UNIVERSIDAD DE ALCALÁ
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS DOCTORAL

MECANISMOS SEMÁNTICOS ORIENTADOS A LA FLEXIBILIDAD DE LOS REPOSITORIOS PARA OBJETOS DE APRENDIZAJE

Autor:

Jesús Soto Carrión

Ingeniero en Informática

Directores:

Dr. Salvador Sánchez Alonso

Dra. Elena García Barríocanal

Alcalá de Henares, Febrero 2008

Dept. Ciencias de la Computación
Escuela Técnica Superior Ingeniería Informática
Ctra. Barcelona km. 33,600
28871 Alcalá de Henares (Madrid)
Teléfonos: 91 885 66 45
Fax: 91 885 66 46

Los Drs. D. Salvador Sánchez Alonso y Dña. Elena García Barriocanal, profesores del Departamento de Ciencias de la Computación de la Universidad de Alcalá, en calidad de directores del trabajo de tesis doctoral titulado **"Mecanismos semánticos orientados a la flexibilidad de los repositorios para objetos de aprendizaje"** y realizado por D. Jesús Soto Carrión,

HACEN CONSTAR:
Que dicho trabajo tiene suficientes méritos teóricos contrastados adecuadamente mediante las validaciones oportunas y aportaciones altamente novedosas. Por todo ello consideran que procede su defensa pública.

En Alcalá de Henares, a _____ de _____ de _____.

Dr. Salvador Sánchez Alonso

Dra. Elena García Barriocanal

Dept. Ciencias de la Computación
Escuela Técnica Superior Ingeniería Informática
Ctra. Barcelona km. 33,600
28871 Alcalá de Henares (Madrid)
Teléfonos: 91 885 66 45
Fax: 91 885 66 46

Dra. Dña. Enriqueta Muel Muel, Profesora Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial, en calidad de Directora del Departamento de Ciencias de la Computación.

CERTIFICA: Que la Tesis Doctoral titulada: **"Mecanismos semánticos orientados a la flexibilidad de los repositorios para objetos de aprendizaje",** realizado por D. Jesús Soto Carrión y dirigida por Dr. D. Salvador Sánchez Alonso y Dra. Dña. Elena García Barriocanal, reúne los requisitos para su presentación y defensa pública.

En Alcalá de Henares, a _____ de _____ de _____.

Fdo: Enriqueta Muel Muel

A mi mujer, Elisa, y al hijo que esperamos con toda ilusión.

Tabla de contenidos

Índice de tablas

Índice de figuras

Resumen

Los nuevos enfoques de reutilización de materiales didácticos en formato digital usan el concepto de "objeto de aprendizaje" como elemento clave para la creación de repositorios distribuidos, tales como MERLOT[1] o CAREO[2]. Dichos repositorios, tendrían la finalidad de describir los multiples recursos didácticos existentes en la Web, almacenando dichos recursos y sus metadatos (o solamente estos últimos), y posibilitando la realización de búsquedas.

Sin embargo, la existencia de diferentes definiciones de "objeto de aprendizaje" dificulta la gestión y tratamiento uniforme de los recursos, lo que sugiere una nueva generación de repositorios flexibles donde tengan lugar todas las conceptualizaciones del término. En este escenario, las ontologías juegan un importante rol para el soporte de un modelo semántico sólido que cumpla con los nuevos requisitos que la flexibilidad impone.

No obstante, la falta de flexibilidad no es la única carencia de que adolecen los actuales repositorios de objetos de aprendizaje. En esta investigación se ha analizado además la falta de cumplimiento de las especificaciones de los estándares actuales de e-learning. Las conceptualizaciones de "objeto de aprendizaje" definidas en IEEE

[1]Multimedia Educational Resource for Learning and Online Teaching - http://www.merlot.org
[2]Campus Alberta Repository of Education Objects - http://careo.ucalgary.ca/

LOM o en ADL SCORM no tienen lugar en la mayor parte de los repositorios, lo cual dificulta en gran medida tanto el procesamiento de la metainformación asociada a los objetos de aprendizaje como el conseguir utilizarla en sistemas gestores de aprendizaje (LMS) conformes a una especificación.

En la mayoría de las aplicaciones actuales no se tiene en cuenta un modelo de representación formal del conocimiento general que incorpore semántica computacional, dentro de la visión de lo que se ha dado en llamar Web semántica. Nuevas bases de conocimiento como OpenCyc, sugieren la posibilidad de definir las relaciones semánticas dentro de la información descrita. Conceptos de nuestro conocimiento general, tales como país , rey, médico, o también relaciones del tipo "X cerca de Y", pueden ser referenciados sin ambigüedades ni variaciones debidas a las posibles interpretaciones de los mismos haciendo uso de una base de conocimiento en forma de ontología como OpenCyc. Este hecho, genera un nuevo espectro de posibilidades de inferencia sobre los registros que contienen la información de metadatos de los objetos de aprendizaje. Para aprovechar al máximo el conocimiento almacenado en los registros de metadatos de los objetos de aprendizaje almacenados en el repositorio, se propone un esquema que permita describir el significado de la metainformación dentro de los registros de metadatos del repositorio.

El propósito de esta investigación es definir este tipo de conocimiento mediante un esquema que permita describir el significado de la metainformación existente en los registros de metadatos de un repositorio de objetos de aprendizaje. Para ello, será necesario definir una nueva arquitectura de repositorio, basada en un modelo formal representado en un lenguaje de ontologías (concretamente OWL) para el procesamiento automático de la metainformación por parte de agentes software externos al repositorio. Esta propuesta aportará nuevas y más potentes funcionalidades sobre

los repositorios actuales, gracias a la posibilidad de ejecutar inferencias sobre el conocimiento albergado en los registros del repositorio.

Palabras clave: Repositorio, objeto de aprendizaje, metadatos, Web semántica, ontología, e-learning, common sense.

