOMv1.0 no</cost>
    <copyrightandotherrestrictions>LOMv1.0 no
    </copyrightandotherrestrictions>
    <description>Para difundir esta información es
    necesario incorporar el nombre del
    autor.</description>
  </rights>
  <annotation>
    <person>Salvador Otón Tortosa</person>
```

```
    <date>02-12-2005</date>
  </annotation>
</educational_content>
```

Listado 4.2 Ejemplo de fichero XEL generado por un sistema que sigue la arquitectura propuesta

Como se puede observar, este fichero posee una serie de campos educativos, agrupados en categorías. Para la mayor parte de las categorías, vemos como se habían rellenado esos mismos campos educativos en el objeto de aprendizaje, y por lo tanto aparecen los datos; pero, sin embargo, el campo *title* de la categoría *general*, no había sido rellenado o no existía en ese objeto de aprendizaje, por lo tanto en el fichero XEL lo veremos vacío.

El servicio Web asociado a cada repositorio, se tendrá que encargar de devolver a SROA todos aquellos objetos de aprendizaje que se correspondan con los parámetros de búsqueda especificados por el usuario. Para realizar esta comparación, el servicio Web deberá llamar a operaciones propias del repositorio, que proporcionen los mecanismos de búsqueda y comparación de sus objetos de aprendizaje. Para la devolución de los diferentes objetos de aprendizaje a SROA, utilizaremos SAAJ (*SOAP with Attachments API for Java*) [Sun, 2006]. Por lo tanto, se enviará un mensaje con archivos adjuntos, que incorpore todos aquellos objetos de aprendizaje que le pudieran ser de interés al usuario. Con esta tarea, finalizaría el proceso realizado por esta capa de la arquitectura.

4.5.2 Capa 2: Capa de Interoperabilidad

Esta capa es la que relaciona los repositorios distribuidos con SROA, además de contener los servicios más complejos de procesado de información, debemos proveer un mecanismo de orquestación de servicios para que su ejecución se haga de una forma controlada y ordenada.

4.5.2.1 Capa 2-a: Directorio de Servicios.

Una vez desarrollados los servicios de la capa anterior, éstos deben ser publicados en un directorio de servicios para su posterior localización. El funcionamiento sería similar al de un motor de búsqueda, como Google, con el protocolo HTTP (*Hypertext Transfer Protocol*) y HTML (*Hypertext Markup Language*) para localizar y publicar contenidos.

En el caso de los servicios Web, para que sea localizable, el repositorio debe publicar su servicio Web de acceso en un registro UDDI, de forma que si desde otro sistema se quiere acceder a sus objetos de aprendizaje, tan solo tendrá que localizar su registro en el UDDI y descargarse el WSDL que describe el servicio, ya que la arquitectura de los servicios Web mediante este mecanismo hace que los sistemas sean interoperables.

Por lo tanto, cuando un usuario quiera dar acceso exterior a un repositorio, lo primero que hará será publicarlo en un registro UDDI. La información del UDDI donde está publicado y la propia del servicio es la que debe proporcionar a SROA para que forme parte de la base de repositorios distribuidos y de esta forma quedará catalogado. De esta necesidad surge un servio Web encargado de mantener una base de datos con un catálogo actualizado de todos los servicios de acceso a los repositorios registrados en SROA. Se almacenará información de todos los repositorios, como el UDDI asociado a cada servicio y del propio servicio, de forma que se disponga de su descripción WSDL y se pueda ejecutar cada uno de los servicios de forma ordenada y controlada. Esta base de datos, también será utilizada por SROA para la actualización de los datos de los servicios Web, la que se encargará de modificar los enlaces a los servicios, en el caso de que estos varíen en un momento dado. Realmente este servicio es el que hace interoperable a SROA con el resto de sistemas, ya que es el encargado de enlazar los distintos repositorios.

Esta capa puede utilizarse para hacer más adaptativa la arquitectura. Puede darse el caso de que en el futuro alguno de los servicios de acceso a un repositorio cambie de localización (cambie de IP por ejemplo); en este caso, si SROA utiliza una localización fija, dejaría de funcionar. Si por el contrario se utilizara algún sistema de almacenamiento para fijar las URL de conexión (una base de datos por ejemplo), tendríamos que modificarla para hacer que SROA siga funcionando. Sin embargo podemos ir un paso más adelante; proponiendo un sistema que se pueda actualizar automáticamente. Para ello, utilizaremos del mismo modo estos registros de descripciones de servicios Web (UDDI), de tal forma que en el momento que se detecte que no se puede acceder a un servicio determinado, podamos acudir al UDDI y obtener su nueva descripción, en la que se observan, entre otras cosas, el enlace indicado para dicho servicio. De esta forma, el usuario se podrá despreocupar de si los enlaces a los servicios de la aplicación, pueden variar en el futuro.

4.5.2.2 Capa 2-b: Servicios de integración.

En esta capa es donde se encuentran los servicios más complejos, ya que serán los encargados de realizar las búsquedas federadas en diversos repositorios y localizar los objetos de aprendizaje, a través de la localización del servicio asociado a cada repositorio. Una vez recuperados los objetos de aprendizaje deberán ser catalogados, para ello se procederá a su filtrado, conversión al estándar que espera el cliente y ordenación por su índice de coincidencia.

Cuando se ejecuta un servicio de búsqueda de contenidos (explicado en la capa 3), en SROA se desencadena la llamada a una serie de servicios. El servicio principal es el servicio de búsqueda federada que se encargará de realizar las llamadas a los distintos servicios de búsqueda de los repositorios a los que se tiene acceso. Estos servicios,

como se comentó anteriormente, se deberían realizar mediante mensajería SOAP con archivos adjuntos (SAAJ), y de esta manera se recibiría el objeto o los objetos de aprendizaje que coincidan con los parámetros de búsqueda.

Antes de poder realizar el conjunto de esta búsqueda, será necesario conocer el tipo de especificación con la que trabaja cada repositorio configurado, para de esta manera, se pueda adaptar el fichero XEL de metainformación a la especificación utilizada, y se pueda realizar así la búsqueda solicitada. Para realizar esta tarea, SROA contará explícitamente con un servicio que realizará esta conversión en aquellos casos en los que sea necesario, este será el servicio de adaptación del fichero XEL.

Cuando finaliza la búsqueda federada sobre los distintos repositorios, tendremos un conjunto de objetos de aprendizaje que se deberán filtrar, convertir al estándar requerido por el cliente y ordenar por índice de coincidencia. Por lo tanto, tendremos un servicio de catalogación encargado del filtrado y la ordenación, y otro servicio de conversión. La principal labor del servicio de catalogación es el filtrado, es decir, hacer una clasificación de todos los LO recibidos de forma que se eliminen los duplicados. La siguiente tarea que se realizará es, en su caso, la conversión de los objetos de aprendizaje al formato del estándar esperado por el cliente. Para ello se convertirá la metainformación de un estándar a otro.

Centrándonos en una aplicación práctica, tendremos como bien se ha comentado, un servicio Web de búsqueda federada, encargado de ir realizando las diferentes peticiones sobre los servicios de búsqueda asociados a cada repositorio, utilizando previamente el servicio de conversión del fichero XEL en caso de ser necesario. Cada uno de estos servicios, como recordaremos, retornará un conjunto de objetos de aprendizaje, que cumplen las condiciones marcadas por el usuario. Dicho servicio, tendrá por lo tanto, que tener precaución en no “reemplazar” dos objetos de aprendizaje de diferentes repositorios por el simple hecho de, por ejemplo, llamarse igual, ya que aunque su nombre sea el mismo, puede tratarse de objetos diferentes. Una vez que se hayan realizado todas las llamadas sobre los diferentes servicios de búsqueda asociados a los repositorios distribuidos, tendremos en el SROA un conjunto de objetos de aprendizaje que cumplen las características marcadas por el usuario, pero será necesario filtrar esta información, ya que se pueden tener objetos iguales, que residían en diferentes repositorios. Para ello, se contará con una operación de filtrado dentro del servicio de catalogación, que eliminará todos aquellos objetos que sean iguales. Para realizar esta tarea, habría que basarse en la descripción del objeto. En nuestro caso, identificaremos a dos objetos iguales, por el índice de coincidencia y el tamaño del mismo. Se considera que estos son unos parámetros lo suficientemente restrictivos como para que se eliminen objetos iguales, y no eliminemos objetos que no tengamos que eliminar.

Podría darse el caso de que dos objetos iguales estén descritos de diferente forma, en este caso, SROA los considerará objetos diferentes, ya que la arquitectura se basa en su metainformación para identificarlos.

Otra tarea del servicio de catalogación es la identificación de aquellos objetos de aprendizaje cuya metainformación no coincida con el estándar utilizado por el cliente. En estos casos, pedirá al servicio de conversión que realice la oportuna transformación de metadatos para tener el objeto de aprendizaje correctamente descrito en el estándar esperado.

Por último, el servicio de catalogación realizará la ordenación de los objetos de aprendizaje, realizada por el índice de coincidencia con los parámetros que especificó el usuario en el formulario de entrada a SROA. Por lo tanto, en la capa de acceso y presentación, SROA le presentará una tabla ordenada (en función del índice de coincidencia) en la que dicho usuario podrá seleccionar el/los objeto(s) que le interesen para su descarga.

Como se puede apreciar, la ejecución de estos servicios conlleva la realización de la orquestación de los mismos, es decir, necesitamos realizar una ejecución controlada de una serie de servicios que muchas veces dependen de la ejecución de otros. Para llevar esto a cabo y siguiendo las recomendaciones expuestas en SOA podemos utilizar BPEL [2006].

4.5.3 Capa 3: Servicios de Aplicación y Servicios Comunes

En esta capa residen los servicios de aplicación y comunes correspondientes a la funcionalidad explicada en los casos de uso de la arquitectura. Por lo tanto encontraremos los servicios que el usuario invoca a través de la capa de acceso y presentación. Alguno de estos servicios desencadenan llamadas a los servicios de las capas inferiores. A continuación se comentan los servicios que incluye esta capa:

- **Servicios de Aplicación**:

  - Servicio de búsqueda de contenidos: Sin duda es el más importante y es el que desencadena todas las llamadas a los servicios de las capas inferiores. Lo primero que realiza este servicio es la recogida de los metadatos del formulario que el usuario ha rellenado con sus preferencias de búsqueda, así como la especificación del estándar educativo que utiliza. Hace una llamada al servicio de creación del fichero de metadatos para que genere el fichero XEL que será la clave de las búsquedas en los repositorios distribuidos. Una vez generado el fichero, realizará la llamada al servicio de búsqueda federada al que le pasará el

fichero XEL y el estándar utilizado por el cliente. Como resultado del flujo de llamadas a los servicios de la capa 2 recibirá el conjunto de objetos de aprendizaje encontrados en los repositorios distribuidos ya catalogados (junto con el índice de coincidencia) y convertidos al estándar esperado por el cliente.

- Servicio de creación del fichero de metadatos: Este servicio se encarga de generar un fichero XEL con la información de metadatos que el usuario ha completado en el formulario de entrada para la realización de una búsqueda.

- Servicio de descarga de contenidos: Una vez se presenten los contenidos catalogados al usuario, este podrá descargarlos uno a uno. Si alguno de los contenidos presenta la característica de estar sujeto a cobro por derechos de autor se realizará una llamada al servicio capaz de gestionarlo antes de proceder a su descarga.

- Servicio de recuperación de metadatos comunes: Este servicio es el encargado de leer de un fichero XSD, el conjunto de campos educativos con los que desea trabajar el usuario, así como el conjunto de restricciones sujetas a los campos que se consideren oportunos. Será utilizado por SROA para generar el formulario dinámico en el que el usuario indicará los datos de búsqueda.

- Servicio de registro de repositorios: Este servicio se encarga de registrar un repositorio en SROA, para ello, recogerá los datos de un formulario que previamente habrá rellenado el usuario, y que describen al repositorio como el registro UDDI donde está publicado el servicio, dirección de localización del servicio, claves asociadas a su registro etc. Con estos datos realizará la llamada al servicio de catalogación de repositorios de la capa 2 donde quedará registrado.

- **Servicios Comunes**:

  - Servicio de registro de usuarios: Este servicio es el encargado de registrar a cada uno de los usuarios de SROA y establecer sus privilegios. Como datos básicos se tendrá el nombre de usuario y contraseña, aunque se pueden añadir cualquier dato extra que se pueda utilizar con posterioridad, como una dirección de correo donde enviarle algún tipo de información.

  - Servicio de autenticación de usuarios: Será el encargado de autentificar a los usuarios cuando quieren utilizar SROA y establecer sus privilegios.

  - Servicio de gestión de cobros por derechos de autor de los contenidos: Cuando algún contenido esté sujeto a derechos de autor en los cuales se deba pagar por

su utilización, este servicio se encargará de establecer los mecanismos necesarios para realizar el cobro pertinente.

- Servicio de registro de la trazabilidad de las acciones que se realizan en SROA: Todas las acciones que se realizan en SROA quedarán reflejadas en un fichero de *log*. Este servicio se encargará de ir actualizándolo.

4.5.4 Capa 4: Acceso y Presentación

En este nivel se describe cómo sería la interacción de un cliente con un sistema desarrollado con esta arquitectura. Se describen los principales interfaces (Web o tradicional) y las llamadas a los métodos de la capa 3 que se invocarían.

Para empezar a utilizar SROA, el cliente debe estar registrado; por lo tanto, lo primero que tendría que hacer es registrarse en SROA. Para ello se presentará un formulario donde quedarán recogidos los datos más significativos de los usuarios.

Una vez registrado en SROA el usuario deberá autenticarse, para ello introducirá su nombre de usuario y contraseña y se determinarán los privilegios que este usuario posee en SROA.

Las acciones principales que se le presentarán serán la de realizar una búsqueda federada a través de los distintos repositorios registrados en SROA o la de registrar un nuevo repositorio.

Cuando se quiere realizar la acción de búsqueda, se le presenta al cliente una interfaz donde debe cumplimentar los campos de un formulario. El primer dato que se le pide al usuario, es el estándar de objetos de aprendizaje que utiliza para que cuando se recuperen de los distintos repositorios sea posible su conversión. A continuación, se genera un formulario con los campos educativos preseleccionados. Sería conveniente determinar el conjunto mínimo de campos que debe rellenar para que se proporcione una búsqueda eficiente e indicárselo.

Una vez completado el formulario este pasará a ser procesado por el servicio Web de búsqueda de contenidos que se encargará de confeccionar el fichero XEL de la metainformación que será la base para la realización de las búsquedas posteriores.

Este servicio será el encargado de llamar al servicio de búsqueda federada de la capa 2 y desencadenará todo el proceso explicado anteriormente. Como resultado, recibirá ya clasificados, los objetos de aprendizaje que coincidan con el patrón de búsqueda, los

cuales podrán formar parte del repositorio del cliente (por ejemplo, el de la plataforma LMS que esté utilizando).

Según la implementación práctica que se ha realizado, el fichero XSD será el que indique el conjunto de campos educativos con los que se desea trabajar. Estos pueden pertenecer a una especificación estándar (como los que se utilizarán en el sistema "por defecto", que pertenecen a LOM y descritos en el apartado 4.4), o bien el usuario puede establecer los suyos propios. Este es uno de los factores más importantes de la arquitectura, ya que la hace totalmente independiente de cualquier especificación, siendo completamente compatible con todos. De esta manera queda garantizada su vigencia, a pesar de que surjan nuevas especificaciones en el futuro, o las tendencias del mercado varíen la utilización de unas u otras especificaciones. Además, a través de este fichero XSD, se podrán definir ciertas restricciones de valores asociados a determinados campos, para que de esta manera, se limite el conjunto de valores recibidos.

Por lo tanto, SROA generará un formulario de entrada de datos a partir de este fichero de campos educativos, en el que el usuario introducirá los parámetros de búsqueda. Además, indicará el tipo de búsqueda solicitada (exacta o inexacta) así como el porcentaje de coincidencia mínimo que tendrían que tener todos los objetos de aprendizaje para que sean retornados al usuario.

El diseño realizado, está pensado para que se establezcan dos formas diferentes de búsqueda, para que de esta manera, se le proporcionen al usuario más posibilidades para aproximar sus búsquedas. Las búsquedas que se le proporcionan al usuario son las siguientes:

- Búsqueda Exacta: Utilizando este tipo de búsqueda, se darán solamente válidos aquellos campos que coincidan exactamente (sin considerar diferencias entre mayúsculas y minúsculas).

- Búsqueda Inexacta: Utilizando este tipo de búsqueda, se darán por válidos aquellos campos que cumplan la condición anterior, o que además coincidan en un determinado porcentaje de coincidencia. Dicho porcentaje está establecido en un 25% por defecto, aunque podrá ser modificado por el usuario fácilmente. Esto quiere decir que si una de las cadenas, coincide en al menos un 25% con la otra, se dará por válida.

Para ambos tipos de comparaciones, se contará el número total de campos coincidentes, y se dividirá entre el total de campos, obteniendo con ello un porcentaje de coincidencia. Si dicho porcentaje iguala o supera al indicado por el usuario, este será un objeto válido para devolverlo al SROA.

Una vez rellenados todos estos datos, este servicio Web tendrá que generar un fichero XEL con toda esta información. Este fichero, se utilizará por SROA en la comparación con los diferentes objetos de aprendizaje, siendo adaptado en los casos en los que se requiera para adaptarlos a determinados repositorios.

Una vez finalizada esta tarea, lanzará toda la secuencia de llamadas, obteniendo con ello una lista ordenada de objetos de aprendizaje (en el caso en el que existan coincidencias), a través de la cual, el usuario podrá seleccionar uno o varios de estos objetos para su descarga a la máquina local, y con ello, por ejemplo, incorporarlos a su plataforma LMS.

Otra de las funcionalidades que debe ofrecer SROA, es el registro de nuevos repositorios para su posterior análisis. SROA en este caso, permitirá los tres tipos diferentes de registros siguientes:

- Registro de Servicios sobre el UDDI de SROA: Este será el tipo más sencillo de registro que ofrece SROA, ya que el usuario solamente tiene que conocer la URL en la que reside el servicio Web que va a asociar a su repositorio, así como el nombre del mismo. El resto de los parámetros ya se encuentran preconfigurados en SROA; estos datos son por ejemplo el nombre de usuario y contraseña necesarios para el registro, la URL del UDDI de SROA, o el nombre del negocio en el que se publicará. Una vez que SROA haya registrado este servicio, mostrará la información relativa al servicio registrado (por ejemplo, las claves del mismo).

- Registro de Servicios sobre un UDDI externo al de SROA: En este tipo de registro, el usuario tendrá que proporcionar toda la información que anteriormente hemos indicado que estaba preconfigurada en SROA, ya que ahora el usuario quiere que se registre el servicio Web asociado a su repositorio, sobre un UDDI externo; es decir, sobre el que no se conoce la información necesaria para proceder a su registro. Una vez completados estos datos, de igual forma, se procede a su registro y de ser éste exitoso, se mostrará la misma información que se ha indicado en el caso anterior.

- Incorporación de un servicio Web asociado a un repositorio, pero ya registrado sobre un UDDI: En este tipo de solicitud, no será necesario registrar el servicio Web sobre un UDDI, ya que el usuario ya se encargó de esta tarea. Lo que desea es incorporar este servicio a SROA, por lo que simplemente se incorporará una referencia a la Base de Datos de SROA, pero para ello, se necesita cierta información, como nombre del servicio Web, URL en la que reside, URL del UDDI sobre el que está registrado y negocio sobre el que está publicado. Los valores referentes a las claves del servicio, serán obtenidos tras

buscarlos sobre el UDDI indicado, para que de esta manera, le sean mostrados al usuario, y este los confirme para proceder a incorporarlos también a la Base de Datos.

4.6 DESCRIPCIÓN DE LOS SERVICIOS

En este apartado describiremos con más detalle cada uno de los servicios presentados en el apartado anterior. Para ello utilizaremos las recomendaciones que IMS hace en su *Abstract Framework* [IMS, 2003b] y en su especificación sobre *General Web Services* [IMS, 2005] cuando describe un servicio.

De forma esquematizada realizaremos una tabla como la siguiente para describir cada servicio:

Servicio	Nombre del servicio.
Descripción	Descripción informal del servicio.
Punto de Acceso al Servicio (PAS)	Signatura de las operaciones que realiza el servicio.
Operaciones	Descripción de cada una de las operaciones con los parámetros de entrada y los valores de retorno.
Dependencias	Dependencias de otros servicios.
Excepciones	Principales errores gestionados por el servicio.

4.6.1 Capa 1: Datos y Sistemas Existentes

Servicio	**Búsqueda en repositorio**
Descripción	Servicio asociado a cada uno de los repositorios que están registrados en el sistema. Debe dar acceso al mismo y permitir las búsquedas.
Punto de Acceso al Servicio (PAS)	lista_objetos_aprendizaje *buscarContenido*(fichero_XEL, tipo_busqueda, porcentaje_coincidencia) especificación *especificaciónUtilizada*() objeto_aprendizaje *devolverObjeto*(identificador)
Operaciones	► *Buscar contenido:* ✓ Parámetros de entrada: Fichero XEL (adaptado a la especificación utilizada por el repositorio), tipo de búsqueda solicitada (exacta o inexacta) y porcentaje mínimo de coincidencia de los objetos de aprendizaje. ✓ Resultado: Todos los LO cuyos metadatos coincidan con los especificados en XEL superando el mínimo marcado por el usuario.

	► *Especificación utilizada:* ✓ Parámetros de entrada: ninguno ✓ Resultado: Nombre de la especificación de metadatos utilizada (LOM, SCORM, IMS, etc.) ► *Devolver objeto:* ✓ Parámetros de entrada: Identificador del objeto de aprendizaje ✓ Resultado: El LO seleccionado.
Dependencias	Servicios internos del repositorio para realizar las operaciones que solicita el servicio de búsqueda.
Excepciones	Problemas relacionados con el procesado o envío con algún LO al SROA.

4.6.2 Capa 2: Capa de Interoperabilidad

4.6.2.1 Capa 2-a: Directorio de Servicios

Servicio	**Catalogación de repositorios**
Descripción	Servicio encargado de mantener actualizada toda la información de los repositorios asociados al sistema (en base de datos), a través de los servicios utilizados para su encapsulación.
Punto de Acceso al Servicio (PAS)	bool *actualizarServicioRepositorio*(servicio, nuevo_estado) estado *comprobarEstado*(servicio) bool *registrarRepositorio*(descripción_repositorio)
Operaciones	► *Actualizar Servicio asociado a un Repositorio:* ✓ Parámetros de entrada: Servicio Web a actualizar, así como su nuevo estado (actualizado/desfasado). ✓ Resultado: Booleano con valores verdadero o falso dependiendo de si se logra actualizar el servicio Web. ► *Comprobar Estado:* ✓ Parámetros de entrada: Servicio Web a comprobar. ✓ Resultado: Estado del servicio Web (actualizado/desfasado). ► *Registrar repositorio:* ✓ Parámetros de entrada: La descripción del repositorio, entre la que podremos destacar: el nombre del servicio Web asociado al repositorio, claves del registro (serviceKey y tModelKey) y UDDI sobre el que está registrado. ✓ Resultado: Booleano con valores verdadero o falso dependiendo de si se logra registrar el repositorio.

Dependencias	Ninguna.
Excepciones	Errores de base de datos al acceder o modificar la información. Errores de acceso o búsqueda sobre los UDDI especificados.

4.6.2.2 Capa 2-b: Servicios de integración

Servicio	**Búsqueda federada**
Descripción	Servicio encargado de realizar las llamadas a los distintos servicios de búsqueda de los repositorios a los que se tiene acceso.
Punto de Acceso al Servicio (PAS)	lista_LO *búsquedaFederada*(fichero_XEL, especificación, tipo_busqueda, porcentaje_coincidencia)
Operaciones	► *Búsqueda federada:* ✓ Parámetros de entrada: Fichero XEL con los metadatos a buscar, especificación utilizada por el cliente (para proceder a la posterior conversión de los LO), así como el tipo de búsqueda a realizar en los repositorios y el mínimo nivel de coincidencia que deberán cumplir los LO para ser considerados como válidos. ✓ Resultado: Todos los LO (de todos los repositorios distribuidos especificados en el sistema), cuyos metadatos coincidan con los especificados en el fichero XEL.
Dependencias	Servicio de catalogación de repositorios. Servicio de catalogación de objetos de aprendizaje. Servicio de conversión de objetos de aprendizaje. Servicio de adaptación del fichero XEL.
Excepciones	Búsqueda sin resultados. Errores de acceso a algún repositorio.

Servicio	**Catalogación de objetos de aprendizaje**
Descripción	Servicio encargado de filtrar y ordenar los objetos de aprendizaje recuperados de varios repositorios.
Punto de Acceso al Servicio (PAS)	lista_LO *filtrarLO*(lista_LO) lista_LO *ordenarLO*(lista_LO)
Operaciones	► *Filtrar objetos de aprendizaje:* ✓ Parámetros de entrada: Todos los LO recuperados por la búsqueda federada. ✓ Resultado: Todos los LO resultado de eliminar los duplicados. ► *Ordenar objetos de aprendizaje:* ✓ Parámetros de entrada: Todos los LO

	filtrados. ✓ Resultado: Todos los LO ordenados por índice de coincidencia.
Dependencias	Servicio de búsqueda federada.
Excepciones	Errores relacionados con la ordenación o eliminación de LO.

Servicio	**Conversión de objetos de aprendizaje**
Descripción	Servicio encargado de la conversión de los objetos de aprendizaje al formato de metadatos esperado por el cliente.
Punto de Acceso al Servicio (PAS)	LO *convierteLO*(especificación, LO)
Operaciones	► *Convierte objeto de aprendizaje:* ✓ Parámetros de entrada: La especificación a la que queremos convertir el LO y el propio LO. ✓ Resultado: El LO convertido a la especificación.
Dependencias	Servicio de búsqueda federada.
Excepciones	Errores derivados de la conversión.

Servicio	**Adaptación del fichero XEL**
Descripción	Servicio encargado de la adaptación del fichero XEL de una especificación a otra.
Punto de Acceso al Servicio (PAS)	fichero_XEL *convierteXEL*(especificación, fichero_XEL)
Operaciones	► *Convierte fichero XEL:* ✓ Parámetros de entrada: La especificación a la que queremos convertir el fichero XEL y el propio fichero XEL. ✓ Resultado: El fichero XEL convertido a la especificación.
Dependencias	Servicio de búsqueda federada.
Excepciones	Errores derivados de la conversión.

4.6.3 Capa 3: Servicios de Aplicación y Servicios Comunes

4.6.3.1 Servicios de Aplicación

Servicio	**Búsqueda de contenidos**
Descripción	Servicio encargado de desencadenar una búsqueda federada.
Punto de Acceso al Servicio (PAS)	lista_LO *búsquedaContenidos*(fichero_XEL, especificación, tipo_busqueda, porcentaje_coincidencia)
Operaciones	► *Búsqueda de contenidos:* ✓ Parámetros de entrada: Fichero XEL con los metadatos a buscar, especificación utilizada por el cliente (para proceder a la posterior conversión de los LO), así como el tipo de búsqueda a realizar en los repositorios y el mínimo nivel de coincidencia que deberán cumplir los LO para ser considerados como válidos. ✓ Resultado: Todos los LO catalogados y convertidos, cuyos metadatos coincidan con los especificados en XEL de todos los repositorios distribuidos.
Dependencias	Servicio de búsqueda federada. Servicio de creación del fichero de metadatos XEL.
Excepciones	Búsqueda sin resultados. Errores de acceso a algún repositorio.

Servicio	**Creación del fichero de metadatos XEL**
Descripción	Servicio encargado de generar un fichero XEL con la información introducida por el usuario.
Punto de Acceso al Servicio (PAS)	fichero_XEL *crearFicheroXEL*(metadatos)
Operaciones	► *Crear fichero XEL:* ✓ Parámetros de entrada: Los metadatos que el usuario ha seleccionado y completado. ✓ Resultado: Fichero XEL.
Dependencias	Ninguna.
Excepciones	Errores derivados de la lectura de los metadatos o la generación del fichero XEL.

Servicio	**Descarga de contenidos**
Descripción	Servicio encargado de descargar el LO seleccionado por el usuario.
Punto de Acceso al Servicio (PAS)	LO *descargarLO*(LO_seleccionado)
Operaciones	► *Descargar objeto de aprendizaje:* ✓ Parámetros de entrada: El LO que el usuario ha seleccionado. ✓ Resultado: El LO.
Dependencias	Servicio de cobro por derechos de autor.
Excepciones	Errores derivados del envío del LO a la máquina del cliente.

Servicio	**Recuperación de metadatos comunes**
Descripción	Servicio encargado de leer de un fichero XSD, el conjunto de campos educativos con los que desea trabajar el usuario, así como las restricciones asociadas a dichos campos en caso de encontrarse.
Punto de Acceso al Servicio (PAS)	lista_metadatos *recuperarMetadatos*(fichero_XSD)
Operaciones	► *Recuperar metadatos:* ✓ Parámetros de entrada: El fichero XSD. ✓ Resultado: Lista con los metadatos.
Dependencias	Ninguna.
Excepciones	No se encuentra el fichero XSD o está mal formado.

Servicio	**Registro de repositorios**
Descripción	Servicio encargado de registrar cada uno de los repositorios del sistema.
Punto de Acceso al Servicio (PAS)	bool *registrarRepositorio*(descripción_repositorio)
Operaciones	► *Registrar repositorio:* ✓ Parámetros de entrada: La descripción del repositorio, entre la que podremos destacar: el nombre del servicio Web asociado al repositorio, claves del registro (serviceKey y tModelKey) y UDDI sobre el que está registrado. ✓ Resultado: Booleano con valores verdadero o falso dependiendo de si se logra registrar el repositorio.
Dependencias	Servicio de catalogación de repositorios.
Excepciones	Errores de base de datos al dar de alta el repositorio.

4.6.3.2 Servicios Comunes

Servicio	**Registro de usuarios**
Descripción	Servicio encargado de registrar a cada uno de los usuarios del sistema y establecer sus privilegios.
Punto de Acceso al Servicio (PAS)	bool *registrarUsuario*(nombre, contraseña, privilegios)
Operaciones	► *Registrar usuario:* ✓ Parámetros de entrada: Nombre de usuario, contraseña y privilegios dentro del sistema ✓ Resultado: Booleano con valores verdadero o falso dependiendo de si se logra registrar al usuario.
Dependencias	Ninguna.
Excepciones	Errores relacionados con el registro de usuarios.

Servicio	**Autenticación de usuarios**
Descripción	Servicio encargado de autentificar a los usuarios cuando quieren utilizar el sistema y establecer sus privilegios.
Punto de Acceso al Servicio (PAS)	bool *autenticarUsuario*(nombre, contraseña)
Operaciones	► *Autenticar usuario:* ✓ Parámetros de entrada: Nombre de usuario y contraseña con las que se registró en el sistema ✓ Resultado: Booleano con valores verdadero o falso dependiendo de si se logra autenticar al usuario.
Dependencias	Ninguna.
Excepciones	Errores relacionados con la autenticación de usuarios.

Servicio	**Cobro por derechos de autor de los contenidos**
Descripción	Servicio encargado de cobrar a los usuarios cuando quieren descargar contenidos sujetos a derechos de autor que tengan que ser pagados.
Punto de Acceso al Servicio (PAS)	bool *cobrarDerechos*(coste_derechos)
Operaciones	► *Cobrar derechos:* ✓ Parámetros de entrada: Coste de los derechos a cobrar. ✓ Resultado: Booleano con valores verdadero o falso dependiendo de si se logra cobrar los derechos de autor.
Dependencias	Sistemas de pago.
Excepciones	No se puede cobrar.

Servicio	**Registro de la trazabilidad de las acciones que se realizan en el sistema**
Descripción	Servicio encargado de registrar todas las acciones que se realizan en el sistema y quedarán reflejadas en un fichero de *log*.
Punto de Acceso al Servicio (PAS)	*registrarAcción*(fecha, descripción_acción) fichero_log *examinarLog*()
Operaciones	► *Registrar acción:* ✓ Parámetros de entrada: Fecha y descripción de la acción a registrar en el fichero de *log*. ► *Examinar log:* ✓ Resultado: Fichero *log* con todas las acciones registradas.
Dependencias	Ninguna.
Excepciones	Errores de escritura/acceso en el fichero *log*.

5 VALIDACIÓN DE LA ARQUITECTURA CON LA IMPLEMENTACIÓN DE UN PROTOTIPO REAL

5.1 OBJETIVOS Y SERVICIOS

El objetivo de este capítulo es explicar cómo se ha validado la arquitectura propuesta en el capítulo anterior, mediante la creación de un prototipo real. El resultado de la creación del prototipo, ha sido un "**Sistema de Reutilización de Objetos de Aprendizaje (SROA)**", que si bien no implementa todas las funcionalidades detalladas en la especificación de requisitos, si realiza las principales.

Los objetivos a conseguir con el desarrollo del sistema han sido los siguientes:

- Permitir las búsquedas federadas a través de distintos repositorios distribuidos.

- Permitir el registro de diversos repositorios en el sistema, así como su catalogación.

- Permitir la catalogación de los objetos de aprendizaje resultados de la búsqueda federada.

- Permitir la descarga de cada uno de los objetos de aprendizaje resultados de la búsqueda federada.

- Definir un conjunto de metadatos común, pudiendo incorporar determinadas restricciones a los campos que se consideren oportunos, y encapsularlos en un

fichero XSD para permitir la generación de ficheros XEL (con el formato desarrollado en el capítulo anterior).

- Registrar la trazabilidad de las acciones que se realizan en el sistema mediante un fichero de *log*.

Algunos de los requisitos explicados en el capítulo anterior no han sido implementados en este prototipo, quedando pendiente de su realización en posteriores versiones. Estos requisitos serían los siguientes:

- Realizar la conversión de los metadatos de los objetos de aprendizaje para adaptarlos a la especificación del cliente.

- Permitir el registro y autenticación de los usuarios.

- Gestionar el cobro por los derechos de autor de los contenidos.

Los servicios implementados han sido los que se indican en la tabla 5.1. En esta tabla también se ha incluido una columna con el nombre asignado a cada servicio en la implementación.

CAPA	SERVICIOS	NOMBRE EN IMPLEMENTACIÓN
1. Datos y Sistemas Existentes	• Servicio de búsqueda en repositorio.	o *BuscaRepositorio*
2-a. Directorio de Servicios	• Servicio de catalogación de repositorios.	o *CatalogaRepositorio*
2-b. Servicios de integración	• Servicio de búsqueda federada. • Servicio de catalogación de objetos de aprendizaje.	o *BusquedaFederada* o *BusquedaFederada*
3. Servicios de Aplicación y Servicios Comunes	• Servicio de recuperación de metadatos comunes. • Servicio de creación del fichero de metadatos XEL.	o *ProcesarFicheroXSD* o *GeneraFicheroXELCliente*

	• Servicio de búsqueda de contenidos. • Servicio de descarga de contenidos. • Servicio de registro de repositorio. • Servicio de registro de la trazabilidad de las acciones que se realizan en SROA.	o *BuscaRepositorio* o *CatalogaRepositorio* o *UtilidadesFichero*

Tabla 5.1 Servicios implementados en el prototipo

5.2 IMPLEMENTACIÓN

Para el desarrollo del sistema, se ha elegido la plataforma Java, por ser considerada actualmente la más utilizada [TIOBE, 2006], además de estar perfectamente ligada al desarrollo Web[1]. Este sistema, está realizado basándonos en las arquitecturas SOA, por ser un referente en flexibilidad y posibilitar la integración de aplicaciones y sistemas sin importar la plataforma en que operen, aprovechando las ventajas del ambiente Web y el uso de estándares abiertos. Esto será una gran ventaja para el sistema, porque se podrán realizar funcionalidades adicionales, sin tener que estar desarrolladas obligatoriamente en Java, garantizándose totalmente, la completa interoperabilidad entre las diferentes partes de la arquitectura.

Para el desarrollo de los diferentes servicios Web, se ha elegido una plataforma de libre distribución, que permita su implementación, además de posibilitarnos desplegar en una máquina local un registro UDDI, para poder realizar todas las pruebas necesarias durante el desarrollo. Estas plataformas son *Systinet Server* y *UDDI Registry*, ambas herramientas de *Systinet* [2006], nueva división de *Mercury Interactive* [2006].

La interfaz de la aplicación, es una interfaz Web; para el desarrollo de esta parte, se ha utilizado el servidor de aplicaciones de mayor utilización a nivel empresarial, como es *Apache Tomcat* [Apache, 2006], que nos servirá como contenedor de Servlets/JSP de la aplicación. Además, se integra completamente con el servidor de servicios Web seleccionado.

Para realizar las pruebas del prototipo, evidentemente era necesario contar con una serie de repositorios distribuidos, utilizando para la mayor parte de ellas *Planet Digital*

[1] Se podría haber elegido cualquier otra plataforma orientada a servicios Web como por ejemplo .NET [Microsoft, 2006]

Repository [Planet, 2006], herramienta de libre distribución desarrollada también en Java, la cual cumple las especificaciones marcadas por IMS en DRI, por lo que también está basada en servicios Web (esta vez implementados mediante *Apache Axis* [Apache, 2006]). Este repositorio digital permitirá la incorporación de objetos docentes siempre y cuando estén desarrollados conforme a las especificaciones marcadas por IMS, permitiendo también realizar búsquedas para la obtención completa de dichos objetos o el visionado de su metainformación.

Independientemente del repositorio con el que se opere, tendrá un servicio Web asociado en el que se implementan las operaciones que se detallaron en el capítulo anterior, de forma que se recibe un fichero XEL, y a partir del mismo, utilizando las funciones de búsqueda del repositorio, se logrará devolver una lista con todos aquellos objetos de aprendizaje que cumplen las características marcadas por el usuario.

Para la publicación de los servicios Web asociados a los repositorios, se ha utilizado un registro UDDI donde están dados de alta, proporcionado como ya se ha indicado por la empresa Systinet. El prototipo está preparado para publicar los servicios sobre cualquier UDDI, tanto aquellos que sean de la especificación 2.0 como de la 3.0 [UDDI, 2006]; por lo tanto en este sentido, la plataforma es también muy flexible, ya que es completamente adaptable a cualquier repositorio sobre el que trabajar.

5.2.1 Modelo de clases

En este apartado se comentarán algunas de las clases que serán utilizadas como encapsulación de las operaciones y datos necesarios para las devoluciones de los servicios Web. La gran mayoría de los *Bean* utilizados dependen de la clase *RespuestaGenericaBean*, la cual está formada por un *String*, que representará la cadena de texto producida como resultado de la operación, indicando de manera descriptiva dicho resultado; otro de los campos con los que contará es un *booleano*, que indica a *true* o *false* si se ha producido error o no. Por último, contará con un entero que representará el código de la operación. Cada operación tiene un código (representados en la interface CodigosError.java), de esta forma si se produce cualquier error, tendremos más información para poder solucionarlo. Hay otra clase que hereda de *RespuestaGenericaBean* y que también es muy utilizada para el mismo cometido, se trata de *RespuestaGenericaArrayListBean*, la cual además de los campos que obtiene por herencia, cuenta con un *ArrayList* que podrá ser utilizado para distintas funciones.

En la figura 5.1 se pueden ver estas relaciones de clases:

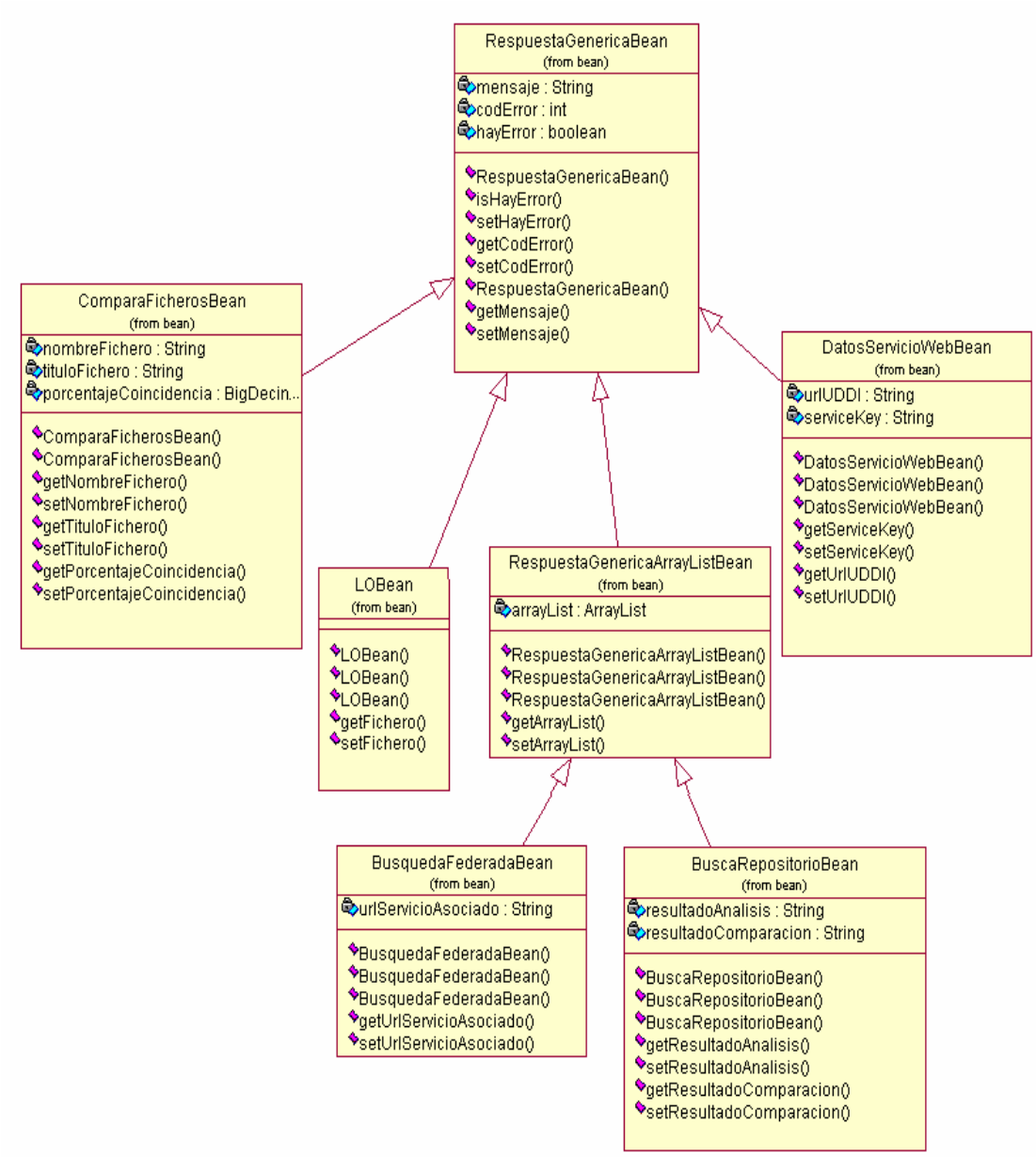

Figura 5.1 Diagrama de clases de los Bean del sistema

Como se aprecia en la figura 5.1 existen bastantes clases que dependen de *RespuestaGenericaBean* y *RespuestaGenericaArrayListBean*, las cuales añaden ciertos atributos, como resultados de sus actividades, *url* de servicios, claves, etc. Muchas de ellas serán comentadas en profundidad cuando se vea la funcionalidad de cada uno de los servicios que las utilizan.

5.2.2 Modelo de orquestación de servicios

Para una mayor comprensión de la implementación realizada de la arquitectura, se va a explicar el modelo BPEL elaborado como diseño de la orquestación de servicios, que se ha realizado con la herramienta Active BPEL [ActiveBPEL, 2006]. En la figura 5.2 se muestra parte de este modelo, la correspondiente a la búsqueda federada en repositorios, ya que es la parte más importante de la arquitectura.

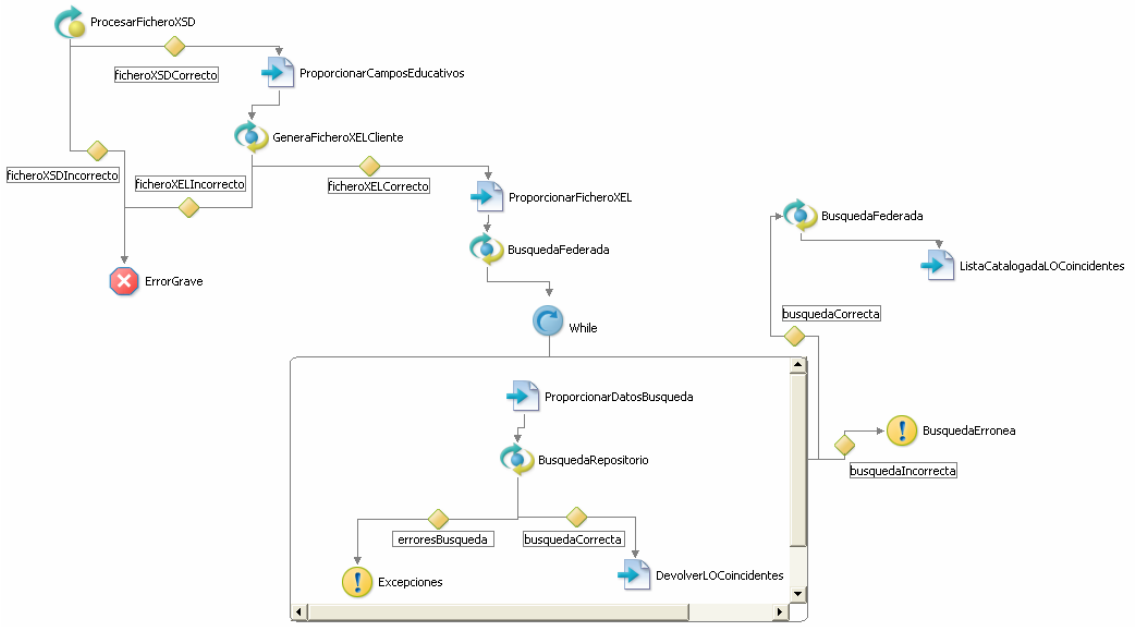

Figura 5.2 Modelo BPEL de la búsqueda Federada en Repositorios

Como se puede ver, la orquestación de servicios comienza a través del servicio Web *ProcesarFicheroXSD*, encargado de procesar el fichero XSD de campos educativos, y obtener el conjunto de campos educativos, así como las distintas restricciones impuestas a los mismos, para que el sistema pueda generar la interface de entrada de información, totalmente adaptada a dichos campos en cada momento. Cuando el usuario rellene la información, se accederá a la funcionalidad *ProporcionarCamposEducativos*, que permitirá acceder al servicio Web *GenerarFicheroXELCliente*, a través del cual se generará el fichero XEL con todos los campos educativos leídos anteriormente (manteniendo su estructuración en categorías), pero ahora incorporando además los valores introducidos por el usuario. A través de este fichero comenzará realmente la búsqueda federada, con la invocación del servicio Web *BusquedaFederada*. Si se produjera algún error durante la obtención de los campos educativos del fichero XSD o durante la generación del fichero XEL, eso supondría un error grave para el sistema, ya que sin esta información, el sistema será incapaz de comenzar la búsqueda, como bien se puede ver en el modelo BPEL.

El servicio Web *BusquedaFederada* es el encargado de leer de la base de datos del sistema todos aquellos repositorios configurados para realizar búsquedas sobre ellos. En realidad, lo que lee son las referencias a los servicios Web de búsqueda que se han asociado a cada repositorio. Por lo tanto, como se puede ver en el modelo, comienza una estructura repetitiva, en la que se irá accediendo a cada uno de estos servicios (en el gráfico llamados *BusquedaRepositorio*), proporcionándole los parámetros de búsqueda (el fichero XEL, el porcentaje de coincidencia mínimo que deberán cumplir los LO y el tipo de búsqueda solicitada), y este servicio devolverá a SROA una lista con todos aquellos LO que se correspondan con los parámetros marcados por el usuario. Si alguno de estos repositorios produjera algún error, eso no supondría, evidentemente, la cancelación de la operación, sino que simplemente se le notificará al usuario de dicho proceso, continuando con la búsqueda si aún quedaran más repositorios por analizar. Para la invocación de los distintos servicios de búsqueda el sistema creará un hilo que se encargará de su procesamiento, siendo este proceso totalmente concurrente.

Cuando el proceso de búsqueda haya finalizado, será momento de catalogar toda esa información. El encargado de realizar el filtrado de los objetos de aprendizaje, eliminando todos aquellos repetidos, así como la ordenación de los mismos, para que de esta manera se le muestre una lista al usuario ordenada por nivel de coincidencia en sentido descendente, será el mismo servicio *BusquedaFederada.*

Posteriormente a esta acción (ya no mostrado en el modelo), el usuario seleccionará de la lista mostrada un objeto de aprendizaje, invocando con ello al servicio Web de búsqueda asociado al repositorio que lo contenía, el cual se encargará de enviarlo hasta su máquina local. Dicho fichero contendrá además de la información docente del mismo, la metainformación asignada por el autor. Esto constituye una de las principales diferencias con respecto a los principales repositorios analizados en el capitulo cuatro, ya que a través de este prototipo podremos obtener el objeto de aprendizaje completo.

A continuación, se van a mostrar algunos de los ficheros WSDL de los principales servicios Web que se han incluido en el modelo BPEL; de esta forma se podrán entender más claramente las acciones realizadas, así como el conjunto de parámetros proporcionados durante la llamada, y los obtenidos como resultado de la misma.

Comenzaremos mostrando la estructura del fichero WSDL del servicio Web de búsqueda asociado a un repositorio creado para el prototipo funcional. Como se puede ver en la figura 5.4, en la llamada al servicio Web se proporcionan tres parámetros, el fichero XEL del cliente (es decir, el fichero que contiene el conjunto de campos educativos estructurados en categorías proveniente del fichero XSD, conjuntamente con los valores introducidos por el usuario), el tipo de búsqueda solicitada (de las dos posibles, exacta e inexacta; esto le permitirá saber al repositorio cuando tiene que dar por válido un campo y cuando no), y el porcentaje de coincidencia (este dato es

sumamente importante, ya que todos los objetos que al menos no igualen este valor, no serán retornados a SROA).

La respuesta de este servicio Web, será un objeto de tipo *BuscaRepositorioBean*, el cual estará formado, como se puede ver en la figura 5.1, por los datos heredados de *RespuestaGenericaArrayListBean*, más las cadenas de caracteres *resultadoAnalisis*, y *resultadoComparacion*, que contendrán el texto explicativo de la operación de análisis en el repositorio (se entiende por análisis, al proceso de extracción de la metainformación de todos y cada uno de los objetos de aprendizaje que estén presentes en el mismo), y la operación de comparación (comparación del fichero XEL con los datos introducidos por el usuario, con el fichero XEL generado por cada uno de los objetos de aprendizaje) respectivamente.

Por último, podemos destacar el *array* de objetos de tipo *LOCoindicenteBean*, que representarán los objetos de aprendizaje que han coincidido con los parámetros marcados por el usuario. Los datos de este objeto son la URL del servicio Web asociado al repositorio que lo contiene, el identificador del fichero (para posibilitar su posible descarga), el nombre de dicho fichero, el porcentaje de coincidencia que ha tenido, así como el tamaño en *bytes* del mismo.