Abstract

Current approaches towards enhancing reusability of learning materials in digital format use the concept of learning object as the key element of a new approach based on the distributed availability of learning resources. This model relies on the existence of learning object repositories creation, such as MERLOT or CAREO, whose main purpose is to classify different Web resources through metadata descriptions.

However, the existence of multiple learning object definitions point out the need of a new generation of flexible learning object repositories capable to fit any existent conceptualization of the term. In this scenario, ontologies play an important role as the basis to provide a sound semantic model that fulfills the new requirements.

Nonetheless, flexibility is not the only lack in these repositories. The lack of conformance with current specifications and standards has also been marked as an issue and thus analyzed in the present research. The learning object conceptualization, as described by current standards and specifications such as IEEE LOM and ADL SCORM, is not the one used in the most repositories. This fact hampers the metainformation management process associated to the use of learning objects.

Today, most applications do not use any formal model of commonsense knowledge.

Advanced knowledge bases such as Opencyc make it possible to create semantic relationships between the concepts defined, which allows concepts such as country, king, doctor or relationships like X near Y to be linked without ambiguities nor changes due to possible interpretations. This fact, if applied to learning object repositories, provides new inference possibilities over learning object metadata records and allows meaningful descriptions of the metainformation stored in those repositories, as it turns text-based metadata records into machine-understandable semantic metadata records.

This dissertation proposes the use of this kind of knowledge through the definition of a semantic schema aimed at allowing the meaningful description of the information in existing metadata records repositories. To that end, it will be necessary to define the complete architecture of what it will be called a semantic learning object repository, based on a formal model described in an ontology language (OWL in particular). This architecture will facilitate external software applications or agents the metainformation management tasks. Running inferences on the knowledge stored inside the repositories'records, will extend current repositories'functionalities, introducing key advantages and features.

Keywords: learning objet repository, learning objects, metadata, ontology, e-learning, common sense.

Agradecimientos

Escribir estas líneas supone una grata satisfacción por el largo trabajo de investigación llevado a término, objetivo que no hubiera sido posible sin la ayuda recibida por mis directores, familiares y amigos. En primer lugar, Salvador Sánchez Alonso, director y amigo que tanto me ha ayudado desde el inicio del trabajo, ¡gracias por toda tu colaboración!; pocos directores están a la altura del trabajo que has realizado. He tenido también la suerte de tener a Elena García Barríocanal como co-directora, que ha sabido dirigir y enfocar la investigación de forma impecable. Gracias, cómo no a Miguel Angel Sicilia, gran investigador que nos ha ayudado con la aportación de sus valiosas ideas.

En lo personal, en estos momentos soy un hombre muy feliz: tengo una mujer encantadora y un hijo dentro de su vientre al que esperamos con toda ilusión. Gracias Elisa, sin tu ayuda y apoyo moral no hubiese terminado este trabajo. Tampoco puedo olvidar a nuestra familia, padres y hermanos, nos han apoyado con cariño en todo el proceso.

Para terminar, un agradecimiento especial a Víctor Martín, que tanto me ha ayudado con su colaboración desinteresada.

¡Gracias a todos por vuestra ayuda!

Capítulo 1

Introducción

Aquel que duda y no investiga, se torna no sólo infeliz,
sino también injusto.
Blaise Pascal

El objetivo de este capítulo es ofrecer una síntesis del problema que se pretende resolver, describiendo los objetivos y el método de trabajo llevado a cabo en la investigación. Se resumen las principales aportaciones, que se irán posteriormente detallando a lo largo del documento.

1.1. Planteamiento del problema

Los nuevos enfoques de reutilización de materiales didácticos emplean el concepto "objeto de aprendizaje" (learning object) como pieza base para la creación de repositorios distribuidos. Un repositorio de objetos para el aprendizaje es un sistema software que almacena recursos educativos y sus metadatos (o solamente estos últimos), y que proporciona algún tipo de interfaz de búsqueda de los mismos, bien para interacción con humanos o con otros sistemas software. Los repositorios proporcionan acceso a colecciones de recursos educativos generalmente en formato electrónico, si bien la mayoría no almacenan los recursos educativos en sí, sino solamente sus

metadatos lo que hace posible encontrar el mismo recurso a través de diferentes repositorios. La funcionalidad fundamental de un repositorio de objetos de aprendizaje es la búsqueda de recursos educativos, observándose dos grandes tipos de repositorios:

- Repositorios con interfaces de búsqueda interactivos, para uso de humanos.
- Repositorios con interfaces de consulta que puedan ser utilizados por aplicaciones externas, por ejemplo, mediante Servicios Web.

En ocasiones, la misma forma de búsqueda puede servir para los dos usos. Hay que tener en cuenta que la búsqueda mediante los habituales mecanismos de recuperación de información [RBY99] de propósito general (como los que usan los buscadores de Internet) debe complementarse con una búsqueda basada en el análisis de la información de los distintos campos de metadatos. No obstante, esas interfaces a veces tampoco resultan satisfactorias debido a los resultados poco relevantes obtenidos o la imprecisión de los mismos, por lo que actualmente se investiga en técnicas avanzadas que permitan hacer uso de conocimiento preciso sobre el dominio de los metadatos.

En esta investigación se estudiaran las distintas interfaces utilizadas por los repositorios, así como los distintos modelos de metadatos utilizados para su gestión. Algunos de los problemas encontrados (y que serán estudiados con mayor detalle en el capítulo 2) son los siguientes:

- Existen múltiples definiciones de objeto de aprendizaje. Cada repositorio hace uso de una conceptualización del término o bien utiliza el perfil determinado de un estándar, limitando así la capacidad de compartir y tratar de forma automatizada la información albergada en sus registros.
- Dificultad para encontrar en el repositorio los objetos de aprendizaje así como los recursos de los que se compone y utiliza. Algunos repositorios ni siquiera

utilizan los estándares propuestos por organizaciones de normalización (como el IEEE o IMS) para los metadatos de objetos de aprendizaje.