Para que el cometido de este servicio quede más claramente reflejado, se va a mostrar el diagrama de clases que lo conforma (figura 5.3), y posteriormente se indicará la funcionalidad de cada una de estas clases:

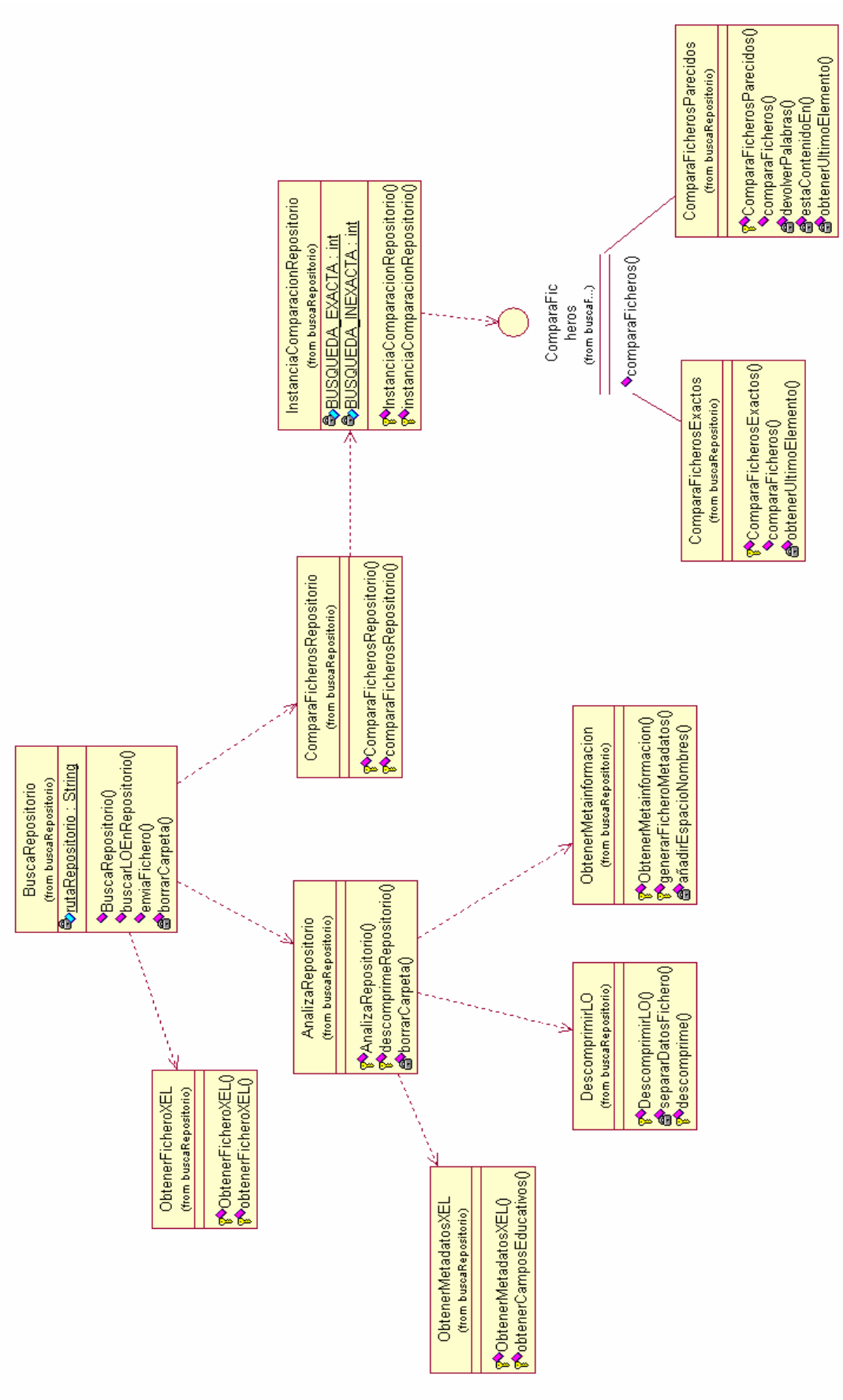

Figura 5.3 Diagrama de clases del servicio Web BusquedaRepositorio

La entrada de la funcionalidad al servicio Web se realiza a través de la clase *BuscaRepositorio*, la cual en primer lugar transformará el flujo de *bytes* que representa el fichero XEL del cliente en un fichero temporal, tarea que será realizada por la clase *ObtenerFicheroXEL*. Una vez completada esta funcionalidad, el flujo pasará a la clase *AnalizaRepositorio*, encargada en primer lugar de obtener el conjunto de campos educativos del fichero XEL, a través de la clase *ObtenerMetadatosXEL*, y por cada uno de los LO del repositorio, irá descomprimiéndolos para obtener su fichero de metainformación (*DescomprimeLO*) y obteniendo su fichero XEL correspondiente (*ObtenerMetainformacion*). Finalizados estos procesos, el servicio Web, se encargará de ir comparando cada uno de los ficheros XEL generados por cada LO del repositorio con el fichero XEL del cliente, y obteniendo así, una lista ordenada con todos los LO que coinciden con los parámetros marcados por el usuario. Esto será realizado por la interface *ComparaFicheros* y sus dos implementaciones *ComparaFicherosExactos* y *ComparaFicherosParecidos*.

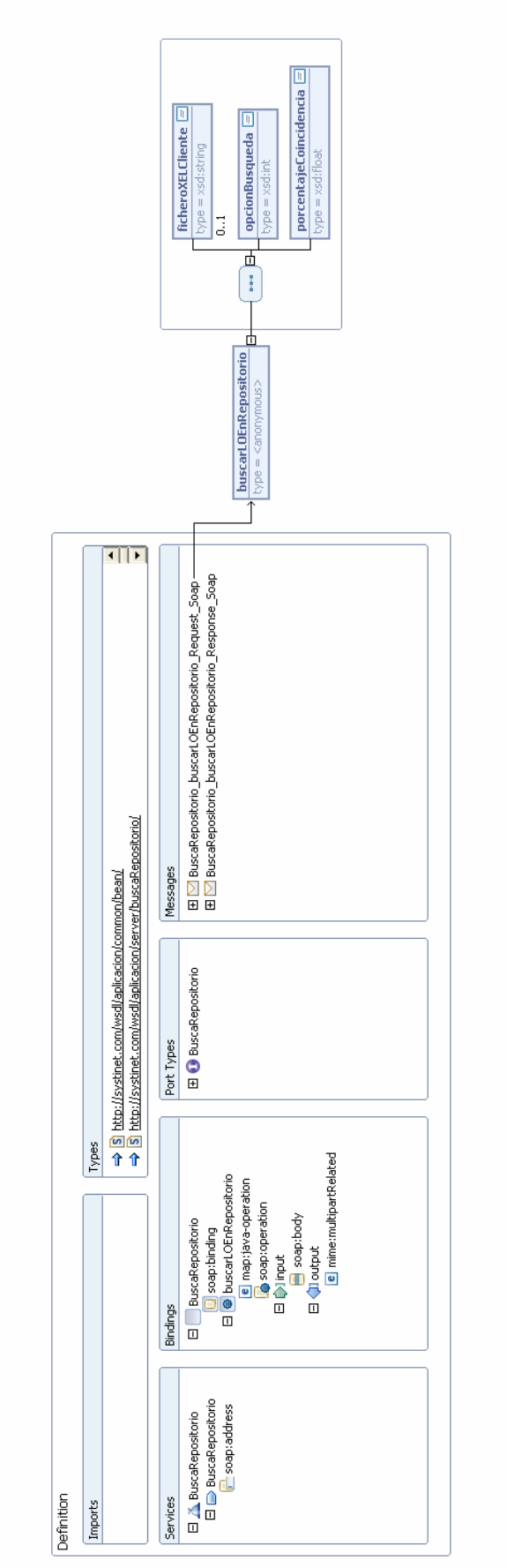

Figura 5.4 Estructura del fichero WSDL del servicio Web de búsqueda (Petición)

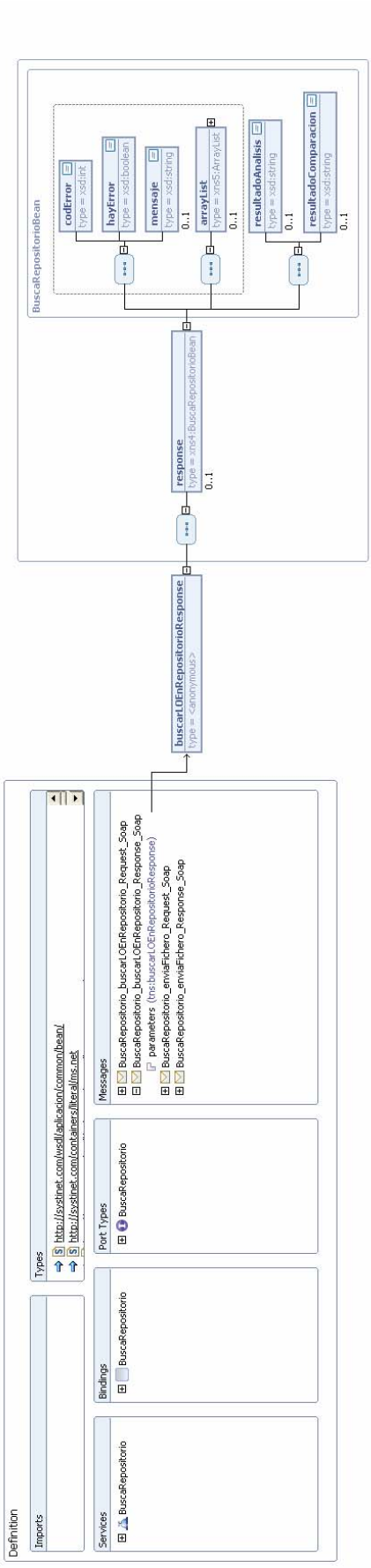

Figura 5.5 Estructura del fichero WSDL del servicio Web de búsqueda (Respuesta)

Este servicio además de aportar la funcionalidad de búsqueda en el repositorio, permite la descarga de uno de los LO cuando lo solicite el usuario. A través de la descripción de su fichero WSDL, se podrá observar al igual que antes, cómo será el proceso de envío de un objeto de aprendizaje desde SROA hasta la máquina local del usuario. La estructura de este fichero es la que se muestra en las figuras 5.6 (cuando se realiza una petición) y 5.7 (cuando se envía una respuesta).

Una vez que el sistema le muestre la lista ya catalogada (ordenada y filtrada), con todos los objetos de aprendizaje que se han correspondido con sus parámetros de búsqueda, el usuario verá que en cada uno de ellos dispone de un enlace para procesar su descarga. Cuando el usuario pulse sobre ese enlace, se accederá a un *Servlet*, el cual está pensado que contacte con el servicio Web que estamos comentando. A dicho servicio le tendrá que proporcionar un nombre o código único que representará a cada uno de los objetos, como se puede ver en la figura 5.6.

Este *Servlet* recibirá como contestación un flujo de *bytes* a través de la clase *org.idoox.wasp.types.ResponseMessageAttachment*. Dicha clase posee un método como es el *setData()*/*getData()* al que se le proporcionará ese flujo de *bytes* y permitirá su obtención. Posteriormente el *Servlet* los recibirá y los irá copiando sobre el *SevletOutputStream*, para que de esta manera le lleguen al usuario. Se ha decidido que en esta implementación, se abra una ventana en la máquina local del usuario, en la que podrá decidir si desea abrir el objeto o guardarlo en cualquiera de sus unidades; para ello se ha utilizado la cabecera http. Todas estas tareas, por ser consideradas de especial interés, se mostrarán a través de su código fuente Java utilizado para su implementación.

Para incorporar un fichero a la estructura *ResponseMessageAttachment*, se hará a través de las siguientes líneas de código, siendo la variable *idFichero* la localización del mismo.

```
……
ResponseMessageAttachment respuesta = new ResponseMessageAttachment();
respuesta.setData(new FileInputStream(idFichero));
……
```

A continuación se presenta el código del *Servlet* que invoca al servicio Web y permite extraer la información de la clase *ResponseMessageAttachment*. *BuscaRepositorio* sería la interface extraída del fichero WSDL del servicio Web, la variable *idFichero* representaría la selección del usuario, es decir, el fichero en cuestión que desea descargarse, mientras que la variable *response*, representa el objeto de tipo *HttpServletResponse*.

```
……
//Haremos que aparezca una ventana en el navegador del cliente para
//que pueda abrir o descargarse el LO.

response.setContentType("application/zip");
response.setHeader("Content-disposition", "attachment;filename=\""+
nombre+"\"");

//Si todo es correcto, le enviamos el fichero al cliente.
BufferedOutputStream bos = new BufferedOutputStream
(response.getOutputStream());
BufferedInputStream bis = new
BufferedInputStream(respuesta.getData());

int c;
while ((c = bis.read()) != -1) {
    bos.write(c);
}

//Es imprescindible que cerremos los buffer para que la copia se
//realice de forma correcta.
bos.close();
bis.close();
……
```

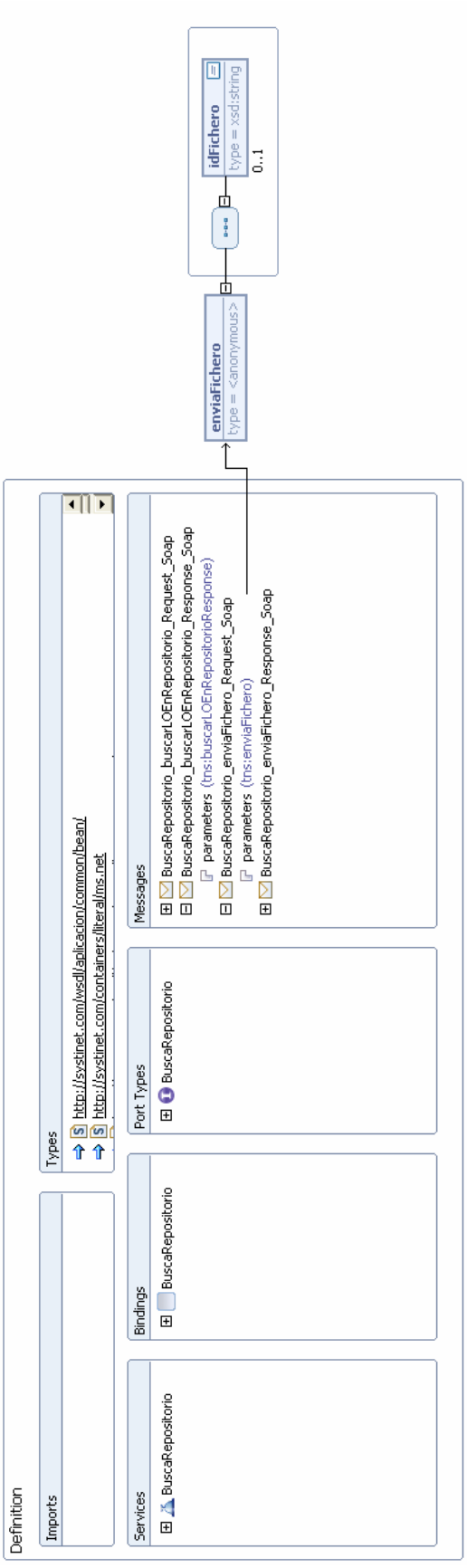

Figura 5.6 Estructura del fichero WSDL de la funcionalidad de envío de Ficheros (Petición)

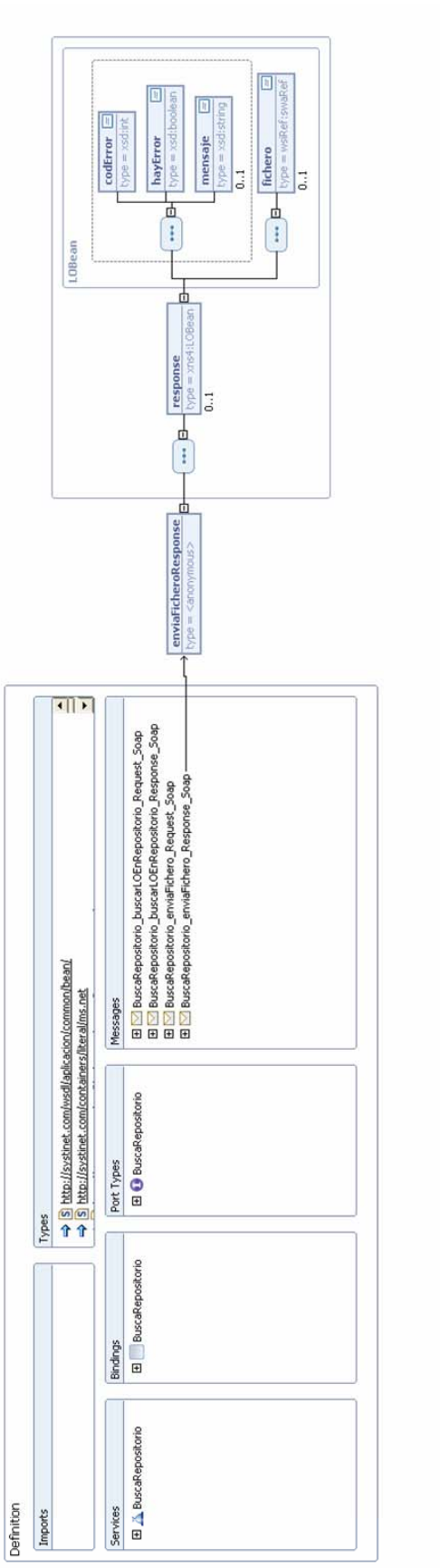

Figura 5.7 Estructura del fichero WSDL de la funcionalidad de envío de Ficheros (Respuesta)

5.2.3 Base de Datos

Como ya se comentó en el capítulo anterior, la arquitectura propuesta y con ello el prototipo presentado, se apoyan en un sistema gestor de base de datos, para aportarle una mayor rapidez, dado que sería muy costoso realizar constantes peticiones a los UDDI registrados en el sistema.

En este prototipo, como aún no se han implementado determinados elementos de la arquitectura, como el registro de usuarios, los permisos de acceso, el cobro por derechos de autor, etc., el sistema resultante es más sencillo del que resultará tras implementar toda la funcionalidad del sistema.

Esta base de datos será utilizada para distintas tareas, entre las que podemos destacar, por ejemplo, configurar el sistema para realizar las búsquedas sobre unos repositorios determinados, almacenar las URL de todos y cada uno de los servicios Web con los que contará el sistema, para que de esta manera no sea necesario realizar búsquedas en UDDI para saber hacia donde redireccionar las peticiones, ni se tenga que trabajar con un direccionamiento estático.

Se ha diseñado una base de datos con tres tablas relacionadas, son las siguientes (figura 5.8):

- *REPOSITORIOS:* Será la que almacene la información referente a los repositorios, por ejemplo, un nombre, dirección o ubicación en el que se encuentra (siempre que el usuario que lo desee registrar, le resulte de interés) o el servicio Web de búsqueda que tiene asociado, evidentemente este parámetro será imprescindible, ya que será a través del cual contactaremos con el servicio.
- *SERVICIOS:* En la que se almacena toda la información relativa a los servicios Web del sistema, como por ejemplo sus claves (*ServiceKey* y *BusinessKey*), el UDDI asociado (en el que está publicado su fichero WSDL), su URL o su estado.
- *UDDIS:*, En la que se relacionan cada uno de los UDDI con sus URL correspondientes (búsqueda, publicación y seguridad), para que sepamos hacia dónde encaminar las búsquedas de los ficheros WSDL de cada servicio Web.

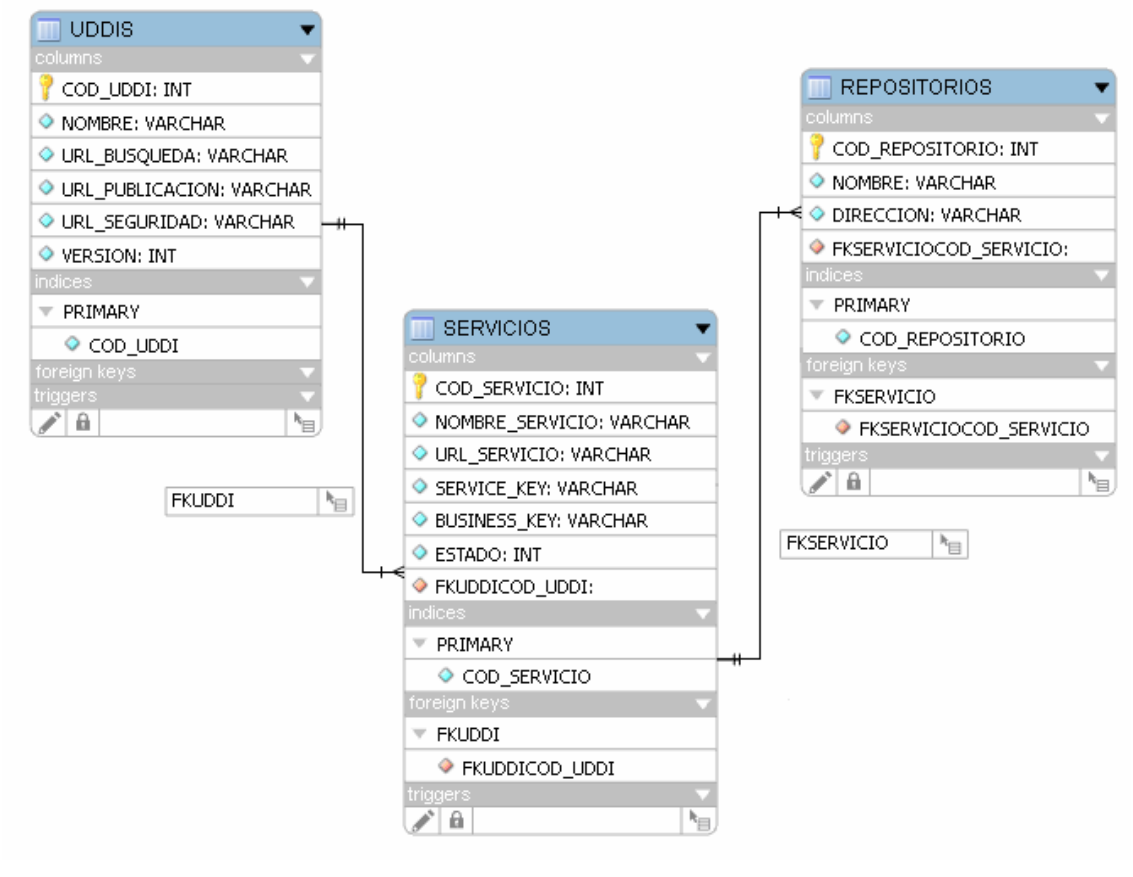

Figura 5.8 Diseño de la base de datos del sistema

Para el prototipo implementado, el sistema gestor de base de datos que se ha utilizado es *MySQL* [2006], por tratarse de un sistema gestor de gran potencia, libre y por ser totalmente independiente de la plataforma.

A continuación, se pasará a comentar cada una de las columnas de la tabla '*REPOSITORIOS*':

- COD_REPOSITORIO: Se trata de una columna obligatoria de tipo numérico (*INTEGER*) que representará el código asociado a cada repositorio que se quiera analizar durante los procesos de búsqueda. Este será la columna clave de la tabla y su valor será generado automáticamente por el gestor de la base de datos en cada inserción.

- NOMBRE: Esta columna (no obligatoria) es de tipo cadena de caracteres (*VARCHAR*) y representa el nombre con el que se conoce al repositorio en cuestión.

- DIRECCION: Esta columna también es de tipo cadena de caracteres y no obligatorio; indicará la dirección en la que se encuentra el repositorio; como, por ejemplo, país, cuidad, población o todo de manera conjunta. En nuestro caso, como se han simulado los repositorios utilizando sistemas de ficheros, representará el nombre de la carpeta así como su localización dentro del sistema de archivos del sistema operativo.

- FKSERVICIOCOD_SERVICIO: Se trata de una clave ajena de tipo entero (*INTEGER*), obtenida de la relación entre el repositorio y el servicio. Permitirá conocer el servicio Web de búsqueda asociado a cada repositorio. El sistema deberá listar el contenido de esta tabla para conocer todos aquellos servicios sobre los que tiene que realizar una petición para realizar búsquedas de objetos de aprendizaje.

La siguiente tabla denominada '*SERVICIO*', será la encargada de representar a los servicios Web del sistema (compuesto por los servicios de búsqueda asociados a cada repositorio y los internos del propio sistema); sus columnas son:

- COD_SERVICIO: Se trata de una columna obligatoria de tipo numérico (*INTEGER*) que representará el código asociado a cada servicio Web del sistema. Este será el campo clave de la tabla y su valor será generado automáticamente por el gestor de la base de datos en cada inserción.

- NOMBRE_SERVICIO: Se trata de una columna de tipo cadena de caracteres (*VARCHAR*), que no será obligatoria, y que representará el nombre que se le podrá asociar a cada uno de los servicios Web del sistema, para mejorar la identificación de cada uno de ellos.

- URL_SERVICIO: En esta columna obligatoria de tipo *VARCHAR*, se almacenará la URL asociada al servicio Web. En caso que se produjera un fallo durante su búsqueda, se recurrirá al UDDI que tenga asociado para buscar su nueva localización. Posteriormente, se actualizará esta columna en la base de datos. Como se ha comentado, será mucho más rápido para el sistema buscar las URL de los servicios en esta base de datos, que no en sus UDDIs correspondientes.

- SERVICE_KEY: Esta una columna obligatoria de tipo cadena de caracteres; representa el código del servicio Web. Durante la publicación de un servicio, el UDDI nos pide (en nuestro caso le indicaremos que nos lo genere automáticamente) una clave asociada a dicho servicio. Este será uno de los campos por los que buscaremos dentro del UDDI cuando sea necesario buscar una posible actualización de la URL asociada a dicho servicio.

- BUSINESS_KEY: Se trata de una columna obligatoria y también de tipo cadena de caracteres, representará la clave del negocio sobre el que está publicado dicho servicio Web. En el caso de este prototipo, el usuario no tendrá que memorizar ni copiar estos campos, puesto que el sistema se los presentará en una lista para su selección.

- ESTADO: Esta columna obligatoria y de tipo entero, representa el estado de actualización que posee el servicio en cuestión. Ya sabemos que el sistema es capaz de buscar en UDDI una posible actualización del servicio Web, con este campo conseguiremos que las búsquedas de actualización se realicen solamente una vez. El estado de un servicio "válido" es "cero". Con un estado así, en caso de producirse un error de localización, se buscaría una actualización y se modificaría su estado a "uno", indicando así que el servicio está actualizado tras un error. Posteriormente se intentaría de nuevo contactar con dicho servicio; en caso de conseguirlo, se volvería a cambiar su estado a "cero", es decir, acceso correcto al sistema. Sin embargo, en caso de producirse un nuevo error con un estado "uno", no se intentaría otra búsqueda en UDDI y se le notificaría el error al usuario.

- FKUDDICOD_UDDI: Esta clave ajena surgida de la relación entre las tablas '*UDDI*' y '*SERVICIO*', representa el UDDI asociado a dicho servicio Web. Se trata de un código de tipo numérico (*INTEGER*), que servirá para enlazar con la tabla '*UDDI'* y saber la URL de búsqueda de dicho UDDI para realizar las actualizaciones de información.

Por último, la tabla '*UDDIS*', se compone de las siguientes columnas:

- COD_UDDI: Este código obligatorio de tipo numérico, que además es la columna clave de la tabla, permitirá distinguir de manera inequívoca un registro UDDI de otro. El valor de esta columna, al igual que el resto de códigos de las otras tablas, será generado automáticamente por el gestor de la base de datos tras cada inserción.

- NOMBRE: Se trata de una columna no obligatoria de tipo cadena de caracteres (*VARCHAR*), que representa el nombre del UDDI. Este campo sólo es informativo para el usuario, ya que los datos que de verdad importan son las URL del UDDI.

- URL_BUSQUEDA: Es otra columna de tipo cadena de caracteres (*VARCHAR*) que representará la localización de dicho UDDI para realizar búsquedas sobre él.

Este campo será obligatorio, ya que es la única información imprescindible que necesitaremos del UDDI.