- Uso del lenguaje natural - dificultad de procesar la información e inferir determinado tipo de conocimiento. La información introducida en los campos de metadatos carece de un lenguaje formal que permita describir de forma precisa y sin ambigüedades un recurso, así como las relaciones existentes con otros conceptos de nuestro conocimiento general o específico.

Las nuevas terminologías utilizadas para clasificar actores, actividades y artefactos (elementos en sentido general) son el resultado de los esfuerzos recientes de estandarización de las tecnologías del aprendizaje. Así, el estudio de McGreal [McG04] intenta unificar las diferentes definiciones de 'learning object'que varían desde la más general a la más específica. Además, Downes [Dow04] ha introducido la noción de "perfil de un recurso" como mecanismo de descripción amplio y variado, al intentar conseguir una conclusión alrededor de si los objetos de aprendizaje pueden ser recursos de clases básicas o fundamentales.

Estos ensayos muestran que las entidades utilizadas para la elaboración de materiales de un repositorio requieren un alto grado de flexibilidad en las caracterizaciones de los objetos de aprendizaje. La inexistencia de un vocabulario común, así como la coexistencia de diferentes definiciones de *objeto de aprendizaje*, señalan la necesidad de una nueva generación de repositorios flexibles donde todas las conceptualizaciones existentes tengan un lugar.

Greenberg [Gre03] define los metadatos como: *"datos estructurados sobre un objeto orientados a permitir funciones"*. Según está definición, existe una balanza entre la flexibilidad de descripción y la especificidad requerida para el uso concreto de los

metadatos, en el caso que las funciones soportadas sean completamente o parcialmente automáticas.

Para permitir comparar distintas representaciones de objetos de aprendizaje de acuerdo con la flexibilidad de su cobertura y términos asumidos de propiedades es necesario aplicar una disciplina basada en una ontología formal [WG01]. En efecto, las ontologías desempeñan un papel importante como soporte de modelos semánticos sólidos que aportan un numero de nuevas posibilidades relacionadas con los procesos automáticos, tales como búsquedas, recuperación o composición de nuevos materiales didácticos y otros. La existencia de esquemas basados en ontologías es esencial cuando algunas de las funciones están siendo delegadas a sistemas automatizados o semiautomatizados, siguiendo la visión de la Web semántica [BLHL01].

1.2. Objetivos y aportaciones originales

El objetivo principal de la presente investigación es aportar un mecanismo que proporcione a un repositorio de objetos de aprendizaje la capacidad de gestionar de forma flexible la semántica de las definiciones de los objetos que almacena, facilitando a su vez el procesamiento automatizado de los recursos a aplicaciones software externas.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

(O1) Estudiar las capacidades de representación de los repositorios de objetos de aprendizaje existentes en relación a su uso por parte de sistemas software.

(O2) Proponer una representación para el almacenamiento semántico de objetos de aprendizaje con los siguientes requisitos:

  (O2.1) Dar cabida a objetos de aprendizaje concebidos bajo diferentes conceptualizaciones del término “learning object”.

(O2.2) Cubrir los estándares actuales de metadatos (particularmente IEEE LOM).

(O2.3) Introducir semántica computacional en las descripciones de metadatos de los objetos de aprendizaje.

(O3) Diseñar una arquitectura para la gestión flexible de objetos de aprendizaje que utilice la representación propuesta en el objetivo O2, que extiende funcionalidades que proporciona los repositorios actuales. Dicha arquitectura estará orientada a permitir funcionalidades extendidas con respecto a las que proporcionan los repositorios actuales.

Todos estos objetivos han sido objeto de investigación previa, que se ha materializado en diversas publicaciones científicas desarrolladas por el autor donde se describe, por una parte, el ciclo de vida semántico de los metadatos en un repositorio de objetos de aprendizaje [nSGSAS05], seguido de la descripción de recursos dentro del repositorio [SnSSA05] y el modelo formal y el diseño del repositorio semántico [SSAnS05]. También se han realizado trabajos relacionados con la trazabilidad del aprendizaje y la planificación por costes del estudio [SG05].

La aportación fundamental de esta investigación es el diseño de una arquitectura flexible para los repositorios de objetos de aprendizaje, que permita inferencia y delegación de tareas a aplicaciones o agentes software. Entre las aportaciones significativas que surgen como resultado del proceso de investigación, es posible destacar las siguientes:

- Estudio de los mecanismos de persistencia para el lenguaje de ontologías web (OWL).
- Resultados de evaluación de usuarios sobre la descripción semántica de los metadatos en el estándar IEEE LOM.

- Posibilidad de describir los recursos almacenados en un repositorio de objetos de aprendizaje con distintas ontologías.
- Mejora de las actuales funciones de búsqueda y navegación que proporcionan los repositorios actuales.
- Diseño e implementación de un prototipo de repositorio semántico de objetos de aprendizaje.
- Generalización del concepto *repositorio semántico*.

1.3. Método de trabajo

En esta sección se describe la planificación general del trabajo realizado, así como el método de evaluación seguido.

1.3.1. Planificación general

La metodología seguida en esta investigación comprende las siguientes fases:

1. **Estudio del estado de la cuestión**: se estudian el concepto y usos de los objetos de aprendizaje, así como las diferentes recomendaciones y estándares de metadatos aplicables a los mismos. Además se aborda un estudio de las características de los repositorios de objetos de aprendizaje más importantes.
2. **Descripción del contexto del problema**: el problema se plantea en el marco de la representación utilizada en los repositorios de los objetos de aprendizaje. Se parte de la constatación de las significativas carencias existentes en los metadatos de los objetos de aprendizaje, tal y como se definen en la actualidad, haciendo énfasis en las dificultades de su utilización por parte de sistemas automatizados. Se expone la necesidad de modelar una representación que soporte la semántica de los metadatos para facilitar los procesos de búsqueda y

navegación, así como el tratamiento automatizado de los mismos por medio de sistemas o aplicaciones software.

3. **Definición del modelo**: se propone un modelo de repositorio basado en una ontología capaz de albergar las diferentes conceptualizaciones del término "objeto de aprendizaje". Se parte del estudio de las distintas acepciones del término hasta la definición del modelo formal de la ontología.