- URL_PUBLICACION: Esta otra URL, también de tipo cadena de caracteres, será el enlace con el UDDI para realizar modificaciones sobre él; como, por ejemplo, publicaciones de servicios. En ese caso, el usuario necesitará además un código de usuario, con su contraseña correspondiente y un negocio sobre el que publicar. Este campo no es obligatorio.

- URL_SEGURIDAD: La última de las URL con las que contará el sistema, es solamente para los UDDI de la versión 3.0, ya que uno de los añadidos de esta versión es el uso de certificados, por lo que hará falta una URL de acceso al UDDI que aporte SSL. Este campo tampoco será obligatorio.

- VERSION: Este dato obligatorio de tipo entero (*INTEGER*) representará la versión del UDDI en cuestión, y será un dato importante para realizar peticiones y modificaciones sobre el mismo, ya que las API's de programación varían en función de una u otra versión.

5.3 PRUEBA DEL SISTEMA

Para validar la implementación de la arquitectura presentada en el apartado anterior, se ha desarrollado una aplicación que hace uso de los servicios creados. Se trata de una aplicación Web que en su página principal ofrece al usuario la posibilidad de realizar una búsqueda federada en distintos repositorios distribuidos o la de registrar un nuevo repositorio en el sistema (figura 5.9).

Figura 5.9 ***Página principal de entrada al sistema***

En los siguientes apartados se describen con detalle ambas opciones.

5.3.1 Búsqueda federada en repositorios distribuidos

Cuando el usuario elije esta opción en la página principal de la aplicación, el sistema analizará el fichero XSD de campos educativos y completará el formulario de entrada de datos de forma dinámica, mostrándole al usuario, los datos que tendrá que rellenar. Como puede verse en la figura 5.10, los campos que conforman el formulario son exactamente los mismos que estaban indicados en el fichero de campos educativos explicado en el capítulo anterior. Se ha mantenido la coherencia de dichos campos, respetando su estructuración en categorías. Además, a través del fichero XSD, se han especificado ciertas restricciones para algunos de los campos, como por ejemplo, para *format*, *language* o *cost*; por lo que el sistema mostrará un menú desplegable para dichos campos, en vez de un campo de texto para la libre introducción de valores.

A continuación, el usuario podrá incorporar los parámetros de búsqueda que desee. Cuantos más sean los parámetros que éste introduzca, mayores serán las probabilidades de obtener más objetos de aprendizaje. Además, el usuario podrá indicar un porcentaje de coincidencia mínimo con el que desea trabajar, así como el tipo de búsqueda solicitada, a partir de las dos propuestas: Búsqueda Exacta, en la que se darán solamente por válidos aquellos campos que coincidan exactamente; o Búsqueda Inexacta, en la que además se darán por válidos aquellos campos que coincidan al menos en un 25% (parámetro configurable).

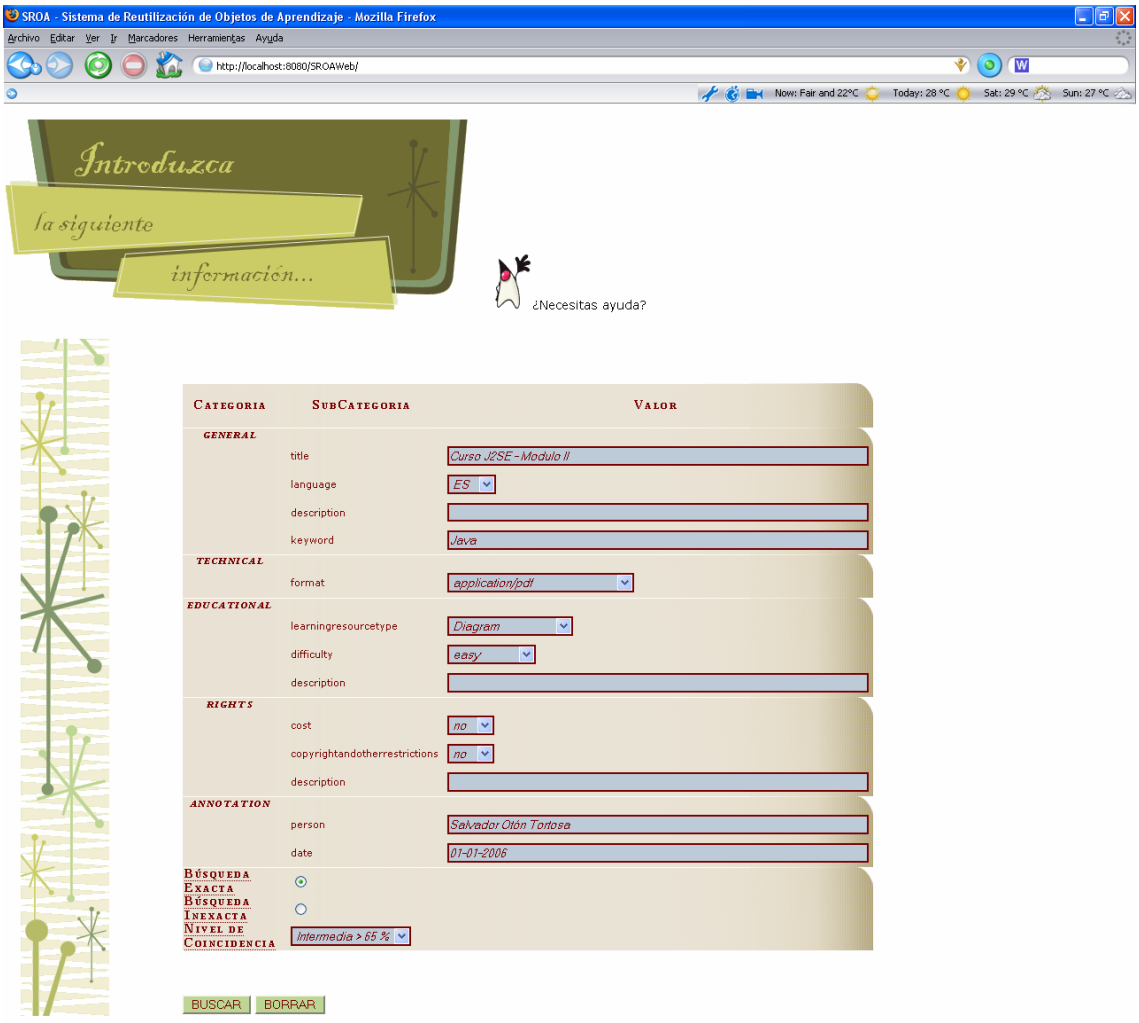

Figura 5.10 Formulario de metadatos para la búsqueda federada

Para realizar las pruebas del sistema, se contará con tres repositorios (tabla 5.2) implementados a través de la herramienta anteriormente comentada, *Planet Digital Repository*. A continuación se mostrarán algunas imágenes (figura 5.11 y 5.12) de dichos repositorios.

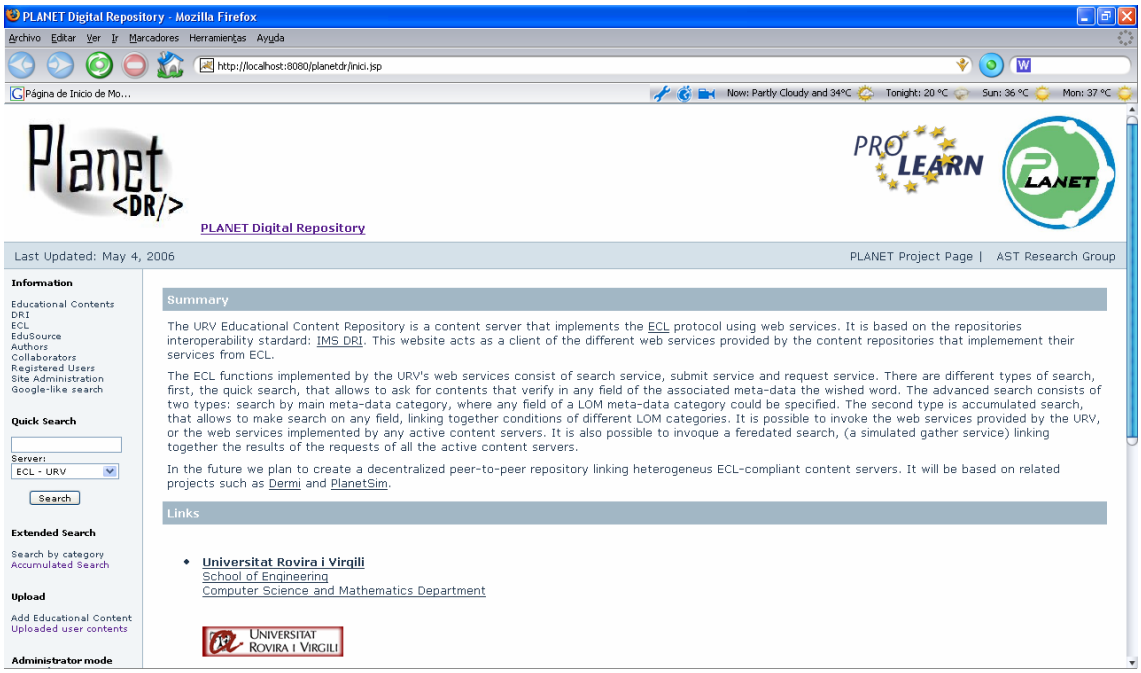

Figura 5.11 Página principal del Repositorio

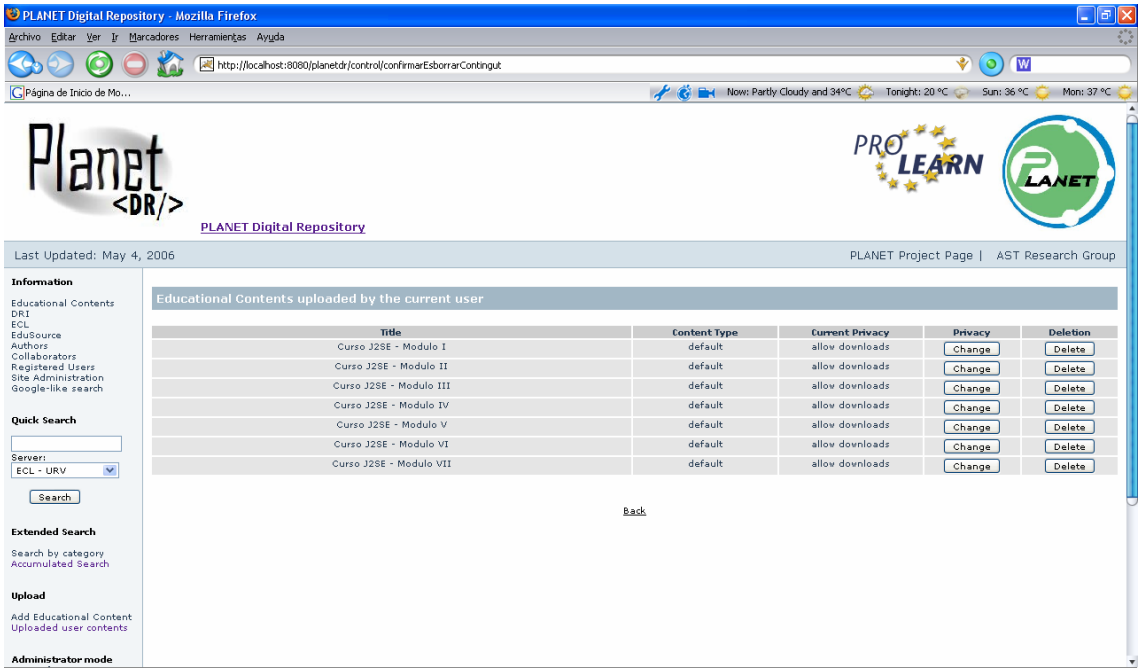

Figura 5.12 Instancia del primero de los Repositorios

En el primero de ellos, hay un conjunto de objetos de aprendizaje que pertenecen a un curso de J2SE; además se han incorporado una serie de objetos incorrectos para que el sistema los detecte y se compruebe que supera este tipo de errores sin problemas. En el segundo de los repositorios, hay un grupo de objetos de aprendizaje de un curso de J2EE. El último de los repositorios contiene el mismo grupo de objetos que teníamos en el segundo repositorio, para que de esta manera se compruebe el sistema de filtrado con el que cuenta la arquitectura, y además también un curso sobre el lenguaje de programación C.

Nombre	Número de objetos de aprendizaje	Formato	Ubicación
Curso de J2SE	7	IMS	Repositorio1
Curso de J2EE	4	IMS	Repositorio2 Repositorio3
Curso de C	4	IMS	Repositorio3

Tabla 5.2 Repositorios reales de objetos de aprendizaje utilizados[2].

Cuando el usuario rellene toda la información que él considere oportuna en el formulario de metadatos para la búsqueda federada (figura 5.10), el sistema buscará en todos los repositorios registrados, todos aquellos objetos de aprendizaje que coincidan con los parámetros marcados por el usuario, devolviéndole a éste, una lista ordenada de la que podrá ir descargando todos y cada uno de ellos (figura 5.13).

Figura 5.13 Coincidencias encontradas.

[2] Estos objetos de aprendizaje se han creado con la financiación del Ministerio de Industria a través del proyecto PROFIT "FIT-350101-2004-7", en el que ha participado el autor de esta tesis.

En la pantalla de coincidencias el sistema mostrará el nombre de cada objeto de aprendizaje, junto con su porcentaje de coincidencia y un enlace para procesar su descarga. Esta es una de las principales diferencias con respecto a los demás repositorios analizados en el capítulo tres: la arquitectura presentada posibilita la descarga del objeto de aprendizaje, junto con su metainformación, de manera totalmente independiente del formato o manera de organizar los objetos docentes.

Si se deseara descargar un objeto de aprendizaje, solamente se tendrá que pulsar sobre “Descargar”, y aparecerá una ventana como la que se muestra en la figura 5.14.

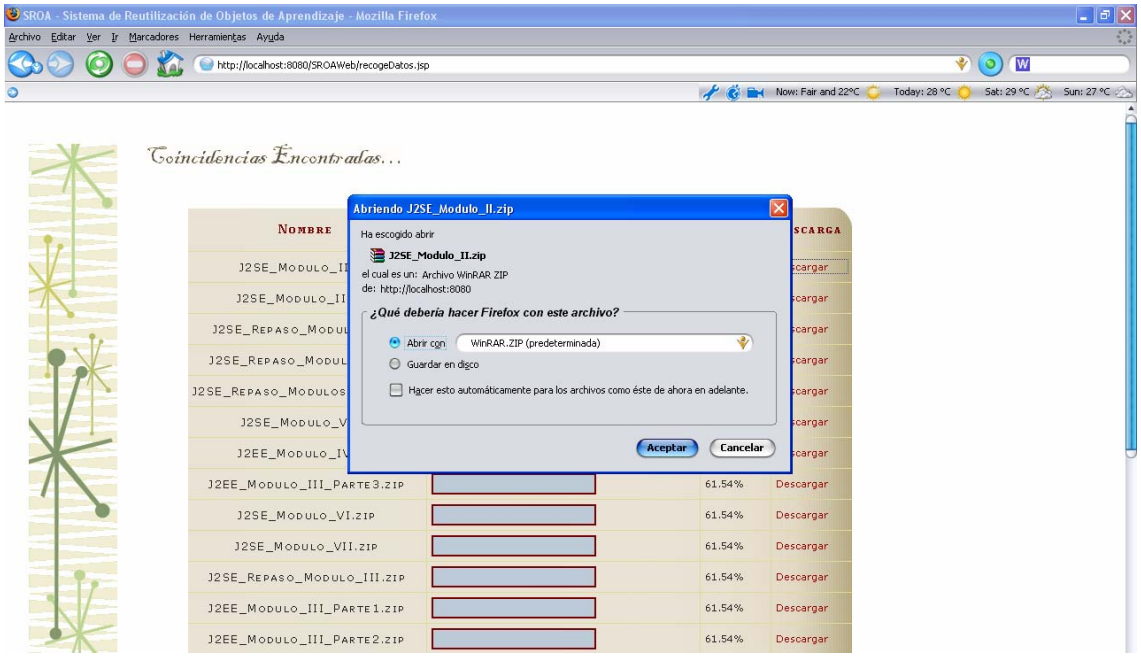

Figura 5.14 Descarga de un objeto de aprendizaje.

En la ventana que se muestra, el usuario puede elegir si desea abrir el fichero o descargarlo a su máquina local. Si desea abrirlo, pulsará sobre esa opción, y el sistema mostrará el contenido del fichero (figura 5.15).

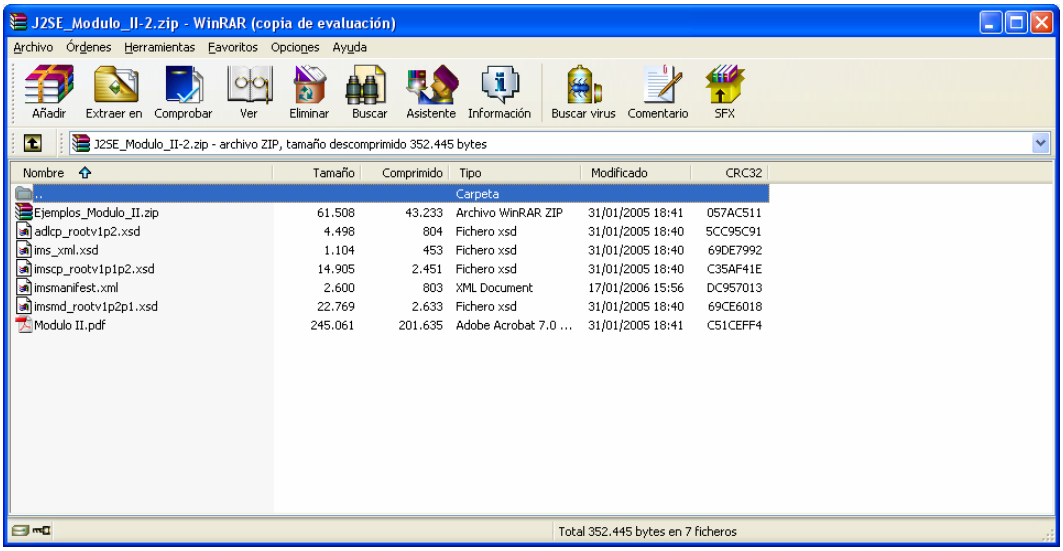

Figura 5.15 Contenido de un objeto de aprendizaje

Una de las características que tenía el sistema, era la capacidad de poder buscar posibles actualizaciones de los servicios Web, para detectar, por ejemplo, si estos cambiaban de localización. Para probar el cumplimiento de este requisito se modificarán los enlaces que hay en la base de datos, para que de esta manera, se verifique un correcto funcionamiento de la aplicación a pesar de ello. Esto se puede comprobar analizando el contenido del *log* del sistema (figura 5.16).

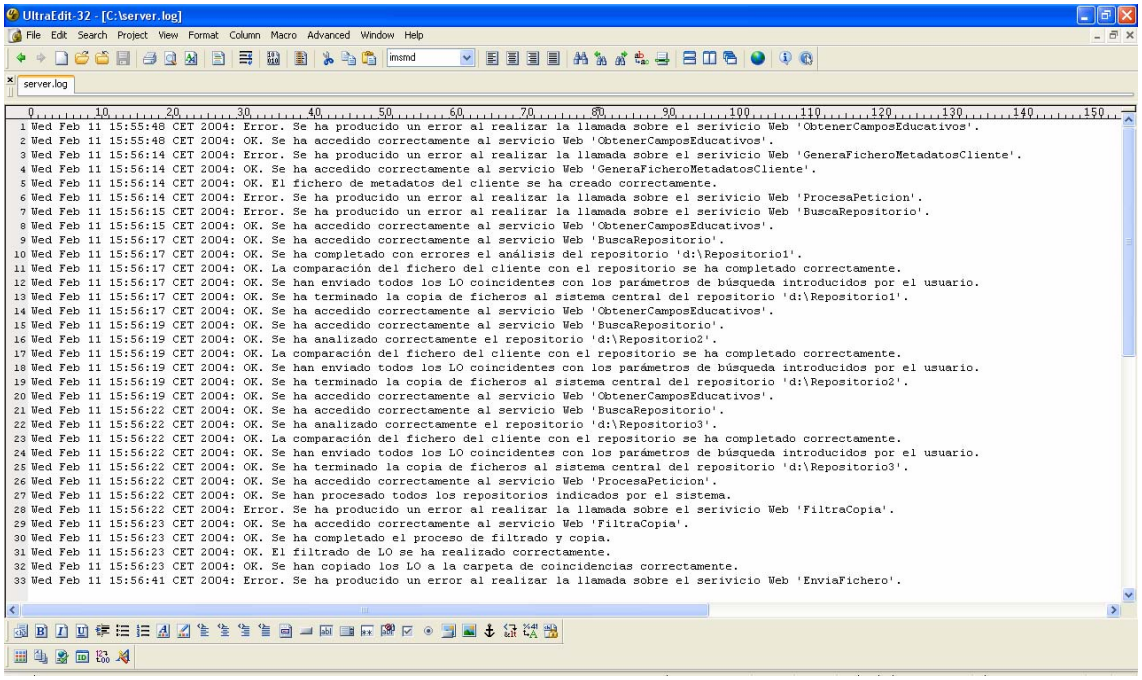

Figura 5.16 Fichero log del sistema

En la figura 5.16 se puede observar que la primera llamada al servicio Web ha resultado fallida; mientras que en el segundo intento se ha conseguido acceder

correctamente al servicio. Así, de esta manera, se ha comprobado como el sistema también es capaz de reaccionar correctamente ante estas posibles situaciones.

Si, por el contrario, aún buscando una posible actualización del servicio, no se es capaz de acceder correctamente al servicio indicado, el sistema retornará un error como el que se muestra en la figura 5.17.

Figura 5.17 Error de acceso a un repositorio

5.3.2 Catalogación de repositorios

Una vez vistas las interfaces de usuario referidas a la búsqueda federada por los diferentes repositorios, se describen los desarrollos para validar el correcto funcionamiento de la opción de registro y catalogación de nuevos repositorios que se ofrecía en la página principal de la aplicación. Cuando el usuario elige dicha opción, el sistema le muestra una ventana como la de la figura 5.18.

Figura 5.18 Registro de Repositorios

El sistema presenta tres opciones diferentes para el registro de repositorios: las dos primeras, implican, además, el registro de un servicio Web (el que se encontraría asociado al repositorio), sobre el UDDI especificado; mientras que la tercera asume que el servicio ya se encuentra registrado sobre un UDDI, y por lo tanto no es necesario su registro.

La primera de las opciones asume que el usuario no tiene ninguna cuenta de usuario sobre un UDDI externo al SROA, y por lo tanto, desea registrar su servicio Web de búsqueda sobre el UDDI del sistema; para ello, la aplicación le pedirá solamente el nombre del servicio Web y su URL, como podemos comprobar en la figura 5.19.

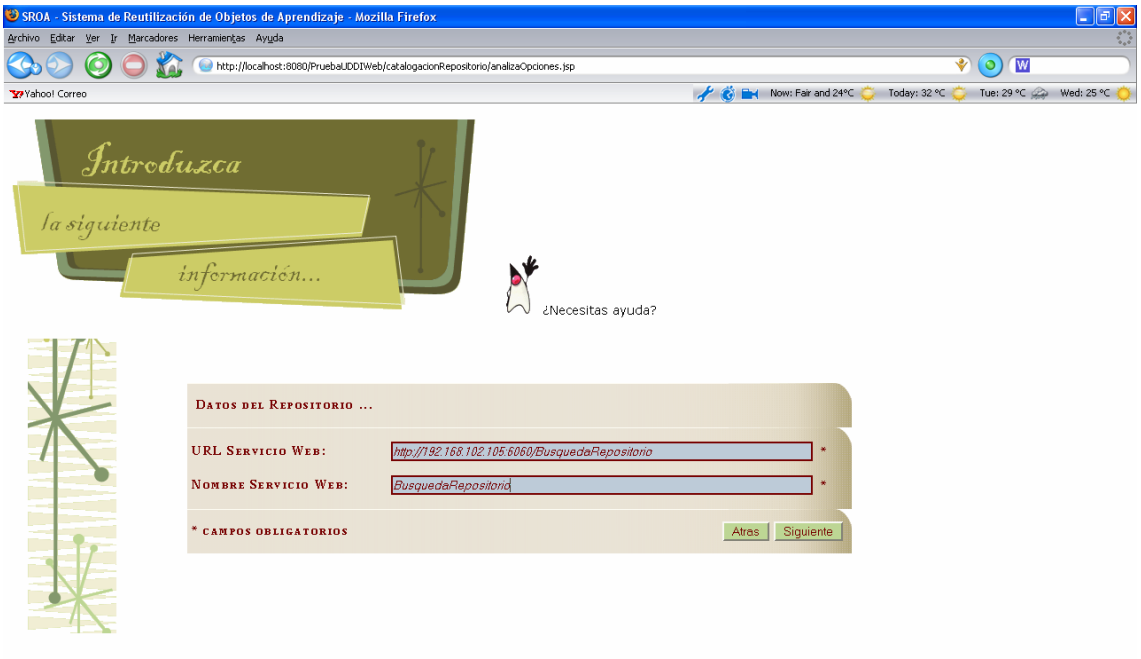

Figura 5.19 Introducción de los datos del Repositorio

Tras introducir los datos, el sistema se conectará al UDDI interno de SROA, para mostrar al usuario todos los datos de su servicio; como, por ejemplo, podría ser la clave del servicio generada por el UDDI, versión del UDDI sobre el que se ha publicado, así como sus URL de búsqueda, publicación y seguridad (sólo en caso de ser UDDI V3); para que, de esta manera, el usuario confirme estos datos. Podría darse el caso de que ya hubiera otro servicio Web en el sistema con el mismo nombre; entonces el sistema en la confirmación de datos, mostraría un menú desplegable con las diferentes claves de los servicios que coincidan con estos parámetros, para que el usuario seleccione la correcta para su almacenamiento en la base de datos. El resultado de esta interfaz se muestra en la figura 5.20.