4. **Diseño e implementación de un prototipo**: se ofrece un panorama de los distintos estándares y herramientas de la Web semántica que guardan relación con la elaboración del diseño del prototipo. Seguidamente se propone el diseño del prototipo del repositorio semántico que utiliza la ontología formal propuesta en la fase anterior. Finalmente se realiza la implementación el núcleo del prototipo del repositorio semántico con las funciones básicas adaptadas al estudio del modelo.

5. **Evaluación de los objetivos**: Haciendo uso del prototipo implementado en la fase anterior, se lleva a cabo la evaluación del modelo propuesto en varios ámbitos. Por un lado, se comprueba la compleción del modelo propuesto, verificando la consistencia de la ontología creada para el repositorio. Por otro lado, se evalúa la satisfacibilidad de los mecanismos semánticos propuestos. Por último, se contrasta su factibilidad técnica, analizando el prototipo creado con una serie de casos de prueba que permiten comprobar el cumplimiento de los objetivos planteados en la investigación.

6. **Formulación de conclusiones**. Se concluye con un análisis del grado de cobertura de los objetivos alcanzados, y una exposición de líneas futuras de investigación y ampliación del trabajo realizado.

1.4. Estructura del documento

Esta memoria de tesis se estructura como se describe a continuación.

1. En el presente capítulo se recogen los objetivos y aportaciones de la investigación, y se presenta la metodología de trabajo.

2. En el capítulo 2 se describe el estado de la cuestión en lo relativo a los repositorios de objetos de aprendizaje actuales, las características de almacenamiento y recuperación de metadatos en relación con el tratamiento automático por medio de agentes software, centrando la discusión en las carencias de dichos repositorios.

3. En el capítulo 3 se detalla el problema, describiendo los actuales inconvenientes y limitaciones de los repositorios de objetos didácticos. A la luz de las limitaciones observadas y tras exponer sus posibles soluciones, se muestra la necesidad de diseñar un repositorio semántico de objetos de aprendizaje capaz de representar con semántica computacional el significado de la información existente en los metadatos como paso previo para poder habilitar su uso por parte de agentes software y potenciar las capacidades de búsqueda y navegación en los mismos.

4. En el capítulo 4 en primer lugar se describen las tecnologías y herramientas de la Web semántica, útiles en relación con la investigación presente, para crear aplicaciones que entiendan el significado de la información. Seguidamente se describen las características de diseño del repositorio semántico de objetos de aprendizaje: La representación semántica de metadatos de objetos de aprendizaje, la técnica propuesta para la descripción de recursos utilizando conceptos de ontologías y un diseño del prototipo del repositorio.

5. En el capítulo 5 ofrece una descripción técnica de los componentes más relevantes del prototipo de repositorio semántico, desde la visión general de la

arquitectura hasta los detalles de los componentes más importantes.

6. En el capítulo 6 se describe la evaluación realizada al prototipo implementado, así como los casos de prueba utilizados para demostrar el cumplimiento de los objetivos planteados.

7. Finalmente, en el capítulo 7 se ofrecen las conclusiones del trabajo realizado y se esbozan algunas líneas futuras de investigación relacionadas con la aplicación de la técnica descrita y su posible ampliación.

Capítulo 2

Estado de la cuestión

"People have silly reasons why computers don't really think. The answer is we haven't programmed them right; they just don't have much common sense."

Marvin Minsky

En este capítulo se ofrece una descripción del estado actual de las investigaciones y trabajos desarrollados en el área de los repositorios de objetos de aprendizaje, así como los estándares y especificaciones de metadatos utilizados y su relación con la automatización de los procesos de almacenamiento y búsqueda. La discusión se centra fundamentalmente en aquellos aspectos que resultan relevantes para los objetivos del presente trabajo.

2.1. El nuevo espacio europeo de enseñanza

En esta sección se explican los distintos esfuerzos realizados por la Unión Europea en materia de enseñanza electrónica, así como el fuerte impulso introducido en materias de enseñanza continua durante toda la vida[1]. Se expone así, para explicar el entorno político-social en el cual surge esta investigación.

[1]Del término inglés: lifelong learning

2.1.1. La educación y formación continua como motor de la "modernización social europea"

El Consejo Europeo en la reunión celebrada en el año 2001 en Lisboa fijó un nuevo objetivo estratégico:

> "*Convertirse en la economía basada en el conocimiento más competitiva y dinámica del mundo, capaz de crecer económicamente de manera sostenible con más y mejores empleos y con mayor cohesión social.*"

El Consejo señaló que esos cambios exigían no solamente una transformación radical de la economía europea, sino también un programa ambicioso de modernización del bienestar social y de los sistemas educativos. Nunca antes el Consejo Europeo había reconocido de esa manera el papel que desempeñan los sistemas educativos y de formación dentro de la estrategia económica y social y el futuro de la Unión. La modernización social planteó una reflexión sobre la educación, concluyendo que para ser *más competitivos* en una economía *basada en el conocimiento* era necesario:

1. Mejorar la calidad y la eficacia de los sistemas de educación y formación.
2. Facilitar el acceso a dichos sistemas.
3. Abrir los sistemas de educación y formación al mundo exterior.

Como consecuencia de estas reflexiones, el Consejo Europeo solicitó al Consejo Superior de Educación que emprendiese una reflexión general en el Consejo sobre los futuros objetivos precisos de los sistemas educativos, centrada en intereses y prioridades comunes y respetuosa al mismo tiempo con la diversidad nacional.

El resultado de este estudio definió los objetivos respecto a la educación que en el 2010 deberían estar concluidos:

- Se llegará a la más alta calidad en cuanto a la educación y a la formación, y se considerará a Europa como una referencia mundial por la calidad y pertinencia de sus sistemas educativos y de sus instituciones.

- Los sistemas de educación y formación europeos serán lo suficientemente compatibles como para que los ciudadanos puedan estudiar en cualquier lugar de la Unión Europea.