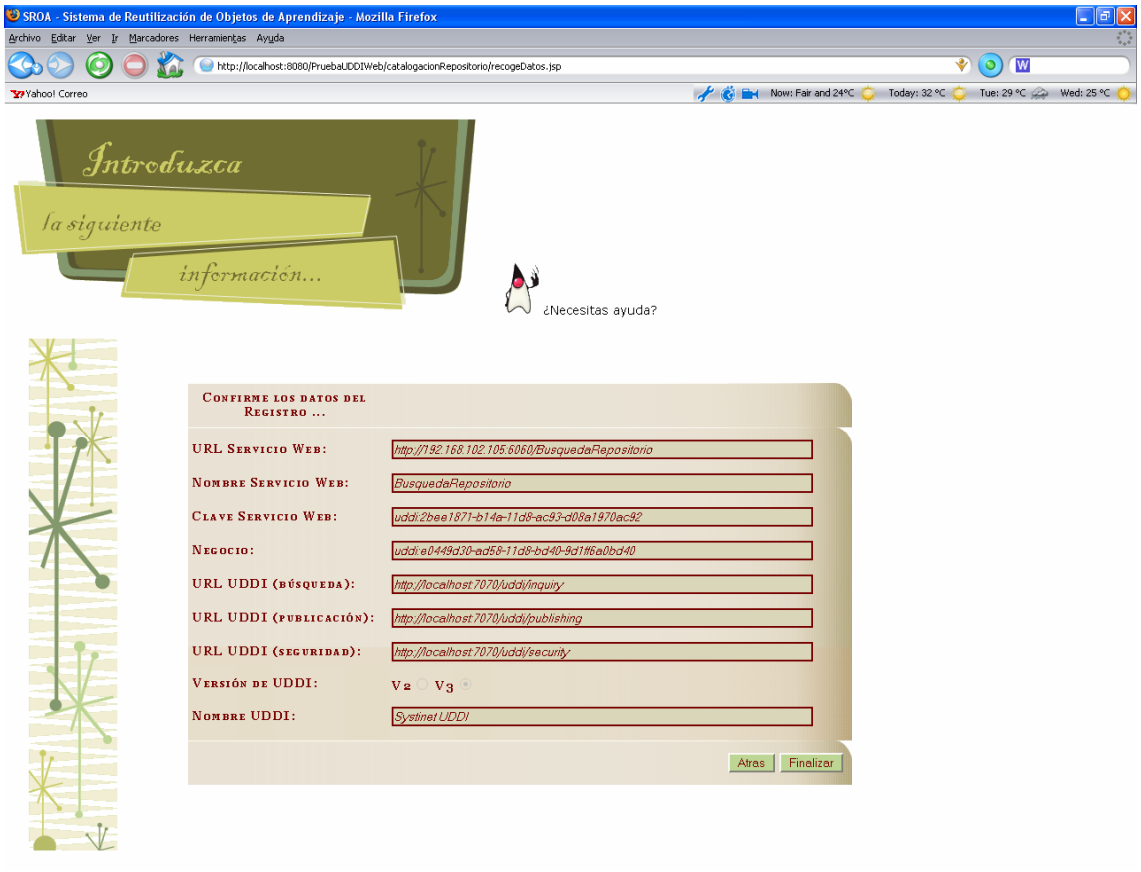

Figura 5.20 Confirmación de los datos del Repositorio

Una vez verificados todos estos datos, el sistema procederá a su almacenamiento en la base de datos, apareciendo la pantalla de la figura 5.21

Figura 5.21 Resultado de la operación

Si el usuario no deseara registrar su servicio sobre el UDDI del sistema, sino sobre otro externo a este, podría realizar esta tarea a través de la segunda de las opciones. Para ello, el usuario tendría que proporcionar más información que en el caso anterior. Estos datos serán los que aparecen en la figura 5.22, es decir: el nombre del servicio Web, así como su URL; las URL de búsqueda y publicación del UDDI donde desea publicarlo (si este fuera UDDI V3, también tendría que proporcionar la URL de seguridad); el nombre del negocio sobre el que va a publicar el servicio Web (en realidad haría falta la clave del negocio, pero por sencillez, el sistema está pensado para que a partir del nombre del negocio, se le proporcione al usuario su clave asociada); así como un usuario y su contraseña, ya que como ya se comentó en los apartados anteriores, para publicar un servicio en un UDDI es necesario estar registrado en éste.

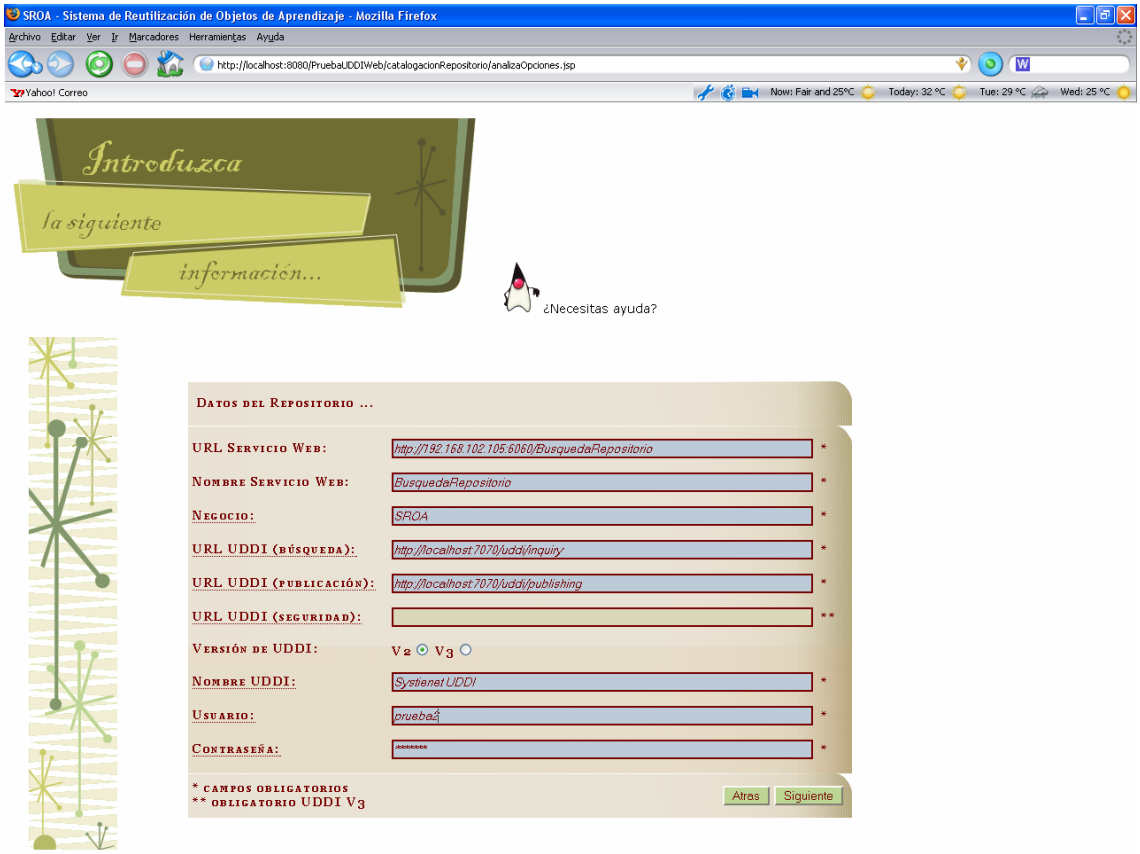

Figura 5.22 Introducción de los datos del Repositorio

Una vez rellenados estos datos, el sistema se conectará al UDDI especificado (a su URL de búsqueda), y a partir del nombre de negocio, le devolverá una lista con todas las claves de los negocios que tengan ese nombre. De esta manera el usuario no tendrá que recordar todo el conjunto de caracteres que conforman la clave. Esta interfaz se muestra en la figura 5.23.

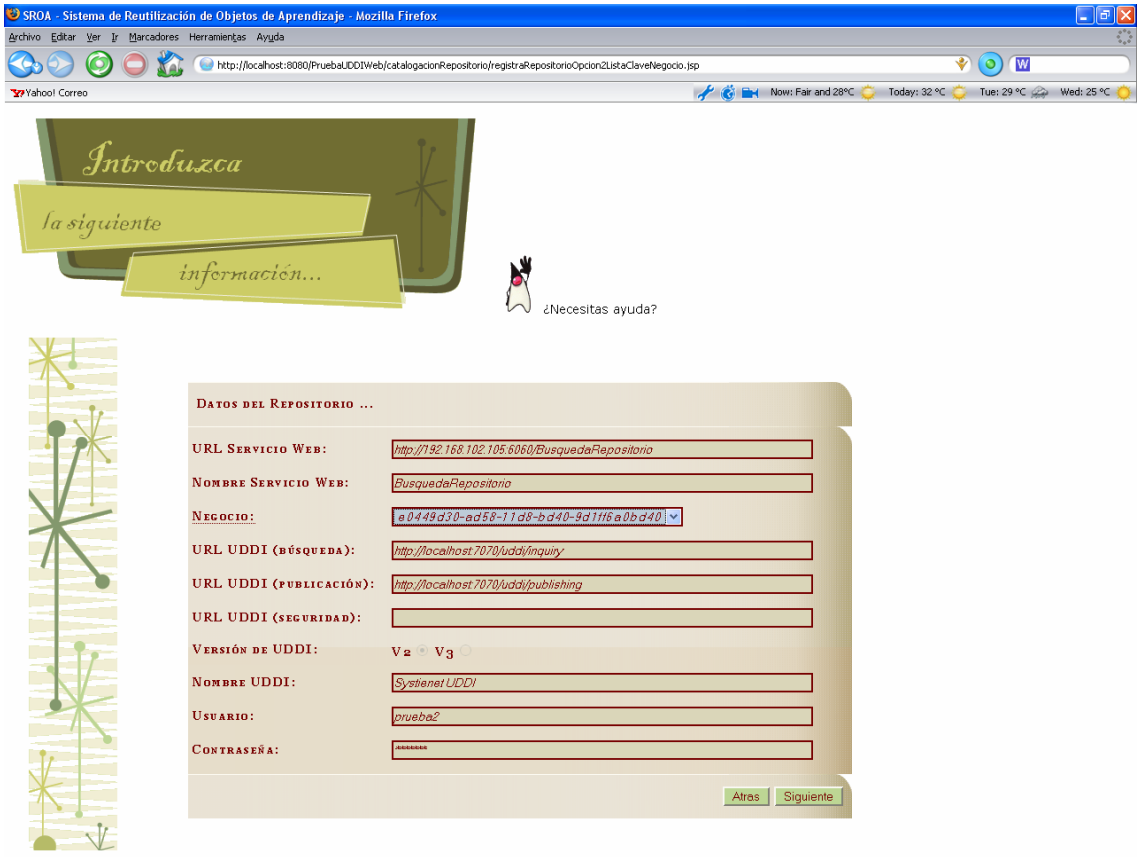

Figura 5.23 Confirmación de la clave de Negocio

Cuando el usuario confirme la clave del negocio sobre el que se va a publicar, el sistema, igual que se indicó en la opción anterior, le mostrará todos los datos de su registro para la confirmación del usuario. En este caso, hemos seleccionado un servicio Web que tiene el mismo nombre que en el anterior caso, por lo que el sistema para su almacenamiento en la base de datos, desconoce cuál es la clave del servicio asociada, por lo que necesita la intervención del usuario. Un ejemplo como éste puede verse en la figura 5.24, en la que se le pide al usuario que confirme cuál es la clave del servicio.

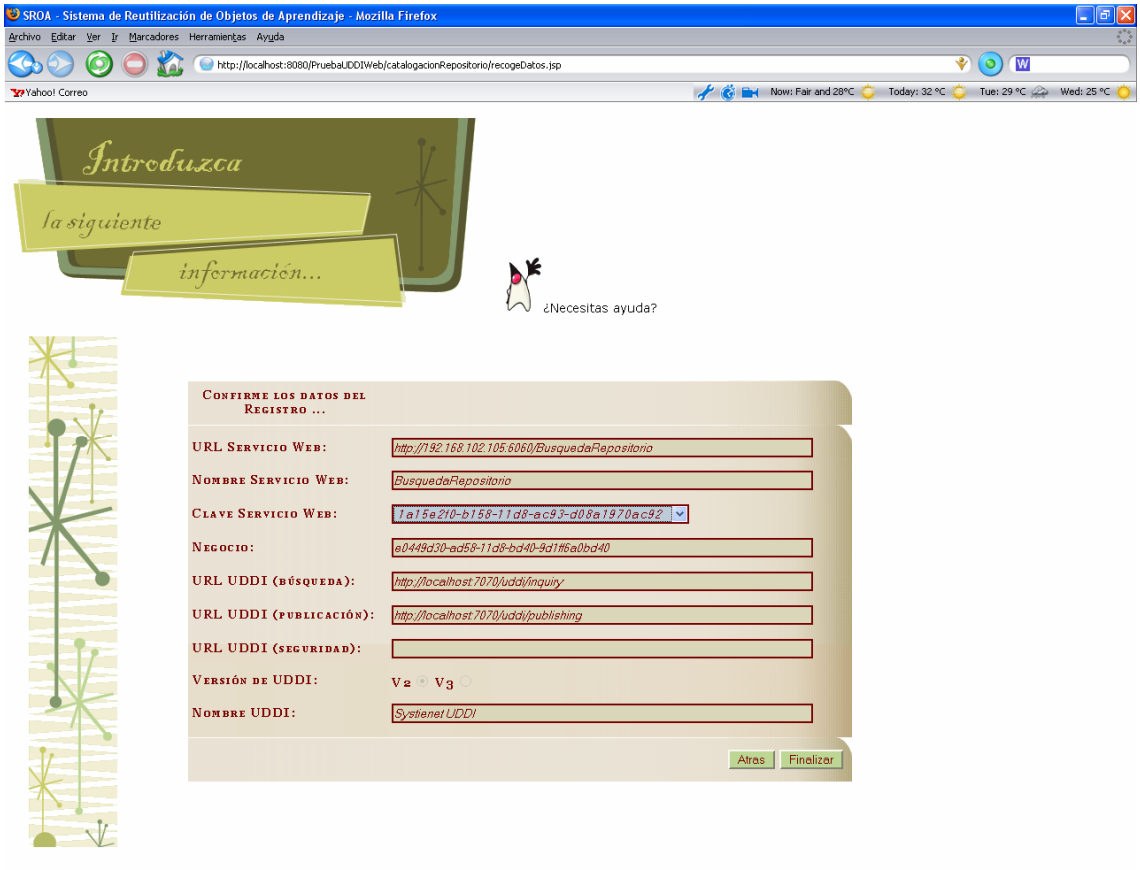

Figura 5.24 Confirmación de la clave de Servicio

Dicha clave de servicio, evidentemente, tendrá el formato del tipo de UDDI sobre el que se vaya a registrar. En caso de tratarse de UDDI V3, la clave comenzará por "uddi:", mientras que en UDDI V2 no, como se puede comprobar en la figura 5.24. Una vez confirmados estos datos, el sistema procederá a su incorporación a la base de datos del sistema (figura 5.25).

Figura 5.25 Resultado de la operación

La última de las opciones que se permiten durante el registro y catalogación de nuevos repositorios es la de añadir un servicio Web de búsqueda, asociado a un repositorio, al sistema, pero que ya se encuentra dado de alta en un UDDI.

En este caso, los datos que se le pedirán al usuario serán: el nombre y la URL del servicio Web, el nombre del negocio y la URL de búsqueda del UDDI sobre el que está publicado, así como la versión del mismo. Estos datos, se pueden ver en la figura 5.26.

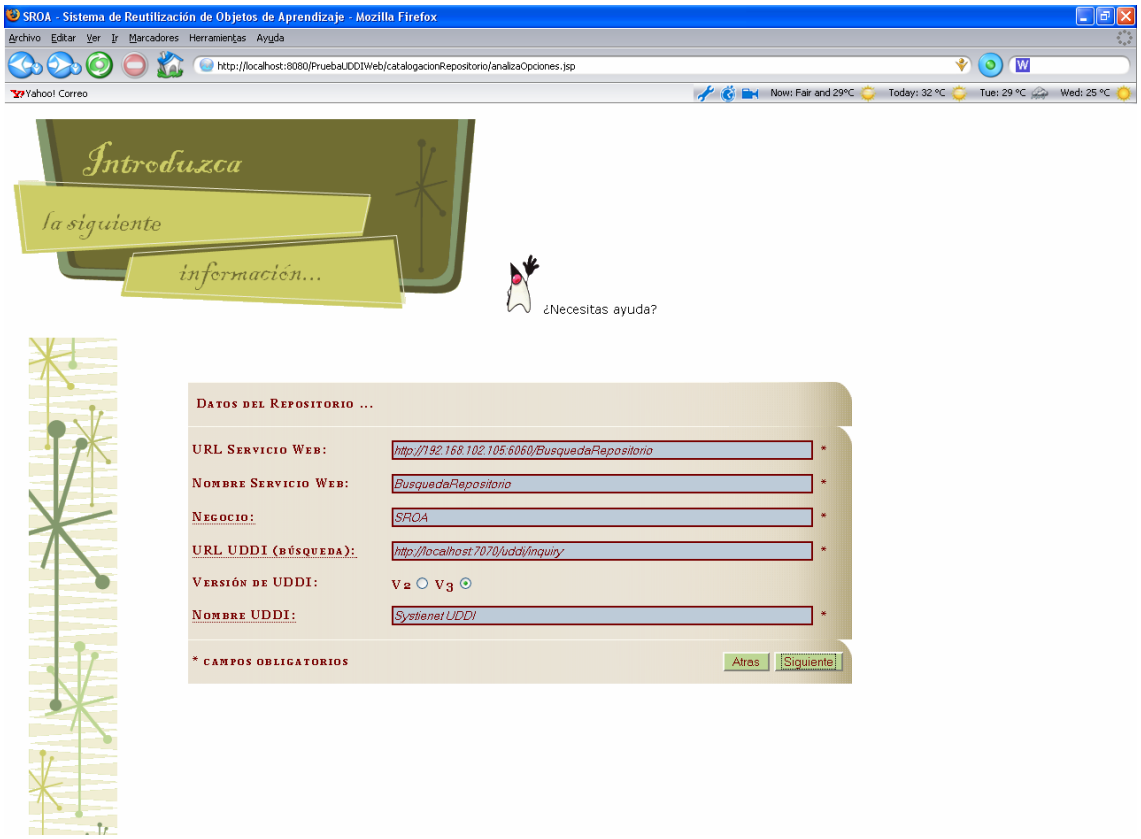

Figura 5.26 Introducción de los datos del Repositorio

En este caso, se solicita incorporar el servicio Web a la base de datos del sistema, suponiendo que ya se encuentra registrado en un UDDI especificado por su URL de búsqueda. El UDDI especificado es V3, para que de esta manera se vea cómo se buscan las claves para esta versión. Al igual que se comentó en el caso anterior, el usuario introduce el nombre del negocio sobre el que se encuentra publicado el servicio, y el sistema le busca su clave (V3 en este caso), como se puede ver en la figura 5.27.

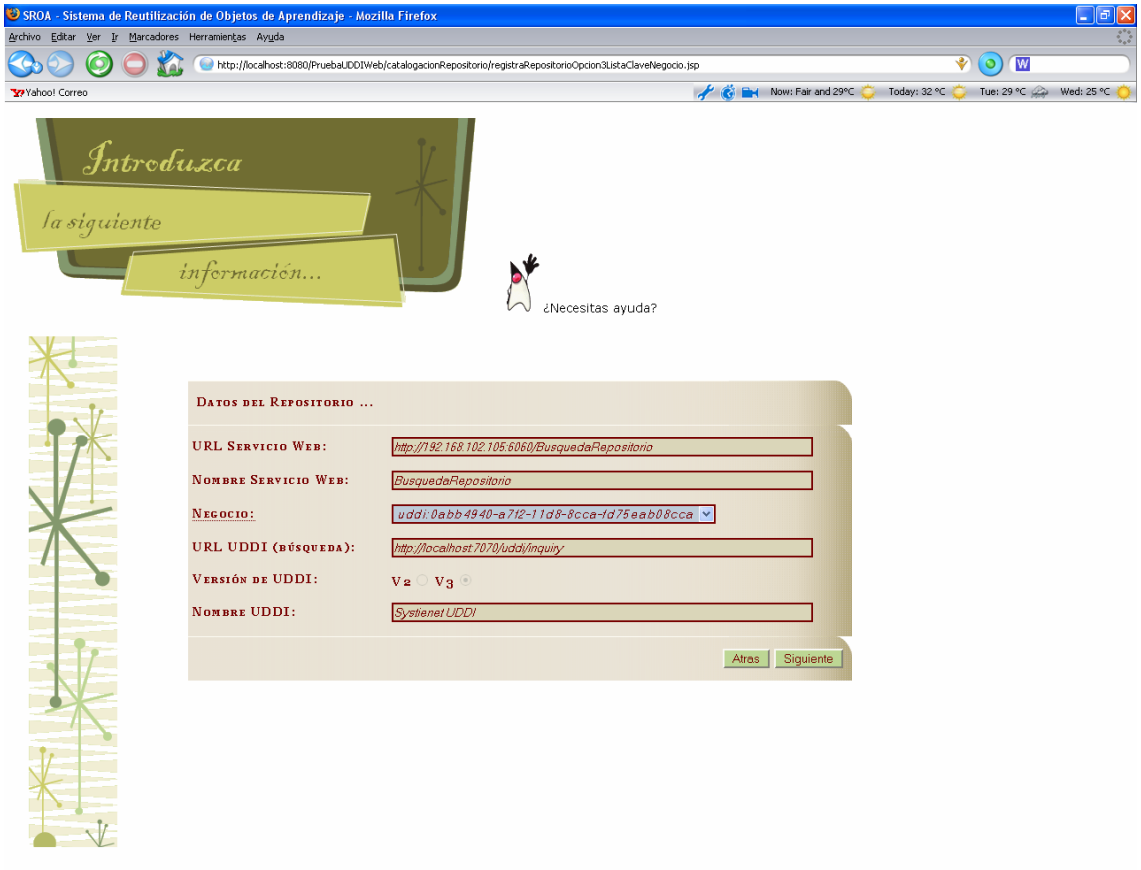

Figura 5.27 Confirmación de la clave de Negocio

A continuación, el usuario confirmará los datos del servicio. Estos datos se extraerán del UDDI sobre el que está publicado. Como en este caso el servicio seleccionado es el mismo que en las opciones anteriores, el sistema pedirá la intervención del usuario para la confirmación de la clave del servicio, como ocurriera en el caso anterior; por lo que la interfaz que mostraría el sistema, sería como la que se muestra en la figura 5.28.

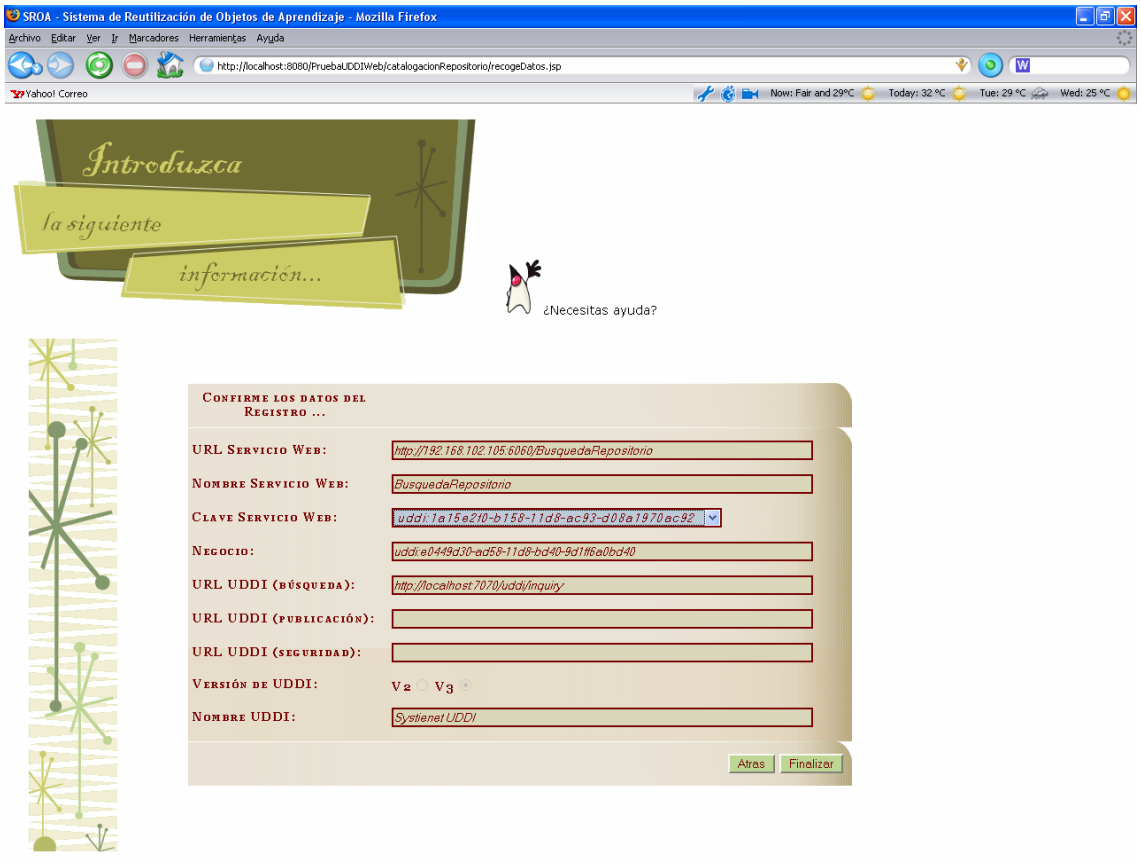

Figura 5.28 Confirmación de la clave de Servicio

Una vez confirmada la clave del servicio (en este caso clave con formato UDDI V3), éstos pasarán a formar parte de la base de datos del sistema. El sistema notificará el resultado de la acción, con una ventana similar a las anteriores, pero en este caso informándole solamente que los datos se han insertado en la base de datos (figura 5.29).

Figura 5.29 Resultado de la operación

Estas serían las tres opciones permitidas en cuanto al registro de nuevos repositorios en el sistema. No se han mostrado las ventanas de todos los posibles errores que se han tenido en cuenta; como, por ejemplo, errores relacionados con los accesos a los UDDI correspondientes, bien por accesibilidad, permisos sobre el UDDI o sobre el negocio a publicar; así como las restricciones del mismo. Otro tipo posible de errores a tener en cuenta, son los relacionados con la base de datos del sistema: accesibilidad, integridad de los datos, o posibles caídas temporales del gestor de base de datos. Todos estos posibles errores, serán gestionados correctamente por el sistema, garantizando en todo momento la integridad de los datos.

5.4 CONCLUSIONES

Con el desarrollo de este prototipo se ha demostrado que llevar a la práctica la arquitectura propuesta es posible y que, por tanto, la realización de sistemas similares es una realidad.