- Las personas que posean títulos y conocimientos adquiridos en cualquier lugar de la UE podrán convalidarlos efectivamente en toda la Unión a efectos de sus carreras y de la formación complementaria.

- Los ciudadanos europeos de todas las edades tendrán acceso a la educación permanente.

Una de las conclusiones fundamentales fue que para conseguir alcanzar el nuevo objetivo estratégico era necesario revisar y mejorar las nuevas políticas relativas a la sociedad de la información. Consecuencia directa de ello fue la definición de las acciones necesarias en el plan "eEurope" utilizadas para construir la economía digital basada en el conocimiento como pilar de un poderoso motor para el crecimiento, la competitividad y el empleo. La repercusión actual de este plan ha quedado plasmada en diversas direcciones:

- e-Learning: la utilización de las nuevas tecnologías multimedia e Internet para la mejora en la calidad de la enseñanza, proporcionando el acceso a recursos y servicios así como la colaboración e intercambio a distancia de los mismos.

- e-Government: los servicios de la administración pública a disposición del ciudadano en Internet. Los resultados de este programa esperan mejorar la calidad de vida de los europeos.

- e-Health: servicios de salud pública por Internet. Objetivo clave de este programa son la reducción de las listas de espera, la recepción de los resultados de las pruebas por Internet, historial electrónico compartido con cualquier hospital, telemedicina, etcetera.

- e-Business: aprovechar el uso de la tecnología para hacer más eficiente el negocio entre las empresas europeas, la creación de estándares de intercambio de información entre distintos negocios así como la seguridad en las transacciones electrónicas son algunos de los campos destinados a esta línea de actuación.

- e-Inclusion: tiene como objetivo facilitar la sociedad de la información a todos los ciudadanos de la Unión Europea, analizando la dimensión social y regional así como la convergencia mediante plataformas basadas en nuevas tecnologías de la información.

Este plan refleja los trabajos que están siendo desempeñados para que Europa mantenga su posición de vanguardia en ámbitos tecnológicos clave, tales como las comunicaciones móviles.

Como se ha visto, el e-learning es uno de los pilares del proyecto e-Europe. La formación permanente y el proceso de Bolonia [Kee06] son los puntos fuertes de aplicación de este tipo de tecnología. La Unión entendió que la educación y formación continua durante toda la vida es el método más efectivo para realizar la *modernización social* necesaria. Pero la integración de la familia, el trabajo y la formación continua para la excelencia, pasan por acomodar los horarios de enseñanza en función del tipo de estudiante. Por lo tanto, la pieza clave de la formación continua es el e-learning, adecuado a cualquier edad, accesible desde cualquier lugar y en cualquier momento.

Por ello existen esfuerzos comunes en crear tecnologías que rompan las barreras temporales y espaciales de la educación y formación. En este nuevo espacio europeo la organización de los recursos educativos cobra un aspecto importante debido a la gran cantidad de contenidos generados por entidades subvencionadas en gran parte por la Unión Europea. También es de resaltar el trabajo necesario para almacenar los contenidos y localizarlos de forma rápida y exacta.

A lo largo de la próxima sección se exponen las distintas tecnologías que permiten administrar y encapsular los contenidos educativos en nuevas entidades abstractas llamadas "objetos de aprendizaje" para posibilitar la reutilización de los contenidos entre distintos sistemas de enseñanza.

2.2. Objetos de aprendizaje

El concepto "e-learning" según [Mor02] se define como el aprendizaje que utiliza herramientas relacionadas con las tecnologías de la información y las comunicaciones (Internet, intranets, redes inalámbricas, ordenadores personales, ordenadores de mano o televisión interactiva) aunque también puede utilizar la tecnología como simple soporte de una enseñanza similar a la tradicional, por ejemplo utilizando pizarras electrónicas o videoconferencia. Es por tanto un nuevo tipo de aprendizaje que requiere de una plataforma de gestión y distribución de contenidos didácticos.

Poco a poco se ha hecho evidente la necesidad de estructurar y definir los cursos distribuidos sobre una plataforma de e-learning, un tema, un ejercicio o un grupo de prácticas pueden ser recursos utilizados dentro de varios cursos. Desde esta visión modular, se ha definido el concepto de objeto de aprendizaje, discutido a lo largo de esta sección. Los objetos de aprendizaje pueden agregarse para conformar recursos

educativos más complejos que tengan sentido. Es el uso de estas pequeñas piezas y la posibilidad de ensamblarlas a voluntad para construir con ellas modelos agregados de estructura superior al estilo de los bloques LEGO [Wil99] es una de las características más atractivas del e-learning y lo que ha multiplicado el interés generado por esta tecnología.

En esta sección se discute sobre las más importantes definiciones del concepto de objeto de aprendizaje, y se proporciona una definición que será utilizada en el resto del trabajo. Además, se abordan sus propiedades y características técnicas y se analizan las actuales propuestas de estandarización relacionados con los objetos de aprendizaje.

2.2.1. Definición de objeto de aprendizaje

El estándar LOM [LTS02], define objeto de aprendizaje como:

> *"Cualquier entidad, digital o no digital, que puede ser utilizada para el aprendizaje, la educación o la enseñanza"*

Esta definición, deliberadamente genérica, presenta un concepto de objeto de aprendizaje débil desde el punto de vista teórico [BC01]. Diversos autores señalan la necesidad de establecer una definición sin ambigüedades que resulte realmente útil a efectos prácticos. A este respecto, autores de referencia Sosteric y Hesemeier [MYO01], Wiley [Wil02] y Polsani [Pol03], han considerado las definiciones existentes en la bibliografía y presentado su propia definición de cara a posibilitar la identificación, desarrollo y análisis de los objetos de aprendizaje. De entre las muchas definiciones existentes, la definición crítica de Polsani [Pol03], una de las más citadas en la bibliografía y consistente con las definiciones proporcionadas por otros autores, define objeto de aprendizaje como:

> *"Unidad didáctica independiente y autocontenida predispuesta para su reutilización en diversos contextos educativos"*

No obstante, para los propósitos de este trabajo resulta especialmente importante reseñar la naturaleza digital de los objetos de aprendizaje, puesto que las definiciones existentes a menudo obvian este detalle. Wiley [Wil02] aporta la siguiente definición:

> *"Objeto de aprendizaje es cualquier recurso digital que pueda ser reutilizado como soporte para el aprendizaje"*

Otros autores han desechado las definiciones teóricas y han definido el concepto de objeto de aprendizaje mediante la enumeración de un conjunto de características que dichos elementos deben reunir, idealmente, para ser considerados como tales. A este respecto, Longmire [Lon00] enumera los atributos de un objeto de aprendizaje reutilizable:

- Es modular, autocontenido y puede utilizarse en diferentes aplicaciones y entornos.
- No es secuencial.
- Satisface un único objetivo didáctico.
- Está orientado a un público amplio (es decir, puede adaptarse a destinatarios distintos a los originales).
- Es coherente y unitario dentro de un esquema predeterminado, de manera que mediante un número limitado de meta-etiquetas se pueda capturar la idea principal.
- No está en ningún formato específico, por lo que puede reutilizarse para diferentes propósitos sin que se alteren sus valores esenciales, ni el contenido de su texto, imágenes o datos.

Esta clasificación de atributos es compartida por otros autores como Friesen o Polsani, que únicamente difieren en la denominación de los atributos: accesibilidad, modularidad e interoperabilidad [Fri01]. Por su parte, Rehak y Mason añaden un atributo más, la perdurabilidad, que definen como la "inmunidad" ante los cambios en el software y hardware que los utilizan [RM03].

No obstante, esta definición resulta aún demasiado general para los objetivos del presente trabajo, ya que no considera la orientación de los metadatos hacia determinados procesos especializados de gestión y tratamiento de los objetos de aprendizaje, como su intercambio, composición e incluso su producción personalizada [Mar02]. Por ello, se propone la siguiente matización sobre la definición anterior junto con la aportación realizada por la tesis de Salvador Sánchez [Alo05] de cara a obtener una definición de objeto de aprendizaje que sirva como referencia para el presente trabajo:

> *"Unidad didáctica, independiente, autocontenida y perdurable, predispuesta para su reutilización en diversos contextos educativos mediante la inclusión de información autodescriptiva en forma de metadatos en formato digital estandarizados específicamente orientados a la automatización de procesos de gestión"*

2.2.2. Metadatos en los objetos de aprendizaje

Los metadatos son datos acerca de los datos. Formalmente, se definen como datos relativos a ciertas propiedades asociadas a una entidad que permiten dotar a la misma de la capacidad de crear información superior a la que podría generar por sí sola. En el ámbito de los objetos de aprendizaje, el uso de metadatos es universalmente aceptado como medio para aumentar su calidad y reusabilidad [SG03], ya que una correcta anotación de metadatos permite conocer los contenidos y objetivos

didácticos de cada objeto a priori y facilita tanto el almacenamiento de los mismos en repositorios, como los procesos de búsqueda, selección y recuperación [Sin00].

Aunque existe consenso acerca de la necesidad de utilizar un registro de metadatos que describa el contenido de los objetos de aprendizaje, en la literatura se pueden encontrar dos posturas divididas sobre la forma de asociación entre ambos:

- Un primer grupo de autores postula que el objeto de aprendizaje final se compone en realidad de dos partes: el contenido real del objeto (texto, imágenes, video, etc.) y un registro de metadatos que describe dicho contenido. Para la mayoría de autores, el registro de metadatos resulta esencial cuando se trata de reutilizar el objeto, y por tanto debe almacenarse "empaquetado" junto con el contenido formando una unidad dispuesta para ser archivada en un repositorio y posteriormente reutilizada [Lon00, Smy03].

- Otros autores mantienen que aunque los metadatos puedan ser asociados al contenido, no tienen por qué necesariamente ir asociados con ellos ni almacenarse conjuntamente [Rob02, MR01]. Este modelo se caracteriza por la separación de funciones entre los creadores del contenido y las personas que crean los metadatos, existiendo una relación de relativa independencia entre ambos similar a la separación entre el registro de un catálogo en una biblioteca y el libro al que hace referencia. Un estudio comparativo de los más importantes repositorios en línea demuestra que la mayoría de ellos sigue este modelo [ND02], en el cual se proporcionan miles de referencias sobre elementos didácticos pero no se almacenan los objetos de aprendizaje en sí. Estos repositorios, entre los que destaca MERLOT [Caf02], sólo guardan información de metadatos sobre los objetos de aprendizaje y un enlace que permite acceder a los mismos. Los contenidos

finales se encuentran distribuidos por todo Internet.

El estudio y definición de los metadatos para los objetos de aprendizaje es uno de los principales campos de investigación dentro del *e-learning*. Todos los esfuerzos importantes derivan del conjunto de elementos de metadatos de Dublin Core Metadata Initiative [DCM03], organización dedicada a "promover la adopción generalizada de estándares de metadatos interoperables y el desarrollo de vocabularios especializados de metadatos para describir recursos con el objetivo de facilitar la creación de sistemas inteligentes de búsqueda de información". El conjunto de elementos de metadatos de Dublin Core constituye un medio para la descripción de recursos de información, entendidos como "cualquier cosa que tenga identidad" [BLFIM98], y es por tanto la especificación más amplia de metadatos.

En el caso particular de los objetos de aprendizaje, los metadatos resultan especialmente importantes de cara a la búsqueda y reutilización de los recursos. Por ello, IEEE ha desarrollado LOM [LTS02], una especificación estándar de metadatos para objetos de aprendizaje basada en Dublin Core y llevada a cabo con el esfuerzo conjunto de destacadas organizaciones como IMS[2], ARIADNE[3] o AICC[4]. LOM "determina un esquema de datos conceptual que define la estructura de una instancia de metadatos para un objeto de aprendizaje. Dicha instancia describe características del objeto agrupadas en categorías: general, ciclo de vida, meta-metadatos, educativas, técnicas, derechos, relación, anotación y clasificación. Su propósito es facilitar la búsqueda, evaluación, adquisición y uso de objetos de aprendizaje por parte de los alumnos, instructores o sistemas automatizados, así como el intercambio de los mismos y su uso compartido, permitiendo el desarrollo de catálogos e inventarios"

[2]IMS Learning Global Consortium - http://www.imsglobal.org/
[3]ARIADNE - http://www.ariadne-eu.org/index.php
[4]Aviation Industry CBT Commitee - http://aicc.org/

[LTS02].

No obstante, en general LOM no es utilizado directamente como esquema final, sino como referencia fundamental para esfuerzos de estandarización a nivel local o regional basados en LOM tales como CanCore [FFR02], FAILTE [Sla01], The Le@rning Federation Metadata Application Profile [TLF03] o UK LOM Core [Cam04]. Estos esfuerzos se constituyen generalmente en perfiles de aplicación, término utilizado para definir caracterizaciones del estándar dirigidas a una comunidad particular de implementadores con requisitos de aplicación comunes [Lyn97]. Un perfil de aplicación se define pues como "un compendio de elementos de metadatos seleccionados a partir de uno o más esquemas existentes para formar un nuevo esquema combinado" [DHSW02]. El nuevo esquema constituye un nuevo conjunto de metadatos cuyo propósito es cubrir los requisitos funcionales de una aplicación concreta o de una comunidad de práctica y proporcionar directrices a los implementadores de metadatos, manteniendo la capacidad de interoperar con aplicaciones que trabajan con recursos definidos según los esquemas originales. Por ejemplo, el *Le@rning Federation Metadata Application Profile* [MW03], un perfil orientado a la educación en las escuelas de Australia y Nueva Zelanda, está formado por elementos de metadatos tomados del Dublin Core Metadata Element Set [DCM03], de Dublin Core Qualifiers [DCM00], de EdNA [EdN02] y de LOM.

2.3. Usos de los objetos de aprendizaje

Se pueden hacer múltiples clasificaciones para comprender los usos posibles que puedan darse a un objeto de aprendizaje. A continuación se expone un breve análisis de clasificación basado en los criterios expuestos en el artículo [SS05].

- Atendiendo al usuario de los objetos de aprendizaje: según esta clasificación, un objeto de aprendizaje lo puede utilizar un profesor o un alumno. El profesor puede utilizarlo en diferentes escenarios dentro de una clase como herramienta de enseñanza, para componer nuevos materiales didácticos, o como elemento de comparación y validación de otros contenidos. El alumno en cambio es el receptor del contenido, así como la parte activa del proceso de aprendizaje recogido en el diseño interno del mismo. El éxito completo recogido por distintos enfoques de diseño pedagógico e instructivo recala en la consecución de los objetivos propuestos en el objeto de aprendizaje.

- Atendiendo a la herramienta en que se utilizan los objetos de aprendizaje: en LMS o en repositorios.

    - Sistemas de gestión del aprendizaje (LMS): este tipo de sistemas proporcionan un entorno de ejecución para los objetos de aprendizaje, de tal forma que el seguimiento del aprendizaje es monitorizado por un "motor software interno" capaz de ejecutar y entender los estándares de secuenciación y navegación para la consecución de objetivos. La mayoría de estos sistemas también proporcionan un entorno colaborativo para el trabajo en grupo, así como herramientas para la comunicación: foros, chats u otros medios. Algunos permiten además la creación de objetos de aprendizaje, como Moodle, incluyendo capacidades de reutilización de objetos con el apoyo de diversos mecanismos de conexión entre diferentes repositorios.

- Repositorios de objetos de aprendizaje: se verán detallados en la siguiente sección. Son los almacenes de los objetos de aprendizaje, permiten albergar diferentes objetos y habilitan los mecanismos de acceso remoto para la reutilización de los mismos. Gracias a este tipo de almacenes es posible fomentar la reutilización de los contenidos didácticos, de tal forma que el proceso de generación de nuevos contenidos pueda apoyarse en los ya existentes. Incluso en algunos casos sería posible la autogeneración[5] de un curso en función del perfil del usuario.

- Atendiendo a las modalidades de enseñanza, los objetos de aprendizaje se pueden usar en la enseñanza presencial, semi-presencial (blended learning) o en la enseñanza a distancia con el apoyo de TIC (e-learning)[6] completamente online.

- Atendiendo a la metodología de enseñanza-aprendizaje utilizada: dependiendo de si el profesor utiliza una metodología de enseñanza conductista, cognitivista o socio-constructivista, utilizará diferentes objetos de aprendizaje con distintas funcionalidades. Desde este punto de vista, el desarrollo de un objeto de aprendizaje normalmente se basa en una metodología de enseñanza-aprendizaje específica, esto es, actualmente cada profesor desarrolla un objeto de aprendizaje acorde a las necesidades de sus alumnos y a su metodología de enseñanza.

- Atendiendo al tipo de actividades que se pueden desarrollar en un proceso formativo: los objetos de aprendizaje se pueden utilizar en actividades de autoformación si el alumno no desea ningún tipo de seguimiento por parte de un tutor, en actividades de autoaprendizaje si el alumno configura lo que desea aprender,

[5]Si un usuario desea realizar un curso de "inteligencia artificial" y este no posee conocimientos de estadística, una herramienta de autor inteligente sería capaz de conectarse a un repositorio y obtener un objeto de aprendizaje compuesto (curso) sobre estadística, además de enlazarlo en una secuenciación correcta para la consecución de los objetivos.

[6]La enseñanza completamente online, sin presencialidad, mediada por tecnologías de información y comunicaciones (TICs).

en actividades adaptativas si se configuran y secuencian los contenidos en función de las capacidades de un alumno o en actividades colaborativas si se desea que un tutor guíe y evalúe el avance del alumno.