Como se ha visto, la incorporación de un nuevo repositorio al sistema es una tarea relativamente sencilla y esto hace que el sistema sea altamente interoperable. No solo hay que pensar en la incorporación de repositorios, sino que cualquier tipo de sistema que contenga contenidos docentes esta indicado para incorporarse al sistema, ya que sólo tiene que crearse un servicio Web y registrarlo en SROA para que su contenido sea accesible desde otros sistemas.

Avanzando un poco más en la idea de interoperabilidad, también sería posible la comunicación entre diversos sistemas como el desarrollado; simplemente publicando el servicio de búsqueda federada en un registro UDDI para darle acceso exterior, y de esta forma, podríamos comunicarnos con él a través de otros sistemas y realizar búsquedas federadas múltiples. La idea podría ser como la presentada en la figura 5.30.

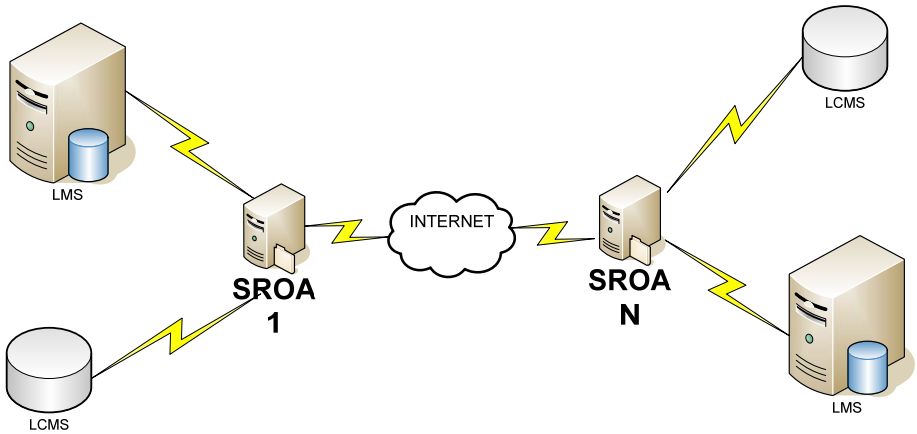

Figura 5.30 Comunicación entre distintos sistemas SROA

6 CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

En este capítulo se recogen las conclusiones de la investigación realizada; muchas de ellas, aunque no todas, ya explicadas en los diversos apartados de conclusiones parciales de los capítulos anteriores. Se plantearán algunas futuras líneas de investigación que han surgido del trabajo y que supone, la evolución de la arquitectura y el prototipo construido. Antes de acometer estos apartados enumeraremos los objetivos que perseguíamos al comienzo del trabajo para determinar dónde se han logrado, así como las aportaciones principales derivadas del esfuerzo investigador.

6.1 OBJETIVOS Y APORTACIONES

Conviene recordar el objetivo inicial que nos propusimos lograr al iniciar este trabajo de investigación, y que enunciábamos de la manera siguiente:

> ***Creación de una arquitectura software orientada a servicios, e implementada mediante servicios Web, capaz de cubrir la necesidad de reutilizar objetos de aprendizaje almacenados en diferentes repositorios, que permitirá localizarlos independientemente de su ubicación física y de su tecnología de almacenamiento a través de Internet de manera que haga que los sistemas de teleformación sean interoperables.***

Para alcanzar este objetivo se plantearon otros objetivos intermedios. En la tabla 6.1 se muestran estos objetivos y en qué capítulo de la tesis se ha conseguido satisfacer dicho objetivo.

OBJETIVO	CAPÍTULO
1. Definir una arquitectura orientada a servicios para la localización universal de objetos de aprendizaje.	Capítulo 4
2. Describir cada uno de los servicios presentes en la arquitectura y su organización en la misma.	Capítulo 4
3. Estudiar las características de los repositorios existentes en la actualidad y su adaptabilidad a las especificaciones y estándares.	Capítulo 2 y 3
4. Facilitar la interoperabilidad de sistemas de teleformación aunque utilicen distintos estándares de metadatos de los objetos docentes que intercambian.	Capítulo 4
5. Ofrecer una capa de abstracción superior con servicios comunes disponibles para todos los sistemas que cubran sus necesidades de comunicación.	Capítulo 4
6. Basar la propuesta en estándares reconocidos y sólidos.	Capítulo 2 y 4
7. Construir un prototipo basado en la arquitectura propuesta.	Capítulo 5

Tabla 6.1. Objetivos de la tesis y capítulo donde se justifica su cumplimiento

Las aportaciones principales que se han llevado a cabo a lo largo de la investigación se pueden resumir en los siguientes artículos enviados a diversos congresos, tanto nacionales como internacionales:

- ✓ Otón ,S., Gutierrez, J.M., Barchino, R., Hilera, J.R., Macias, J., "*Sistema de almacenamiento y gestión de documentos para el desarrollo de bibliotecas digitales*". III Jornadas de Bibliotecas digitales. JBIDI 2002.

Esta primera aportación a congreso fue resultado de la investigación sobre bibliotecas digitales, tema inicial en el que el autor de la tesis empezó a trabajar y que posteriormente relacionó con el mundo del e-learning, como se puede apreciar en el siguiente trabajo.

- ✓ Otón, S., Gutiérrez, J.M., "*Creación de una biblioteca digital para la gestión de unidades de aprendizaje para cursos de teleeducación*". 3ra. Conferencia Iberoamericana en Sistemas, Cibernética e Informática. CISCI 2004.

En esta aportación se realizaba un estudio de la utilización de bibliotecas digitales en el apoyo a la teleformación mediante el almacenamiento y gestión de las unidades de aprendizaje. La biblioteca digital era capaz de almacenar información digital heterogénea independiente de los soportes en que se pueda difundir y de su localización.

Se presentaba una serie de servicios que podían ayudar a la gestión de contenidos y su distribución, de forma que la transformación de los contenidos para adaptarlos a las distintas especificaciones de las herramientas de teleformación era la tarea principal de la biblioteca digital.

- ✓ Gutiérrez, J.M., Otón, S., "*Necesidad de estándares en una arquitectura distribuida para sistemas de teleformación*". 3ra. Conferencia Iberoamericana en Sistemas, Cibernética e Informática. CISCI 2004.

En este trabajo se hacía un estudio de las necesidades de estándares de metadatos asociados al material docente para el intercambio de estos contenidos entre distintas herramientas de teleformación. Mediante estos estándares se pueden construir sistemas que sean capaces de interoperar y utilizar de forma inteligente los repositorios y los objetos docentes contenidos en ellos.

- ✓ Otón, S., Gutiérrez, J.M., Barchino, R., "*Biblioteca digital de objetos de aprendizaje*". Conferencia Iberoamericana WWW/Internet 2004. CIAWI 2004.

Este trabajo fue una evolución del primero, ya que proponía la utilización de bibliotecas digitales como medio de almacenamiento de los contenidos docentes que podían ser utilizados por diversas herramientas de teleformación, proporcionando la conversión necesaria entre estándares. El trabajo incluía un estudio de la utilización de servicios Web para el acceso a los contenidos docentes almacenados en la biblioteca digital.

- ✓ Ortiz, A., Otón, S, Barchino, R., "*Arquitectura para publicación y localización universal de Objetos de Aprendizaje mediante Servicios Web*". I Congreso IberoAmericano sobre Computación Ubicua (CICU 05). 2005.

En este trabajo se presentaron las primeras ideas y características que formarían parte de la arquitectura propuesta en la tesis. Se plantearon las ideas de la utilización de los servicios Web asociados a cada repositorio para dar acceso a sus contenidos, y una forma de realizar conversiones de metadatos utilizando Dublin Core como formato intermedio.

- ✓ Otón, S., Hilera, J.R., Gutiérrez, I., Ortiz, A., "*Arquitectura orientada a servicios Web para la implementación de repositorios distribuidos de objetos y unidades de aprendizaje*". I Simposio Nacional de Tecnologías de la Información y de las Comunicaciones en la Educación (SINTICE 2005).

En este trabajo se presentó la primera versión de la arquitectura propuesta en la tesis. Donde se planteaba la utilización de los servicios Web y SOA para la construcción de la arquitectura.

- ✓ Otón, S., Jiménez, M.L., Barchino, R., Hilera J.R., Gutiérrez J.M., "*Web services oriented architecture for the implementation of learning objects distributed repositories*". IADIS International Conference Applied Computing 2006.

En este trabajo se presento un resumen de la arquitectura propuesta en la tesis ya que se trataba de un póster.

- ✓ Otón, S., Ortiz, A., Barchino, R., "*Arquitectura orientada a servicios para la reutilización de objetos de aprendizaje distribuidos*". IADIS Conferencia Ibero Americana WWW/Internet 2006.

Este es el último trabajo relacionado con la arquitectura mandado a un congreso, en él se explica la última versión de la arquitectura así como el SROA implementado con toda su funcionalidad.

- ✓ Otón, S., Ortiz, A., "*Búsqueda de objetos de aprendizaje para dispositivos móviles en repositorios distribuidos*". II Congreso IberoAmericano sobre Computación Ubicua (CICU 06). 2006.

En este trabajo se propone la utilización del prototipo implementado en la tesis para la integración de dispositivos móviles en el proceso de búsqueda de contenidos docentes distribuidos. Se realizará un estudio de esta idea en el apartado de futuras líneas de investigación.

- ✓ Ortiz, A., Otón, S., Hilera J.R., "*Hacia la utilización de Servicios Web Semánticos en Repositorios de Objetos de Aprendizaje*". III Simposio Pluridisciplinar sobre Objetos y Diseños de Aprendizaje Apoyados en la Tecnología (od@06). 2006.

En este trabajo se describen diferentes tecnologías que pueden mejorar las propiedades de descubrimiento, accesibilidad y reutilización universal de objetos de aprendizaje almacenados en repositorios. Se plantea la posibilidad de incorporar mecanismos de búsqueda semántica a dos niveles: a nivel del propio repositorio basado en servicios Web, y a nivel de los registros UDDI que describen los componentes de dichos servicios Web.

Sin lugar a dudas la mayor aportación de la tesis la tenemos en dos proyectos de investigación PROFIT: "FIT-350101-2004-7" denominado "*Plataforma para la Gestión y Explotación de Recursos Educativos Virtuales*", y "FIT-350101-2005-4" denominado "*Sistema para la Publicación y Localización Universal de Objetos de Aprendizaje*" del que el autor de esta tesis forma parte del equipo investigador, coordinado por el Dr. D. José Antonio Gutiérrez de Mesa.

El primer PROFIT tenía como objetivo el desarrollo de un sistema informático que permitiera a los usuarios la utilización de un entorno virtual de aprendizaje. Éste proyecto se subdividía, a su vez, en dos subsistemas:

- Gestor del Aprendizaje: Para el control del acceso a los contenidos y gestión de la plataforma.

- Gestor de Recursos: Repositorio de contenidos docentes virtuales, además de permitir la realización de búsquedas de esos contenidos.

El segundo PROFIT tiene como objetivo fundamental el análisis y diseño de un prototipo de sistema para la construcción de cursos a través de la reutilización de contenidos docentes distribuidos en diversos repositorios. Además se ha estudiado una nueva forma de especificar los contenidos docentes independientemente de la propia visualización y la incorporación de herramientas de comunicación a los sistemas de teleformación.

6.2 CONCLUSIONES

Partiendo del objetivo principal del trabajo de investigación podemos decir que lo pretendido inicialmente se ha logrado de una manera efectiva: hemos realizado una propuesta de Arquitectura multicapa, demostrando la validez de la propuesta con la construcción de un prototipo que implementa un sistema concreto basado en dicha Arquitectura. El prototipo realizado es útil y valioso, ya que demuestra que se puede llevar a la práctica la arquitectura y de esta forma permitir la interoperabilidad entre distintos sistemas de formación y repositorios distribuidos.

La arquitectura propuesta puede resolver los problemas de reutilización de los objetos de aprendizaje mediante su publicación y localización universal. Esto permite distribuir contenidos educativos entre distintas plataformas de e-learning y hacer interoperables los repositorios de las mismas.

Además de todo ello, a lo largo del tiempo en el que se ha realizado la investigación, se ha obtenido una serie de conclusiones como consecuencia directa de la misma y de

los problemas que se han tenido que solventar en la construcción del prototipo. Vamos seguidamente a ver cuales son estas conclusiones:

1. Como ha quedado reflejado en el capítulo 2, hemos llevado a cabo un análisis exhaustivo de la situación actual relacionada con los estándares y recomendaciones de teleformación. Sobre la base de estas experiencias, mantenemos el criterio de que la utilización de estándares, a ser posible "de facto" y no solamente "de jure", ha sido siempre beneficiosa para el desarrollo y consolidación de tecnologías emergentes. Además de ayudar a la interoperabilidad entre sistemas, establecen marcos comunes de actuación que permiten aprovechar los avances de los diferentes equipos de investigación, así como la reutilización de contenidos en otros sistemas de parecidas características.

   En la medida en que seamos capaces de difundir la utilización de los estándares ya existentes y de desarrollar nuevas y útiles recomendaciones de uso y utilización de los mismos, se crearán mejores condiciones para la implantación de tecnologías que ayuden a la mejora de la enseñaza y del aprendizaje.

   Como se ha comentado a lo largo del trabajo, para que un objeto de aprendizaje sea reutilizable necesitamos que su construcción se haya realizado conforme a estándares o especificaciones como los establecidos por IMS, ADL (SCORM) o IEEE (LOM). De esta forma aseguramos que, a través de su empaquetamiento y descripción mediante metadatos, pueda ser integrado en cualquier plataforma de e-learning compatible con estos estándares. Sin embargo, al existir varios estándares sería recomendable el desarrollo de uno único que simplifique la interoperabilidad entre distintos sistemas de teleformación o repositorios. Como actualmente esto no ocurre y debido a la variedad de estándares y especificaciones existentes, nace pues, de manera casi natural, la necesidad de tener utilidades y herramientas que permitan la adaptación tanto de los contenidos docentes como de la metainformación que se gestiona en el proceso de aprendizaje.

   Esta es una las características que hemos integrado en la arquitectura y que permite desarrollar sistemas capaces de realizar las conversiones necesarias de formato de metainformación de forma transparente al usuario.

   Sería deseable que las herramientas de preparación y creación de contenidos, cuyo número es muy elevado, incorporaran utilidades o herramientas que, centrándose en aquellos estándares que se perfilen como realmente utilizados, es decir, que lo sean "de facto", permitan, incluso también de una manera

guiada, la adaptación de los contenidos a dichos estándares. Y siguiendo con esta idea, permitir que los repositorios trabajen con diversos estándares "de facto" para conseguir la integración de contenidos educativos heterogéneos.

2. Siguiendo con la idea de la reutilización de los objetos de aprendizaje, además de estar construido siguiendo los estándares, debe ser localizable universalmente. De poco sirve un objeto de aprendizaje con un alto nivel de calidad si sólo es accesible por unos cuantos usuarios de una determinada plataforma o repositorio. Las instituciones educativas requieren mecanismos de interoperabilidad, ya que sería muy costoso quedar con contenido aislado en un mundo cada vez más interconectado y que clama por la colaboración institucional como mecanismo para garantizar una educación de calidad.

   Si queremos que los objetos de aprendizaje sean reutilizables por un número potencialmente alto de usuarios, debemos integrarlos en un sistema capaz de localizarlos y exponerlos a estos usuarios y sistemas. De esta forma también los proveedores de contenido podrán fácilmente reutilizarlos y distribuirlos entre diferentes sistemas.

   Pero para que esto ocurra no podemos obligar a que los actuales sistemas de teleformación o los repositorios existentes modifiquen su estructura y funcionamiento. Debemos proporcionarles los mecanismos necesarios para que la incorporación de esta funcionalidad sea lo menos traumática posible. De ahí la utilización de los servicios Web y las arquitecturas orientadas a servicios, tecnologías que han demostrado su principal valía en la integración de aplicaciones, de forma que añadimos cierta funcionalidad sin modificar lo ya desarrollado.

3. En cuanto a la definición de arquitecturas de sistemas de e-learning se ha podido comprobar que la implantación de arquitecturas orientadas a servicios (como la presentada en la tesis) será una realidad inmediata. La propia definición del *Abstract Framewok* de IMS lo demuestra, y ellos mismos examinan un conjunto de arquitecturas actuales para desarrollar la suya. De esta forma conseguiremos que en un futuro los sistemas de teleformación desarrollados siguiendo este tipo de arquitectura sean capaces de integrarse con otros sistemas de una manera muy sencilla y por lo tanto realizar sistemas interoperables. Esto se conseguirá con la integración de servicios de integración capaces de comunicarse o compartir información con otros sistemas.

Como se explicó en las conclusiones del capítulo tres, se quería desarrollar una arquitectura que tuviera una serie de características, a continuación presentamos estas características y las razones por las que se han conseguido integrar.

- **Abierta.** La arquitectura planteada permite la creación de sistemas e-learning interoperables y conectables entre sí de forma sencilla; es decir, que sistemas y herramientas comerciales de fabricantes distintos puedan ensamblarse en un único sistema global. Esto se ha conseguido utilizando SOA y servicios Web.

- **Escalable.** La arquitectura está definida de tal forma que permite su crecimiento. Este crecimiento se puede ver desde dos perspectivas. Por un lado tenemos el crecimiento de datos, representados por los nuevos objetos de aprendizaje que se incorporarían al sistema al incluir un nuevo repositorio. Y por otro lado tenemos el crecimiento en funcionalidad que se daría al incluir nuevos servicios a la arquitectura, tarea relativamente sencilla dada la naturaleza del modelo SOA aplicado en su desarrollo.

- **Global.** Permite la diversidad lingüística y cultural. El sistema resultante de la implementación de la arquitectura puede presentarse en diferentes lenguas en función del usuario a quien esté destinado. También es capaz de localizar contenidos docentes independientemente del lenguaje en el que estén realizados.

- **Integrada.** Un sistema desarrollado siguiendo la arquitectura propuesta es capaz de integrarse con una gran cantidad de sistemas de e-learning existentes hoy en día (y en un futuro), consiguiendo la interoperabilidad entre todos ellos. Para conseguirlo, tan solo se tiene que desarrollar un servicio Web que de acceso a sus contenidos y registrarlo en nuestro sistema.

- **Flexible.** Nuestra arquitectura tiene la capacidad de implementar nuevas soluciones sin tener que efectuar grandes cambios en la misma. Como se comentó anteriormente, esto se consigue al haber realizado una arquitectura orientada a servicios en la que la inclusión de nuevos servicios no resulta una tarea traumática, sino todo lo contrario.

A modo de resumen podemos presentar como características más destacadas de la arquitectura y el prototipo desarrollado, las siguientes:

- ✓ Presenta una arquitectura abierta y utiliza protocolos y formatos estándar para permitir la interoperabilidad e integración de plataformas de aprendizaje y repositorios.

- ✓ La posibilidad de publicar y descubrir cualquier objeto de aprendizaje independientemente de su localización y sus metadatos.

- ✓ Presenta una forma uniforme y bien definida de acceso a los objetos de aprendizaje.

- ✓ Hace que los objetos de aprendizaje sean reutilizables.

Por último, destacar que todas estas conclusiones, conducen directamente al resultado de que la arquitectura propuesta es necesaria y deseable en el desarrollo actual de sistemas de e-learning, y plantea soluciones interesantes y viables para la interoperabilidad de sistemas heterogéneos y la reutilización de contenidos docentes.

Además, el desarrollo de la arquitectura se ha basado en estándares y especificaciones actuales que la convierten en una opción sólida y que ayuda a la consolidación de normas.

En la evolución de los sistemas de e-learning se está tendiendo hacia la idea de interoperabilidad e interconexión planteada en esta tesis, por lo tanto creemos que nuestra arquitectura puede suponer un gran avance en este sentido.

6.3 FUTURAS LÍNEAS DE INVESTIGACIÓN

Como continuación del trabajo de investigación objeto de esta tesis, en la actualidad se han abierto varias líneas de investigación en las que se está ya trabajando. La primera y más avanzada es la integración de dispositivos móviles en la Arquitectura propuesta en esta tesis. La segunda es la incorporación de la Web semántica y las ontologías como forma de mejorar las búsquedas. La última es la integración de agentes inteligentes para la búsqueda personalizada y adaptada a las necesidades de los usuarios.

Los objetivos que nos hemos planteado en este sentido, y una descripción de estas líneas de investigación se detallan a continuación.

6.3.1 Integración de dispositivos móviles en la arquitectura

La educación apoyada por la tecnología ha pasado por diferentes fases en los últimos años como se explicó en el capítulo dos: EAO (Enseñanza Apoyada por el Ordenador), multimedia educativo, tele-educación, enseñanza basada en Web (web-based teaching), aprendizaje electrónico (e-learning), etc. Desde el apoyo simple del ordenador y los soportes multimedia, se ha ido incorporando el uso de la red en general, y de Internet y

las tecnologías Web en particular, para apoyar el proceso de enseñanza-aprendizaje en sus diferentes modalidades y aspectos. Recientemente, se incorporan a este panorama las tecnologías móviles, dando lugar a lo que se ha dado en llamar "*mobile learning*" o "*m-learning*".

El uso en educación de pequeños dispositivos móviles, tales como teléfonos móviles, agendas electrónicas, pero también *tablet PCs*, es un tema que está levantando gran expectación en la actualidad y sobre el que se están realizando interesantes iniciativas empresariales y proyectos de investigación. M-learning emerge para satisfacer las necesidades individuales de diferentes usuarios, permitiéndoles el acceso a información específica desde cualquier lugar en cualquier momento y en cualquier dispositivo [Sharma, 2004].

Por lo tanto, surge la necesidad de que el proceso educativo se pueda adaptar a diversos tipos de dispositivos más o menos flexibles y potentes. Un nuevo reto que surge es la adaptación de las interfaces Web que se utilizan en las aplicaciones de teleformación a estas condiciones, cambiantes, flexibles y de máxima disponibilidad.

En el desarrollo de esta línea de investigación estamos también teniendo en consideración el fuerte proceso de estandarización por parte de organismos internacionales que los sistemas de teleformación están sufriendo. Estos procesos están dando como resultado una identificación de necesidades y características deseables y una aportación de soluciones de interacción entre sistemas LMS y entre estos y terceros sistemas.

Una de las principales características que tiene la arquitectura presentada y el prototipo desarrollado, es que está dotado de una interfaz Web y, por lo tanto, es completamente accesible desde cualquier ubicación. En el capítulo cinco, se mostraba el resultado de la interfaz de entrada al sistema, así como el resultado de la búsqueda sobre un equipo convencional. Este sistema es perfectamente accesible no sólo desde un equipo de estas características, sino desde cualquier sistema dotado de un navegador Web, como se puede apreciar en la figura 6.1, en la que se muestra un ejemplo de acceso al sistema desde una PDA dotada con conectividad WIFI y con navegador Web. De esta manera, la reutilización de objetos de aprendizaje se ve claramente reforzada, ya que permitiremos acceder a información docente desde este tipo de terminales, cada vez más utilizados, mejorando con ello la ubicuidad y accesibilidad del sistema.

Figura 6.1. ***Interface de entrada al sistema y resultado de una búsqueda sobre un terminal móvil.***

Otro aspecto destacable, es que uno de los metadatos que se utilizan en el sistema por defecto, es el del formato del contenido educativo del objeto de aprendizaje, como es *format*, dentro de la categoría *technical*. A través de este campo, el usuario podrá buscar solamente aquellos objetos educativos con un formato determinado, para que así estén adaptados a las posibilidades computacionales del terminal sobre el que se vaya a visionar el contenido educativo, mejorando sensiblemente el filtrado sobre este tipo de material. Pero no solamente nos podremos centrar en este campo educativo de LOM previamente seleccionado para el sistema, ya que, según las características de adaptabilidad de nuestra arquitectura, podremos incorporar campos como el tamaño del archivo, requisitos técnicos, tipo de interactividad o coste, también presente en la especificación LOM, para limitar así, determinados objetos de aprendizaje demasiado grandes para nuestro terminal, o para nuestra tasa de conexión en un momento determinado. También serán interesantes para indicar el tipo de recursos con los que contamos en un momento determinado y así evitar buscar objetos de los que no vayamos a poder disfrutar con nuestro terminal.

El sistema desarrollado y la arquitectura en la que se sustenta, permite distribuir contenidos educativos entre distintas plataformas de e-learning y distintos tipos de clientes. Queda patente que una descripción de la metainformación de un objeto de aprendizaje es muy importante a la hora de permitir su búsqueda y por lo tanto su reutilización. También es importante señalar que mediante esta metainformación somos capaces de determinar si un objeto de aprendizaje es válido para ser accedido desde distintos tipos de dispositivos.

Seguimos trabajando para que la arquitectura y los sistemas desarrollados permitan una mejor adaptación a este tipo de dispositivos incorporando mecanismos de presentación y filtrado de contenidos para su adaptación a sus limitaciones propias.

6.3.2 Incorporación de Web semántica y ontologías en la arquitectura

Los Servicios Web, como ya se ha visto, permiten a las organizaciones encapsular sus procesos de negocio, publicarlos como servicios, suscribirse a otros servicios e intercambiar información entre organizaciones. La madurez adquirida por esta tecnología, así como su gran expansión, ha hecho que su uso pase de ser una ventaja a convertirse en una necesidad imperiosa del mercado.

Una de las utilidades de esta arquitectura es su aplicación en la industria del e-learning, potenciando así la completa accesibilidad y reutilización de la información docente albergada en distintos repositorios distribuidos por Internet. Un ejemplo de este tipo de arquitectura es la planteada en esta tesis.

- **Web semántica**

En los últimos años, se ha comenzado a experimentar el uso de técnicas semánticas para la representación del conocimiento, y de esta manera, extender la Web a través de metadatos, en la que los ordenadores no sólo serán capaces de presentar toda la información contenida en ella, sino que además podrán entenderla y gestionarla de forma “inteligente”; se trata de la conocida como “Web Semántica” [Berners-Lee et al., 2001].

La mayor cantidad de búsquedas de información que realizan los actuales navegadores, de forma más o menos acertada, es realizada mediante ciertas palabras clave incorporadas en el código HTML de las páginas Web dispersas en Internet. En los últimos años se están realizando anotaciones de datos introducidas dentro de este código HTML, siguiendo algún esquema de anotación común, normalmente basado en XML. La idea es que los datos puedan ser utilizados y “comprendidos” por los ordenadores, sin necesidad de supervisión humana. Se trata de convertir la información en conocimiento, transformando los datos dentro de las páginas Web por metadatos con un esquema común consensuado sobre algún dominio, llegando a conseguir la deseada “Web Semántica”. Para que esta tarea sea efectiva, se necesita que el conocimiento esté representado de forma que sea legible por los ordenadores, esté consensuado y sea reutilizable, es decir, para que las máquinas puedan llevar a cabo esta función necesitan acceder a colecciones estructuradas de información, y a formalismos actualmente basados en lógica matemática que les permitan tener un cierto grado de razonamiento

automático; las ontologías serán el elemento clave que cubrirá estas necesidades [Lozano, 2001].

Algunas de las tecnologías utilizadas en la Web Semántica son, por ejemplo, RDF (*Resource Description Framework*) [W3C, 2004g] que proporciona un modelo de datos común (basado en XML NameSpaces), el cual se utiliza para formalizar los metadatos, o OWL (*Ontology Web Language*), [W3C, 2004h] otro lenguaje de marcado utilizado para publicar y compartir datos usando ontologías en la Web. OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML, teniendo como antecedentes a DAML+OIL [W3C, 2001e], en los cuales se inspiraron los creadores de OWL para diseñar el lenguaje.

El término *ontología* (utilizado en filosofía para hablar acerca de una 'teoría sobre la existencia') ha sido adoptado por la comunidad de investigadores de Inteligencia Artificial para definir una categorización de un dominio de conocimiento, a través de conceptos y relaciones entre ellos, de forma compartida y consensuada; siendo esta conceptualización representada de una manera formal, legible y utilizable por los ordenadores [Gruber, 1993].

La definición de ontologías relacionadas con estrategias de enseñanza-aprendizaje, es de utilidad porque permite especificar dentro del objeto de aprendizaje información relevante para el procesamiento de dicho objeto desde el punto de vista pedagógico. Otra clase de ontologías que se necesitan, son las relacionadas con la estructura física del objeto de aprendizaje, para que éste pueda ser utilizado e interpretado en diferentes sistemas de enseñanza, como indican las especificaciones IMS o SCORM, estableciendo un mecanismo de empaquetado de objetos de aprendizaje basado en XML.

- **Servicios Web semánticos**

Otra tecnología implicada en la Web Semántica es la conocida como "Servicios Web Semánticos". Como ya se ha dicho en esta tesis, los servicios Web constituyen un mecanismo de reutilización de componentes software, distribuidos en diferentes servidores Web, que realizan una funcionalidad o actividad de negocio, que se ofrece a las aplicaciones que los invocan a través de una interfaz bien definida. Como extensión de los servicios Web, los servicios Web semánticos son un nuevo campo dentro de la Web semántica, porque se basa no sólo en describir la información de acceso a un servicio, sino en definir vocabularios de funcionalidad y procedimientos para describirlos. Tal descripción abarca aspectos como entradas, salidas, procesos, condiciones necesarias para que se puedan ejecutar, los efectos que producen y la información para localizarlos. Se habla de servicios Web semánticos porque se agrega semántica explicita a la descripción de los mismos, por medio de la adición de

metadatos, utilizando para ello ontologías. Aunque la especificación actual de servicios Web contiene metadatos en su descripción, a través del fichero WSDL asociado a cualquier servicio que se publique en la Web, éstos no son considerados como semánticos puesto que no están relacionados con ontologías.

La siguiente expresión, representa esquemáticamente esta evolución de los servicios Web: Servicios Web (Web de aplicaciones) + Web Semántica (Web de datos) = Servicios Web Semánticos (Web de datos y aplicaciones).

En la actualidad existen diferentes tecnologías para el desarrollo de servicios Web tanto semánticos como no semánticos. Una de las mejores alternativas, en el ámbito de los no semánticos, es el Lenguaje para la Ejecución de Procesos de Negocio (BPEL: *Bussines Process Execution Lenguage*) [BPEL, 2006] ya comentado en el capítulo dos. BPEL no incluye una relación estricta con ontologías y por tanto la composición no puede ser considerada como un servicio Web semántico.

En cuanto a las tecnologías para la composición de servicios Web semánticos podemos destacar por ejemplo, el Lenguaje para Ontologías de Servicios Web (OWL-S *Ontology Web Language - Services*) [W3C, 2004i] y la Ontología para Modelar Servicios Web (WSMO: *Web Services Modeling Ontology*) [WSMO, 2005].

- OWL-S: Se trata de una ontología basada en OWL, que permite agregar semántica a la descripción de los servicios, y realizar composiciones a partir de un conjunto de constructores de control, permitiendo describir las interacciones entre servicios, definiendo condiciones, ciclos repetitivos, secuencias y otros comportamientos posibles. OWL-S define tres partes funcionales que son: Perfil del Servicio, Modelo del Servicio y Acceso al Servicio.

- WSMO: El objetivo de esta iniciativa es resolver el problema de la integración, mediante una tecnología coherente que habilite el uso e implementación de los servicios Web semánticos. WSMO únicamente define el modelo conceptual para describir este nuevo tipo de servicios Web; sin embargo, se apoya en otras dos iniciativas para su representación sintáctica y su ejecución. Por una parte el Lenguaje para Modelar Servicios Web (WSML: *Web Services Modeling Language*), encargado de la descripción sintáctica de los servicios siguiendo el modelo establecido por WSMO. Por otro lado, el Entorno de Ejecución para el Modelado de Servicios Web (WSMX: *Web Services Modeling Execution Enviroment*), responsable de soportar la ejecución de los servicios Web semánticos descritos con WSML y basados en el paradigma definido por WSMO.

- **Repositorios semánticos de objetos de aprendizaje**

En los últimos años también se están incorporando mecanismos de búsqueda semánticos en los repositorios de objetos de aprendizaje, como ya han propuesto otros autores a través de la utilización de ontologías en dichos repositorios, para facilitar así la organización y búsqueda de estos recursos didácticos; un ejemplo de ello es el proyecto SLOR (*Semantic Learning Object Repository*) [Soto et al., 2006]. Existen también propuestas para la incorporación de técnicas de Ingeniería Lingüística en combinación con las ontologías, como la de Hilera et al. [2005] que sugiere:

  - Utilizar estas técnicas para crear la ontología del repositorio a partir de información textual, especialmente para extraer los conceptos relacionados con los dominios de conocimiento con los que están relacionados los objetos de aprendizaje almacenados.

  - Combinar ambas tecnologías en los procesos de búsqueda de objetos, utilizando el conocimiento del dominio representado en la ontología para limitar los resultados de la búsqueda, aplicando la Ingeniería Lingüística para extraer información de un texto que coincida con esa ontología, es decir, interpretando sólo expresiones que tengan sentido dentro del marco interpretativo de la ontología.

- **Registros UDDI semánticos**

Otro de los aspectos a tener en cuenta es UDDI, protocolo que permite describir los componentes disponibles de servicios Web. Este estándar permite a las empresas registrarse, y les ayuda a anunciar sus servicios, como se comentó en el capítulo dos.

La versión más reciente de UDDI, la versión 3.0.2, ya es un estándar de OASIS desde febrero de 2005 [UDDI, 2006], aunque fue publicada a finales de junio de 2002. Entre las principales características de esta versión se pueden destacar las mejoras producidas en cuanto a las políticas de seguridad, gracias, entre otros factores, a la incorporación de la firma digital, o el soporte mejorado de documentos WSDL. Sin embargo, a pesar de las nuevas características de esta especificación 3.0, se han encontrado algunas limitaciones al modelo, entre las que destacan las siguientes [Iribarne, 2003]:

  - Protocolos. WSDL utiliza el término protocolo para referirse al protocolo de transporte (HTTP, MIME, FTP, SMTP). Sin embargo, el esquema XML de WSDL no contempla una notación para referirse al concepto de protocolo (o coreografía), en el sentido de conocer el orden en el que se llaman a las operaciones de entrada y de salida.

- Comportamiento. Es necesario que el lenguaje permita definir también el comportamiento semántico de las operaciones (pre/post condiciones e invariantes), y no solo la definición sintáctica de las signaturas usando etiquetas XML.

- Información extra-funcional. Por último, también es necesario que el lenguaje pueda utilizar una notación para definir propiedades extra-funcionales, además de la información funcional (sintaxis, semántica y protocolos de las interfaces) y la información técnica del servicio.

Resumiendo, todas estas limitaciones del lenguaje WSDL deberían ser incorporadas o bien directamente en XML, o bien indirectamente a través de un puntero externo. De alguna forma también repercuten como limitaciones en el modelo UDDI, y básicamente afectan a las operaciones de búsqueda en el repositorio. En un registro UDDI, las operaciones de búsqueda, por lo tanto, están sujetas sólo a los aspectos técnicos de una empresa proveedora de servicios, a los aspectos técnicos de los propios servicios, y también a aspectos de localización, conexión y comunicación de las operaciones de los mismos. Sin embargo, estas operaciones no permiten búsquedas sobre los protocolos de interacción de las operaciones, el comportamiento de las operaciones, o simplemente la tipología de servicio Web, desde un punto de vista semántico y no sintáctico.

La solución sería proveer a UDDI del motor de búsqueda adecuado para generar una capa semántica sobre los registros UDDI. De este modo se puede buscar por palabras clave o por las características (semánticas) del servicio. Al registrar un servicio sobre UDDI se debería enviar además su descripción semántica, de modo que se guarda tanto su descripción UDDI, como su descripción declarativa; esto le da un gran valor agregado a la búsqueda de servicios. Es decir, las operaciones de un servicio Web serán descritas de acuerdo a una ontología, utilizándose para ello WSMO, implementado a través de WSML. La ontología describe operaciones de servicios Web, por lo que cada nueva operación será una nueva instancia de la ontología, la que será almacenada en una base de datos que implementará el modelo relacional elegido.

Ya existen en la actualidad algunos proyectos que intentan modificar los mecanismos de búsquedas de servicios dentro de los registros UDDI, adaptándolos a metodologías semánticas, como por ejemplo el proyecto Sem B-UDDI de la Universidad de Deusto [Sem, 2005].

- **Integración de tecnologías semánticas en la arquitectura**

Teniendo en cuenta todo lo anterior, y con objeto de plantear una nueva línea de investigación a partir de esta tesis, a continuación se analizan algunas limitaciones de la arquitectura propuesta en la tesis, que podrían mejorarse con técnicas semánticas:

- A bajo nivel: el sistema se encarga de realizar peticiones de objetos de aprendizaje utilizando los mecanismos de búsqueda propios de cada repositorio en cuestión, para que de esta manera la arquitectura no tenga que establecer ninguna restricción sobre el repositorio, y así ser totalmente compatible con todos. Normalmente estas búsquedas se hacen por coincidencias sintácticas, por lo que se introduce una gran cantidad de "ruido" durante estos procesos. Esa cantidad de "ruido", variará en función del conjunto de campos educativos que se utilicen, ya que para alguno de ellos los valores ya están prefijados, como por ejemplo tipos de recursos (sonidos, imágenes, archivos doc o pdf, etc.), o idioma en el que está desarrollado el contenido educativo. Sin embargo habrá otros tipos diferentes de campos, en los que el usuario podrá rellenar la información sin limitaciones, como descripciones o resúmenes, en los que será de gran utilidad el dotar de herramientas semánticas a este tipo de búsquedas.

- A más alto nivel: la arquitectura está pensada para que sean servicios Web iguales o muy similares, los que estén asociados a todos los repositorios, de tal forma que no se establecen distinciones en cuanto a tipos diferentes de servicios Web para una obtención directa de una tipología concreta de objetos de aprendizaje. Es decir, podría considerarse la posibilidad de indicarle al registro UDDI que busque sólo servicios Web de obtención de recursos de aprendizaje de una categoría determinada, por ejemplo, "cursos de programación en Java". De esta manera, el sistema no contactará con aquellos repositorios que no tengan asociados servicios Web de obtención de recursos didácticos relacionados con esta categoría, por supuesto, siempre hablando desde un punto de vista semántico, no sintáctico.

En el futuro inmediato pretendemos establecer dos cambios importantes en la arquitectura, por un lado dotar de mecanismos semánticos a la búsqueda de objetos de aprendizaje presentes en los repositorios, y por otro lado, incorporar estos mismos mecanismos para las búsquedas de servicios en los diferentes registros UDDI.

La solución a la primera de las opciones es compleja, puesto que el sistema como ya se ha indicado, utiliza los mecanismos de búsqueda internos del repositorio, lo cual quiere decir que no siempre podremos contar con búsquedas semánticas a este nivel, ya que podrían no estar implementadas en el repositorio. Otra opción sería implementar

estos mecanismos directamente en el servicio asociado al repositorio, pero esto supondría tener un acceso completo al mismo, ya que tendríamos que conocer detalles como la ubicación de los objetos dentro de éste (o referencias hacia el lugar en el que se encuentran ubicados), indexación en el caso que se realice, ordenación de objetos, etc. y estos detalles es posible que no siempre se conozcan.

En cuanto a la segunda de las opciones, su resolución es más sencilla, ya que se debería realizar una división de servicios Web por categorías de material docente, o por las simples características del mismo; de tal forma que si un usuario, por ejemplo desea realizar búsquedas para un tipo determinado de material educativo, se evitará buscar en repositorios que no tengan este tipo de material; o, por ejemplo, para buscar solamente en aquellos repositorios que dispongan de objetos adaptables a dispositivos de capacidad reducida, o con ciertas restricciones. Evidentemente para estas tareas recurriremos a ontologías.

6.3.3 Integración de agentes software en la arquitectura

La tecnología de agentes es un área de desarrollo relativamente joven dentro de las Tecnologías de la Información que está sufriendo un rápido crecimiento. No es una tecnología totalmente nueva, sino más bien la aplicación integrada de múltiples tecnologías, tales como la Inteligencia Artificial, los Sistemas Distribuidos Orientados a Objetos, y la interacción persona-ordenador.

Para poder realizar un análisis adecuado del estado de la tecnología de agentes, es necesario partir de conceptos como agente o sistema multiagente, lo que hace conveniente definirlos adecuadamente [Brenner et al., 1998]. Como suele suceder en las disciplinas aún en estado emergente, no existe una definición universalmente aceptada del concepto de agente software. El diccionario nos proporciona como definición de agente “cualquier entidad que actúa” [RAE, 2006]. Evidentemente, esa definición es demasiado genérica para apoyar en ella el desarrollo de sistemas software.

Algunas de las diversas definiciones del concepto de agente que encontramos en la bibliografía especializada se ven influidas por la aplicación específica sobre la que se realiza el desarrollo, si bien existen ciertos grupos de investigación que son partidarios de definiciones más generales y universales [FIPA, 2006]. Desde el punto de vista de las implicaciones tecnológicas y de diseño, podríamos definir un agente software como un programa autocontenido capaz de controlar su propia toma de decisiones y de actuar, basándose en la percepción de su entorno, para la consecución de uno o más objetivos [Franklin y Graesser, 1996]. Atendiendo más a la perspectiva funcional del usuario, puede definirse como una entidad software en la que se pueden delegar tareas. Esta última definición, aunque más simple, ilustra con claridad el propósito último de los

agentes software: la automatización de determinadas tareas comúnmente llevadas a cabo por los usuarios.

- **Características de los agentes**

A continuación se detallan algunas de las características básicas que pueden presentar los diferentes agentes de un sistema [Milojicic et al., 1998] [Lange y Oshima, 1998]:

- Autonomía:

La autonomía es la capacidad de actuar sin la intervención directa de una persona o de otro agente. Hasta cierto punto, los agentes pueden funcionar sin una intervención externa directa. Un agente debe poder controlar sus propias acciones y estado interno. Una vez que el usuario activa el agente indicando algún objetivo de alto nivel, éste actúa independientemente, seleccionando estrategias y monitorizando el progreso en busca de la meta. Si falla con una estrategia, usará otra, pero sin intervención humana o con la mínima indispensable.

- Reactividad

Se refiere al hecho de que un agente debe poder sentir el estado del ambiente dentro del cual se encuentro inmerso y, en función de esto, actuar, respondiendo de manera adecuada a cambios producidos en el mismo. Los efectos producidos pueden modificar el estado de su entorno.

- Interactividad

La interactividad se define como la capacidad de comunicarse tanto con el entorno como con otras entidades o agentes. La forma más básica de comunicación en sistemas software es el paso de mensajes entre agentes (también llamada invocación de métodos). Otra forma de interacción más compleja es la percepción de cambios en el entorno del agente. Resulta evidente que para que un agente resulte útil debe ser capaz de interaccionar de algún modo con su entorno o con otras entidades, por lo que la interactividad es una propiedad esencial de un agente software.

- Adaptabilidad

La adaptabilidad es la capacidad de un agente de responder a otros agentes o a su entorno en cierta medida. Un nivel mínimo de esta propiedad lo encontramos en agentes que son capaces de reaccionar ante estímulos simples, proporcionando una respuesta determinista ante los mismos. Un nivel más alto de adaptabilidad supone que el agente

sea capaz de razonar, es decir, de realizar inferencias a partir de los datos recibidos. Las formas más avanzadas de adaptabilidad implican agentes que sean capaces de modificar su comportamiento en base a la experiencia adquirida, y de este modo aprender y evolucionar.

Al igual que sucedía con la interactividad, un agente que no sea capaz de responder ante su entorno o ante otras entidades carece de utilidad, por lo que la adaptabilidad es esencial para el concepto de agente. Un ejemplo de esta característica sería un agente de reconocimiento del habla que reconociera los patrones que posee en diferentes entornos, ruidos y personas.

- Movilidad:

La movilidad es la capacidad de un agente para transportarse a sí mismo desde un entorno de ejecución a otro. La ventaja principal de la movilidad para ciertas aplicaciones es que permite al agente trasladarse directamente a la máquina donde se encuentran los servicios que necesita, con lo que se incrementa el rendimiento del intercambio de datos entre el agente y las entidades que le proporcionan los servicios. Las principales desventajas de la movilidad de agentes son las consideraciones de seguridad que genera y la posibilidad de cargar excesivamente la máquina que ofrece un determinado servicio con la ejecución del código de los agentes cliente, con los problemas de escalabilidad que eso puede ocasionar.

La movilidad no es una propiedad indispensable para un agente, sino que modifica la forma por la cual el agente cumple con sus objetivos, en este caso recurriendo a los recursos que puede ofrecer una red de computadoras. Aporta una nueva forma de computación distribuida.

- Inteligencia

No existe un acuerdo claro en lo que se refiere a la definición del término inteligencia aplicada a los agentes software. Una aproximación prudente es entender como inteligencia de los agentes software cierto grado de proactividad, es decir, que los agentes muestren cierto nivel de conocimiento y sean capaces de dar unas respuestas correctas en base a ese conocimiento [Plekhanova, 2002].

La dimensión de inteligencia representa el grado en el cual el agente utiliza el razonamiento, aprendizaje y otras técnicas para interpretar la información o conocimiento al cual tiene acceso.

Esta dimensión tiene diferentes aproximaciones desde una inteligencia marginal a otra muy avanzada. Las formas más modestas de inteligencia permiten al usuario

expresar sus preferencias. En un nivel superior se tiene una formalización de un conjunto de reglas de razonamiento que combinadas con conocimiento a corto y largo plazo, en un proceso de inferencia, puede conducir a la forma de alguna acción. El tercer y último paso es la capacidad del agente de modificar su capacidad de razonamiento en la base de un nuevo conocimiento derivado de un amplio rango de fuentes, esto es, aprender.

- Proactividad

Un agente busca satisfacer cierto estado interno o lograr un objetivo determinado, con mínima intervención humana. Por ejemplo, un agente recuperador de datos tiene especificada una tarea. El agente deberá intentar permanentemente satisfacer la tarea que le fue delegada en base a las estrategias de búsqueda y recuperación con las cuales fue construido, hasta cumplir con el objetivo. Mientras tanto, el agente debe mostrar un comportamiento activo, que lo lleve a tomar la iniciativa de la acción a tomar cuando lo considere apropiado.

- Sociabilidad

Un agente debe ser comunicativo y “sociable”. Debe tener habilidad para interactuar con otros agentes o incluso con alguna persona o usuario, para solicitar información o bien para exponer los resultados obtenidos de la ejecución de las tareas aprendidas. La naturaleza de la comunicación dependerá del tipo de agente con quién se comunique, estableciéndose un protocolo común de intercambio de información entre ambas partes. Los agentes deben poseer algún tipo de interfaz para comunicarse con sus usuarios.

- Continuidad temporal

Un agente es un proceso temporalmente continuo. A diferencia de un programa convencional del cual se conoce su inicio y fin, un agente debe ejecutarse hasta que se haya alcanzado con el conjunto de objetivos solicitados, o bien, mientras su ciclo perdure y su usuario no desee detenerlo. La continuidad temporal es la propiedad que da “vida” al agente, posibilitando que se mantenga alerta a una solicitud o a algún cambio en el medio. El ciclo de vida de un agente depende de sus características, de las tareas que realice y de los deseos de su usuario en cuanto al tiempo durante el cual el agente debe ejecutarse.

- **Sistemas multiagente**

Existen aplicaciones basadas en agentes en las que no se produce prácticamente ninguna interacción entre diferentes agentes (agentes software aislados). Por ejemplo, los servicios de búsqueda a través de Internet se basan en el lanzamiento de múltiples agentes autónomos que recorren las páginas Web y seleccionan las secciones más relevantes de la información que encuentran, la clasifican y la almacenan en una base de datos que posteriormente sirve como índice para devolver al usuario la información cuando realiza una consulta. Sin embargo, se pueden crear sociedades de agentes que se coordinen para llevar a cabo diferentes tareas. Estas sociedades son lo que se conoce como sistemas multiagente [Wooldridge, 2002].

Un sistema multiagente es, pues, un sistema compuesto por agentes que se coordinan a través de las relaciones entre ellos. Estas relaciones pueden ser de cooperación, de competencia, o ser una mezcla de ambas. Algunas de las ventajas de los sistemas multiagente son las siguientes:

- ✓ Permiten dividir la funcionalidad necesaria para desempeñar tareas complejas entre distintos tipos de agentes, lo que proporciona una mayor modularidad, flexibilidad y extensibilidad, además de presentar un mejor rendimiento que la opción de un único agente que desempeñe por sí solo la tarea completa.

- ✓ Permiten repartir también la información manteniéndola accesible, eliminando la necesidad de repositorios de datos centralizados. El agente que necesite hacer uso de la información que posee otro agente sólo necesita pedírsela.

- ✓ Facilitan el desarrollo de aplicaciones de computación distribuida.

Los sistemas multiagente requieren la existencia de un entorno adecuado para el funcionamiento de las entidades que lo forman. En general, dicho entorno constituye lo que se denomina una plataforma multiagente, como por ejemplo JADE [2006], plataforma multiagente desarrollada en Java, que proporciona la infraestructura necesaria para posibilitar la ejecución simultánea controlada de varios agentes y la comunicación fiable y organizada entre los mismos.

- **Integración de agentes en la arquitectura**

Después de ver una introducción de los conceptos más relevantes sobre la tecnología de agentes pasaremos a indicar cómo se pueden integrar en la arquitectura propuesta en la tesis, así como en los sistemas basados en ella, para añadir ciertas mejoras.

La búsqueda federada es la funcionalidad más destacable de la arquitectura, pero como hemos comentado esa búsqueda se produce en un momento dado y con unos parámetros de búsqueda y sobre una serie de repositorios. Sin embargo, podemos tener una necesidad en el tiempo de conseguir contenidos docentes de unas determinadas características. De esta forma, el usuario puede indicarle al sistema que conserve sus parámetros de búsqueda y un histórico con los resultados obtenidos, de forma que cuando se encuentren contenidos nuevos que coincidan con los parámetros de búsqueda los recupere y avise al usuario; por ejemplo, mediante un correo electrónico. De esta forma, si se añaden nuevos repositorios al sistema o se añaden nuevos contenidos que interesen al usuario, éste recibirá un aviso indicándoselo.

Para realizar esta tarea es muy adecuado el uso de agentes software, ya que como se ha visto en su descripción tendrían un objetivo a cumplir, que sería el de cubrir las necesidades de localización de contenidos docentes de unas determinadas características. Podemos decir que el agente utilizaría, de forma preactiva, los datos de búsqueda para automatizar y optimizar la localización de contenidos docentes requeridos por el usuario, de un modo más rápido y eficiente que si el usuario realizara esta labor de forma continuada.

Habría dos formas de desarrollar esta nueva funcionalidad:

1. Realizar agentes aislados en el sistema, encargados de tratar los distintos tipos de búsqueda de manera que consultara de forma periódica cada uno de los repositorios y mantuviera un histórico de los resultados.

2. Realizar un sistema multiagente, de forma que en cada uno de los repositorios hubiera una serie de agentes encargados de comunicar a los agentes del sistema posibles cambios en los contenidos.

Sobre esta funcionalidad estamos trabajando actualmente para dar un nuevo valor añadido a la arquitectura y al prototipo desarrollado.

7 BIBLIOGRAFÍA Y REFERENCIAS

[ActiveBPEL, 2006]

- ActiveBPEL: The Open Source BPEL Engine. http://www.activebpel.org/

[ADL, 2004]

- ADL: Sharable Content Object Reference Model (SCORM) overview. 2004. http://www.adlnet.gov/scorm/index.cfm

[ADL, 2006]

- Advanced Distributed Learning Network. http://www.adlnet.gov/index.cfm

[ADL-R, 2006]

- ADL Registry. https://adlregistry.dtic.mil/

[AICC, 2006]

- AICC - Aviation Industry CBT Committee. http://www.aicc.org/

[Alexander et al., 1977]

- Alexander, C., Ishikawa, S., Silverstein, M., "A Pattern Language: Towns, Buildings, Construction". Oxford University Press. 1977.

[Allamaraju, 1998]

- Allamaraju, S., "Architecture Paradox". 1998. http://www.subrahmanyam.com/articles/architecture/Paradox.html.

[Anido et al., 2002]

- Anido, L. E., Fernández, M. J., Caeiro, M., Santos, J. M., Rodríguez, J. S., Llamas, M., "Educational metadata and brokerage for learning resources" Computers & Education 38: 351-374. 2002.

[Apache, 2006]

- Apache Tomcat. http://tomcat.apache.org/

[ARIADNE, 2006]

- ARIADNE (Alliance of Remote Instructional Authoring and Distribution Network for Europe). http://www.ariadne-eu.org/

[ATA, 2003]

- Architecture Tradeoff Analysis (ATA). 2003. http://www.sei.cmu.edu/ata/ata_init2.html

[Barbacci et al., 1997]

- Barbacci, M. R., Carriere, S. J., Feiler, P. H., Kazman, R., Klein, M. H., Lipson, H. F., Longstaff, T. A., Weinstock, C. B., "Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis". Technical Report. CMU/SEI-97-TR-029. Carnegie Mellon University, 1997. http://www.sei.cmu.edu/publications/documents/97.reports/97tr029/97tr029abstract.html

[Bass et al., 1997]

- Bass, L., Clements, P., Kazman, R., "Software Architecture in Practice". Boston: Addison-Wesley. 1997.

[Batman, 1999]

- Batman, J., "Characteristics of an Organization with Mature Architecture Practices". Essays on Software Architecture of Software Engineering Institute. Carnegie Mellon University. 1999. http//www.sei.cmu.edu/architecture/essays.html.

[Berners-Lee et al., 2001]

- Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web". Scientific American, Vol. 284, No. 5, pp. 34-43. 2001.

[Boehm y Port, 1998]

- Boehm, B., Port, D., "Conceptual Modeling Challenges for Model-Based Architecting and Software Engineering". 1998

[Booch et al., 1999]

- Booch, G., Rumbaugh J., Jacobson, I., "The UML Modeling Language User Guide". Addison-Wesley. 1999.

[BPEL, 2006]

- Business Process Execution Language (BPEL). http://en.wikipedia.org/wiki/BPEL

[Brenner et al., 1998]

- Brenner, W., Zarnekow, R., Wittig, H., "Intelligent Software Agents: Foundations and Applications". Springer-Verlag. 1998.

[Brown et al., 1989]

- Brown, J. S., Collins, A., Duguid, P., "Situated Cognition and the Culture of Learning. Educational Researcher". 1989.

[BSI, 2005]

- BSI (British Standard Institute). http://www.bsi-spain.com

[Buschmann et al., 1996]

- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., "Pattern-Oriented Software Architecture". 1996

[CAREO, 2006]

- CAREO (Campus Alberta Repository of Educational Objects), http://careo.ucalgary.ca/cgi-bin/WebObjects/ CAREO.woa/wa/Home?theme=careo

[CBDI, 2006a]

- CBDI Service Oriented Architecture Practice Portal. "Web Services Protocols Summary". 2006. http://roadmap.cbdiforum.com/reports/protocols/summary.php

[CBDI, 2006b]

- CBDI Forum. http://www.cbdiforum.com

[CCE, 2000]

- COMISION DE LAS COMUNIDADES EUROPEAS. "e-Learning. Concebir la educación del futuro". Comunicación de la Comisión. Bruselas, 25 mayo 2000. Referéncia: Bruselas 25.5.2000 COM(2000) 318 Final. http://europa.eu.int/eur-lex/es/com/cnc/2001/com2001_ 0172es01.pdf.

[Cisco, 2000]

- CISCO SYSTEMS. "Network Architectures for E-learning Applications". Cisco Systems, diciembre 2000. http://www.digitalpipe.com/pdf/dp/white_papers/e_learning/cisco_network_arch.doc

[Cisco, 2001]

- CISCO SYSTEMS. "Model of an E-Learning Solution Architecture for the Enterprise". Cisco Systems, abril 2001. http://www.cisco.com/warp/public/10/wwtraining/elearning/learn/whitepaper_docs/solution_architecture_wp.pdf.

[COM, 2006]

- Microsoft COM (Component Object Model) technology. http://www.microsoft.com/com/default.mspx

[CORBA, 2006]

- OMG CORBA (Common Object Request Broker Architecture). http://www.corba.org/

[CORDRA, 2006]

- CORDRA (Content Object Repository Discovery and Registration/Resolution Architecture), http://cordra.lsal.cmu.edu/cordra/

[DC, 2006]

- Dublin Core Metadata Initiative. http://dublincore.org/

[Deitel et al., 2002]

- Deitel, H. M., Deitel, P. J., DuWaldt, B., Trees, L. K., "Web Services: A Technical Introduction". 2002.

[Dierks y Allen, 1999]

- Dierks, T., Allen, C., "The TLS Protocol Version 1.0". 1999. http://tools.ietf.org/html/2246

[Dodero, 2002]

- Dodero, J. M., "Una arquitectura multiagente para la producción distribuida de conocimiento y su aplicación al desarrollo compartido de objetos educativos.". Departamento de informática. Universidad Carlos Tercero, Madrid. 2002.

[Downes, 2002]

- Downes, S., "Design and Reusability of Learning Objects in an AcademicContext: A New Economy of Education?" National Research Council, Moncton, Canada. 2002. http://www.downes.ca/files/milan.doc

[EAI, 2006]

- Enterprise Application Integration (EAI). http://en.wikipedia.org/wiki/Enterprise_application_integration

[Eclipse, 2006]

- Eclipse Foundation. http://www.eclipse.org/

[EDI, 2006]

- Electronic Data Interchange (EDI). http://www.monografias.com/trabajos/edi/edi.shtml

[EDUCASE, 2006]

- EDUCASE. http://www.educause.edu

[EduTools, 2006]

- EDUTOOLS. "Product Information" [análisis de productos online]. http://www.edutools.info/static.jsp?pj=8&page=HOME

[ESB, 2006]

- Enterprise Service Bus (ESB). http://en.wikipedia.org/wiki/Enterprise_Service_Bus

[FIPA, 2006]

- Foundation for Intelligent Physical Agents. http://www.fipa.org

[Franklin y Graesser, 1996]

- Franklin, S., Graesser, A. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents". Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag. 1996.

[Freier et al., 1996]

- Freier, A., Karlton, P., Kocher, P., "The SSL Protocol Version 3.0". 1996. http://wp.netscape.com/eng/ssl3/draft302.txt

[Friesen, 2004]

- Friesen, N., "International LOM Survey: Report. Information Technology for Learning, Education, and Training". 2004. http://mdlet.jtc1sc36.org/doc/SC36_WG4_N0109.pdf

[Garlan y Perry, 1995]

- Garlan, D., Perry, D., "Special Issue on Software Architecture" (Editor's Introduction). En: IEEE Transactions on Software Engineering. 1995, vol.21, num.4.

[GLOBE, 2006]

- GLOBE (Global Learning Objects Brokered Exchange), http://globe.edna.edu.au/globe/go/pid/2

[GoF, 2003]

- Gamma, E., Helm, R., Jonson, R., Vlissides, J., "Patrones de Diseño". Addison Wesley. 2003.

[Gram et al., 1998]

- Gram, M., Mark, T., McGreal, R., "A survey of new media development and delivey software for internet based learning". Industry Canada. Science Promotion and Academic Affairs Branch. Canada. 1998.

[Gruber, 1993]

- Gruber, T. R., "A Translation Approach to Portable Ontology Specifications". Knowledge Acquisition, No. 5, pp. 199-200. 1993

[Gutiérrez y Otón, 2004]

- Gutiérrez, J. M., Otón, S., "Necesidad de estándares en una arquitectura distribuida para sistemas de teleformación" En: Actas 3ra. Conferencia Iberoamericana en Sistemas, Cibernética e Informática. 2004.

[Hatala et al., 2004]

- Hatala, M., Richards, G., Eap, T., Willms, J., "The Interoperability of Learning Object Repositories and Services: Standards, Implementations and Lessons Learned". 2004.

[Henderson, 2003]

- Henderson, A. J., "The E-Learning question and answer book, a survival guide for trainers and business managers". 2003.

[Hilera et al., 2005]

- Hilera, J. R., Bengochea, L., Sánchez, R. Gutiérrez, J. A., Martínez, J. J., "Aplicación de técnicas de Ingeniería Lingüística en sistemas de e-learning basados en objetos de aprendizaje". CEDI 2005. http://cedi2005.ugr.es/

[Hiltz y Norwood, 1994]

- Hiltz, S. R., Norwood, N. J., "The Virtual Classroom: Learning without Limits via Computer Networks". Ablex. 1994.

[Hodgins, 2000]

- Hodgins, H. W., "Into the Future. A vision paper". Learnativity(ed). 2000. http:/www.learnativity.com/download/MP7.PDF.

[Hodgins, 2001]

- Hodgins, H. W., "IEEE LTSC Learning Technology Standards Committee P1484". ADLNET, USA. 2001.

[Horstmann y Cornell, 2003]

- Horstmann, C. S., Cornell, G., "Java 2 Características Avanzadas". Prentice Hall. 2003.

[Horton, 2001]

- Horton, W., "Leading e_Learning". American Society for Training & Development". 2001.

[HP, 2006]

- HP Web Services. http://www.hpmiddleware.com/webservices

[IBM, 2004]

- Web Services Notification (WS-Notification). 2004. http://ifr.sap.com/ws-notification/ws-notification.pdf

[IBM, 2005]

- BPEL4WS (Business Process Execution Language for Web Services). 2005. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[IBM, 2006]

- IBM Web Services. http://www.ibm.com/developerworks/webservices

[IEEE, 1990]

- IEEE Std 610.12-1990. "IEEE standard glossary of software engineering terminology". New York. 1990.

[IEEE, 2000]

- IEEE Std 1471-2000. "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems". New York. 2000.

[IEEE, 2001]

- Learning Technology Standards Committee -LTSC-. IEEE Computer Society, "IEEE P1484.1/D9, 2001-11-30 Draft Standard for Learning Technology—Learning Technology Systems Architecture (LTSA)". 2001. http://edutool.com/ltsa/

[IEEE, 2002]

- IEEE 1484.12.1-2002. "Draft Standard for Learning Object Metadata". 2002. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

[IEEE, 2006]

- Institute of Electrical and Electronics Engineers. http://www.ieee.org/portal/site

[IMS, 2003a]

- IMS Global Learning Consortium. "IMS Digital Repositories Interoperability". 2003. http://www.imsglobal.org/digitalrepositories/index.html

[IMS, 2003b]

- IMS Global Learning Consortium. "IMS Abstract Framework". 2003. http://www.imsglobal.org/af/index.html

[IMS, 2004]

- IMS Global Learning Consortium. "IMS Enterprise Services". 2004. http://www.imsglobal.org/es/index.html

[IMS, 2005]

- IMS Global Learning Consortium. "IMS General Web Services". 2005. http://www.imsglobal.org/gws/index.html

[IMS, 2006]

- IMS Global Learning Consortium. http://www.imsproject.org/

[IPSec, 2006]

- IPSec Internet Security Protocol. http://www.ietf.org/iesg/1rfc_index.txt

[Iribarne, 2003]

- Iribarne, L. F., "Un modelo de mediación para el desarrollo de software basado en componentes COTS". 2003. http://www.ual.es/~liribarn/tesis.html

[ISO, 1986]

- SGML. Standard Generalized Markup Language, ISO 8879. 1986. http://www.w3.org/MarkUp/SGML/

[ISO, 1992]

- Hipermedia/Timed-based structuring Language (HyTime). ISO/IEC 10744. 1992. http://www.hytime.org/

[ISO, 1996]

- DSSSL. Document Style Semantics and Specification Language. ISO/IEC 10179. 1996. http://www.oasis-open.org/cover/dsssl.html

[ISO, 2000]

- HTML. HyperText Markup Language, ISO 15445. 2000. http://www.w3.org/MarkUp/

[ISO, 2005]

- ISO (Organización Internacional para la Estandarización). http://www.iso.org

[ISO-SC36, 2006]

- ISO/IEC JTC1/SC36. Information Technology for Learning, Education, and Training. 2006. http://jtc1sc36.org/

[Jackson, 2001]

- Jackson, R. H., "Web Based Learning Resources Library". 2001.

[Jacobson et al., 1992]

- Jacobson, I., Christerson, M., Jonsson P., Overgaard, G., "Object Oriented Software Engineering: A Use Case Driven Aproach". Addison Wesley. 1992.

[Jacobson et al., 2000]

- Jacobson, I., Booch, G., Rumbaugh, J., "El Proceso Unificado de Desarrollo de Software". Addison Wesley. 2000.

[JADE, 2006]

- JADE (Java Agent DEvelopment Framework). http://jade.tilab.com/

[Jazayeri et al., 2000]

- Jazayeri, M., Ran, A., Linden., F., "Software Architecture for Product Families: Principles and Practice". Addison Wesley. 2000.

[Koper, 2000]

- Koper, R., "From change to renewal: Educational technology foundations of electronic learning environments." Open University of Netherlands. 2000.

[Lanfranco, 1993]

- Lanfranco, S., Utsumi, T., "Objects, Agents and Events in a Global Learning Environment. Proceedings of Teleteaching'93". Trondheim (Noruega). 1993.

[Lange y Oshima, 1998]

- Lange, D. B., Oshima, M. "Programming and Deploying java mobile agents with aglets". Addison-Wesley. 1998.

[Larman, 2001]

- Larman, C., "Applying UML and Patterns - An Introduction to object Analysis and Design ". 2nd Edition, Prentice Hall. 2001.

[Levy, 1993]

- Levy, P., "As tecnologias da inteligencia, o futuro do pensamento na era da informática". Editora 34. 1993.

[Liu et al., 2003]

- Liu, X., Saddik, A., Georganas, N. D., "An implementable architecture of an e-learning system". 2003.

[Lozano, 2001]

- Lozano, A., "Ontologías en la Web Semántica". I Jornadas Internacionales de Ingeniería Web. 2001.
  http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf

[Mahmoud, 2002]

- Mahmoud, Q. H., "Distributed Programming with JAVA". Manning Editorial. 2002**.**

[Martignago, 1998]

- Martignago, E., "Decentriamo l'insegnamento". En: Sesto potere. Guida per giornalisti, comunicatori aziendali, formatori nell'era di Internet, abril 1998.

[Martínez et al., 2001]

- Martínez, J. J., Sicilia, M. A., Garcia., E., "Extending IMS Course Structures for Conditional Learning Path Support". En: Actas del 3º Simposio Internacional de Informática Educativa. 2001.

[Masie, 2002]

- Masie, E., "Making Sense of Learning Specifications & Standards: A Decision Maker's Guide to their Adoption". The Masie Center, Saratoga Springs. http://www.masie.com. 2002.

[McGovern et al., 2003]

- McGovern, J., Tyagi, S., Stevens, M., Mathew, S. "Java Web Services Architecture". Morgan Kaufmann. 2003.

[Mercury, 2006]

- Mercury Corporate. http://www.mercury.com/

[MERLOT, 2006]

- MERLOT (Multimedia Educational Resource for Learning and Online Teaching). http://www.merlot.org/Home.po

[Microsoft, 1999]

- Microsoft e-learning. https://www.microsoftelearning.com/

[Microsoft, 2006]

- Microsoft .NET. http://www.microsoft.com/net/default.mspx

[Milojicic et al., 1998]

- Milojicic, D., Musliner, D., Schrueder, W., "Agents: Mobility and communication". Proceedings of the Thirty-first Annual Hawaii International Conference on System Sciences. VIII. 2-3. 1998.

[Molnar, 1990]

- Molnar, A. R., "Computers in Education: a Historical Perspective of the Unfinished Task". T.H.E. Journal 18(4): 80-83. 1990.

[Monday, 2003]

- Monday, P., "Web Service Patterns: Java Edition". Apress. 2003.

[Moreno y Bailly-Baillière, 2002]

- Moreno, F., Bailly-Baillière, M., "Diseño instructivo de la formación on-line. Aproximación metodológica a la elaboración de contenidos". Ariel Educación, Barcelona. 2002.

[Morrison, 2004]

- Morrison, D., "E-Learning Strategies How to get implementation and delivery right first time". Wiley Publishing. 2004.

[MS, 2006]

- Microsoft Web Services. http://msdn.microsoft.com/webservices

[MySQL, 2006]

- MySQL. http://www.mysql.com/

[Nasseh, 1996]

- Nasseh, B., "Artificial Intelligence and Internet". Ball State University, 1996. http://www.bsu.edu/classes/nasseh/

[Newcomer, 2002]

- Newcomer, E., "Understanding Web Services: XML, WSDL, SOAP, and UDDI". Addison-Wesley. 2002.

[OASIS, 2002]

- Oasis UDDI. http://www.uddi.org

[OASIS, 2005a]

- OASIS ebXML (Electronic Business using eXtensible Markup Language) http://www.ebxml.org

[OASIS, 2005b]

- OASIS ebSOA (Electronic Business Service Oriented Architecture) TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebsoa

[ObjectWeb, 2006]

- ObjectWeb Lomboz. http://www.objectlearn.com/index.jsp

[OED, 1989]

- Oxford English Dictionary (OED). Edited by John Simpson and Edmund Weiner. Clarendon Press. 1989.

[OMG, 2001]

- OMG ARCHITECTURE BOARD MAD DRAFTING TEAM. "Model Driven Architecture: A Technical Perspective". Architecture Board ORMSC1 (ed.). 2001. Document number ormsc/2001-07-01.

[Oracle, 2005]

- Oracle BPEL. http://www.oracle.com/technology/products/ias/bpel/index.html

[Pagés et al., 2003]

- Pagés, C., Sicilia, M. A., García, E., Martínez, J. J., Gutiérrez, J. M., "On the Evaluation of Completeness of Learning Object Metadata in Open Repositories". Proceedings of the Second International Conference on Multimedia and Information & Communication Technologies in Education (m-ICTE 2003), 1760-1764. 2003.

[Patron, 2003]

- Patron, L., "Online Learning Related Terminology". UNU Online Learning. 2003.
  http://www.onlinelearning.unu.edu/images/documents/UNU%20OL4-terms.pdf

[Pelechano, 2005]

- Pelechado, V., "Servicios Web. Estándares, Extensiones y Perspectivas de Futuro". ISIS. 2005.
  http://pangea.upv.es/N+ISIS05/documents/VicentePelechano/ServiciosWeb.pdf

[Piskurich, 2003]

- Piskurich, G. M., "The AMA Handbook of E-Learning: Effective Design, Implementation, and Technology Solutions". Amacon. 2003.

[Planet, 2006]

- PLANET Digital Repository. http://planet.urv.es/planetdr/

[Plekhanova, 2002]

- Plekhanova, V., "Intelligent Agent Software Engineering". Idea Group Publishing. 2002.

[POA, 2006]

- Programación Orientada a Aspectos (POA).
  http://es.wikipedia.org/wiki/Programaci%C3%B3n_Orientada_a_Aspectos

[Pozo, 2002]

- Pozo I., "Aprendices y maestros: la nueva cultura del aprendizaje.". Madrid: Alianza Editorial. 2002.

[RAE, 2006]

- Real Academia Española. http://www.rae.es/

[Ramsdell, 2004]

- Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification". 2004.
  http://www.rfc-archive.org/getrfc.php?rfc=3851

[Rehak,2003]

- Rehak, D., "e-Learning Standards Questions, Decisions, Actions". Learning Systems Architecture Lab, Carnegie Mellon University, Pittsburg. 2003.

[Reload, 2006]

- Programa Reload. http://www.reload.ac.uk/

[RMI, 2006]

- Java Remote Method Invocation (Java RMI). http://java.sun.com/products/jdk/rmi/

[Rodriguez et al., 2003]

- Rodríguez, J., Anido, L., Fernández, M. J., "How can the Web Services Paradigm improve the E-learning?". 2003.

[Rosenberg, 2001]

- Rosenberg, M. J. "e-Learning. Strategies for delivering knowledge in the digital age". McGraw-Hill. 2001.

[Sancho, 2002]

- Sancho, P., "Lenguajes de marcado y su aplicación en el dominio de las tecnologías de aprendizaje Web". Universidad Complutense de Madrid. 2002.

[Sem, 2005]

- Proyecto Sem B-UDDI. Universidad de Deusto. 2005. http://www.deli.deusto.es/SemBUDDI/

[Sharma y Kitchens, 2004]

- Sharma, S., Kitchens, F., "Web Services Architecture for M-Learning, Electronic Journal on e-Learning", Vol.2, Issue 1. 2004.

[Shaw y Garlan, 1996]

- Shaw, M., Garlan, D., "Software Architecture: Perspectives on an Emerging Discipline". Prentice Hall, 1996.

[Sigh, 2001]

- Sigh, H., "Learning Content Management Systems". E-learning magazine. 2001.

[SkillSoft, 2006]

- SkillSoft: Integrating the learning in the life of the enterprise. http://www.skillsoft.com/

[SOA, 2006]

- Service-Oriented Architecture (SOA). 2006. http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

[SQI, 2006]

- Simple Query Interface. http://nm.wu-wien.ac.at/e-learning/interoperability/SQI_V1.0beta_2005_04_13.pdf

[Soni et al., 1995]

- Soni, D., Nord, R. L., Hofmeister, C., "Software architecture in industrial applications". En: Actas de 17th International Conference on Software Engineering. New York: ACM Press. p.196-207. 1995.

[Sosteric y Hesemeier, 2002]

- Sosteric, M., Hesemeier, S., "When is a learning object not an object: a first step towards a theory of learning objects". En: International Review of Research in Open and Distance Learning vol. 3 num. 2. 2002. http://www.irrodl.org/content/v3.2/soc-hes.html

[Soto et al., 2006]

- Soto, J., García, E., Sánchez, S., "Repositorios Semánticos para Objetos de Aprendizaje". Expoelearning 2006. http://www.aefol.com/expoelearning2006/virtualcampus2.asp

[Sun, 2006]

- SOAP with Attachments API for Java (SAAJ). http://java.sun.com/webservices/saaj/index.jsp

[Systinet, 2006]

- Systinet. http://www.systinet.com

[TEI, 1994]

- The Text Encoding Initiative (TEI). 1994. http://www.tei-c.org/

[TIOBE, 2006]

- TIOBE Programming Community Index, http://www.tiobe.com/tpci.htm

[UDDI, 2006]

- UDDI Consortium. http://www.uddi.org

[UMA, 2006]

- Grupo de Ingeniería del Software de la Universidad de Málaga. http://www.lcc.uma.es/LCC?-f=indexlang.lcc&-l=spanish

[VPN, 2006]

- Virtual Private Network (VPN). 2006. http://es.wikipedia.org/wiki/Red_privada_virtual

[WebLogicPro, 2004]

- WebLogicPro Techical Publication. Julio/Agosto 2004, Vol.1 No. 2.

[W3C, 1996]

- W3C Cascading Style Sheets (CSS). 1996. http://www.w3.org/Style/CSS/

[W3C, 1998]

- W3C Working Group. eXtensible Markup Language (XML). 1998. http://www.w3.org/XML

[W3C, 1999a]

- W3C XSL Transformations (XSLT) Version 1.0 16 November 1999, James Clark. http://www.w3.org/TR/xslt

[W3C, 1999b]

- W3C The Extensible Stylesheet Language Family (XSL). http://www.w3.org/Style/XSL/

[W3C, 2001a]

- W3C Web Services Description Language (WSDL) 1.1, 15 March 2001, Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, http://www.w3.org/TR/wsdl

[W3C, 2001b]

- W3C XML Linking Language (XLink) Version 1.0 27 June 2001, Steve DeRose. http://www.w3.org/TR/xlink

[W3C, 2001c]

- W3C XML Key Management System, 30 March 2001. http://w3.org/TR/xkms

[W3C, 2001d]

- W3C XML Encryption, http://www.w3.org/Encryption/2001

[W3C, 2001e]

- W3C DAML+OIL. http://www.w3.org/TR/daml+oil-reference

[W3C, 2002]

- W3C XML-Signature Syntax and Processing, 12 February 2002, Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, Ed Simon, http://www.w3.org/TR/xmldsig-core

[W3C, 2003]

- W3C SOAP 1.2 Recommendation, 24 June 2003, Hugo Haas, Nilo Mitra, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, http://www.w3.org/TR/soap

[W3C, 2004a]

- W3C Working Group. Note, D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard. “Web Services Architecture”. 11 February 2004. http://www.w3.org/TR/ws-arch/

[W3C, 2004b]

- W3C Working Group. Note, D. Booth, H. Haas, D. Orchard. “Web Services Architecture Usage Scenarios”. 11 February 2004. http://www.w3.org/TR/ws-arch-scenarios/

[W3C, 2004c]

- W3C XML Schema. 2004. http://www.w3.org/XML/Schema

[W3C, 2004d]

- W3C Platform for Privacy Preferences (P3P 1.1) Project, 1 July 2004, Lorrie Cranor, Brooks Dobbs, Serge Egelman, Giles Hogben, Jack Humphrey, Marc Langheinrich, http://www.w3.org/P3P

[W3C, 2004e]

- W3C Web Services Addressing (WS-Addressing), 10 August 2004, Don Box, Erik Christensen, Francisco Curbera, Donald Ferguson, Jeffrey Frey, Marc Hadley, http://www.w3.org/Submission/ws-addressing

[W3C, 2004f]

- W3C WS-MessageDelivery Version 1.0, 26 April 2004, Anish Karmarkar, Ümit Yalçınalp, Mark Hapner, Frederick Hirsch, Dave Ingham, Mark Little, Michael Mahan, http://www.w3.org/Submission/ws-messagedelivery

[W3C, 2004g]

- W3C Resource Description Framework (RDF). http://www.w3.org/RDF/

[W3C, 2004h]

- W3C Web Ontology Language (OWL). http://www.w3.org/TR/owl-features/

[W3C, 2004i]

- W3C OWL-S Ontology Web Language - Services. http://www.w3.org/Submission/OWL-S/

[W3C, 2005]

- W3C Document Object Model (DOM). 2005. http://www.w3.org/DOM

[W3C, 2006]

- W3C Web Services Eventing (WS-Eventing). 2006. http://www.w3.org/Submission/WS-Eventing/

[Wilkes, 2005]

- Wilkes, L., "The Web Services Protocol Stack". 2005. http://roadmap.cbdiforum.com/reports/protocols/index.php

[Winer, 1999]

- Winer, D., "XML-RPC Specification". 1999. http://www.xmlrpc.com/

[Wooldridge, 2002]

- Wooldridge, M., "Introduction to MultiAgent Systems". John Wiley & Sons. 2002.

[WS-I, 2006]

- Web Services Interoperability Organization. http://www.ws-i.org.

[WSMO, 2005]

- WSMO: Web Services Modeling Ontology. http://www.wsmo.org/

[Xu et al., 2003]

- Xu, Z., Yin, Z., Saddik, A. E., "A Web Services Oriented Framework for Dynamic E-Learning Systems". 2003.