2.4. Estándares y especificaciones relevantes para la investigación

Un estándar es un conjunto de especificaciones técnicas documentadas que regulan la realización de un proceso o la fabricación de un producto. Si de lo que se trata es de normalizar la elaboración de un producto, el objetivo de la estandarización es fundamentalmente la interoperabilidad entre artículos construidos por diferentes fabricantes.

La elaboración de un estándar es un proceso que conlleva tiempo y en el que intervienen muchas personas y organizaciones diferentes. En primer lugar, la utilización generalizada de un producto hace surgir consorcios y asociaciones de usuarios, que son las primeras organizaciones que tras un periodo de utilización, no ordenada, promueven la normalización mediante la elaboración de documentos técnicos cuyo objeto es sistematizar el uso del producto entre sus miembros. Estos documentos, con frecuencia de carácter interno, suelen denominarse especificaciones, y si bien no pueden considerarse estándares son frecuentemente el germen de un estándar posterior. Así, no suelen cubrir todo el espectro de usuarios, sino sólo aquello que atañe a los miembros del consorcio donde se han generado. Es importante reseñar que una especificación habitualmente está asociada a comités no acreditados para la publicación y difusión formal de estándares, tales como IETF (Internet Engineering Task Force), OMG (Object Management Group), o W3C (World Wide Web Consortium).

A partir de una o más especificaciones sobre el mismo producto, organizaciones de

certificación tales como AENOR, IEEE ,CEN o ISO, con el concurso de expertos en la materia, mejoran la especificación para cubrir las necesidades de todos los usuarios y fabricantes potenciales del producto. Como primer paso en su tarea, se elabora un borrador del estándar, que se somete a un proceso de refinamiento gradual a lo largo del cual se van publicando sucesivos borradores cada vez más conformados. Cuando se obtiene un borrador lo suficientemente maduro se transforma en una propuesta de estándar y se remite para su aceptación a una entidad de certificación (que puede ser la misma que gestionó la elaboración de los borradores). Si la propuesta es aceptada se reconoce formalmente como estándar, se publica de manera oficial y se promueve su difusión y adopción. La siguiente tabla resume las diferencias entre estándares y especificaciones:

Especificaciones	**Estándares**
Capturan el consenso aproximado	Capturan la aceptación general
Evolucionan rápidamente	Evolucionan lentamente
Facilitan	Regulan
Gestionan los riesgos a corto plazo	Gestionan los riesgos a largo plazo
Experimentales	Conclusivos

Tabla 2.1: Diferencias entre especificación y estándar

En el caso del e-learning, la necesidad de estandarización aparece como consecuencia tanto de la disponibilidad de un mayor número de materiales educativos en formato digital, como del desarrollo de un mercado real para plataformas de gestión del aprendizaje y contenidos formativos. El gran avance que el e-learning supone con respecto al concepto previo de enseñanza basada en cursos, y el agotamiento de los tradicionales cursos presenciales de coste habitualmente elevado, han promovido la aparición de este nuevo enfoque, frecuentemente basado en la fragmentación de los recursos educativos en los denominados “objetos de aprendizaje”.

2.4.1. Beneficios del modelo de los objetos de aprendizaje

La fragmentación de los recursos educativos en objetos de aprendizaje es un importante avance con respecto al concepto previo de enseñanza basada en cursos. El nuevo enfoque aporta, según Longmire, beneficios innegables como la flexibilidad, la facilidad para realizar actualizaciones, búsquedas y gestión de los contenidos, la personalización, la interoperabilidad, la facilidad para dirigir el aprendizaje a unos objetivos concretos y el incremento en el valor de los contenidos desarrollados según este modelo [Lon00].

Obviamente, la sola definición del concepto de objeto de aprendizaje no es suficiente para alcanzar todos estos beneficios. Resultan necesarias una bases mínimas de interoperabilidad y compatibilidad que permitan que componentes desarrollados por distintas entidades puedan intercambiar información y ser utilizados conjuntamente sin necesidad de introducir modificaciones. En este sentido, los estándares técnicos en el ámbito del *e-learning* persiguen la interoperabilidad y compatibilidad entre componentes [Duv04], y aseguran la reusabilidad de los objetos de aprendizaje [SS02]. En particular, la interoperabilidad es considerada por algunos autores una precondición para la reusabilidad, pues supone un lenguaje común que permite la comunicación de sistemas distintos [BC01].

La existencia de estándares que definan particularidades como la estructura y contenido de los metadatos, la forma de empaquetar los objetos de aprendizaje o la secuenciación de los contenidos resulta pues esencial para el desarrollo con éxito de los sistemas de *e-learning*. Los beneficios derivados de la estandarización han sido descritos en varios trabajos y coloquialmente se conocen como *-ilities*[7]

[7]En inglés dichos términos coinciden en el sufijo "ility": *accessibility*, *affordability*, *durability*, *extensibility*, *discoverability*, *interoperability*, *manageability* y *reusability*

[Sin00, HRJ02, SG03]:

- Accesibilidad del contenido, que estará disponible en cualquier momento y desde cualquier lugar.
- Interoperabilidad, entendida como la capacidad de que componentes desarrollados por distintas entidades puedan intercambiar información y ser utilizados conjuntamente.
- Reusabilidad de los contenidos como forma de economizar esfuerzos a la hora de crear nuevos contenidos educativos.
- Extensibilidad, o capacidad de ampliación, gracias a la construcción modularizada de contenidos.
- Facilidad de localización de los contenidos almacenados en repositorios que utilizan metadatos como forma de catalogación.
- Coste razonable, pues la estandarización reduce los costes de desarrollo.
- Facilidad de gestión de contenidos, pues el diseño en pequeñas unidades modulares facilita los cambios y actualizaciones.
- Perdurabilidad, pues el desarrollo de contenidos estándar evita la obsolescencia de los mismos ante cambios en las plataformas.

Además de lo anterior, la estandarización fomenta la comunicación y el intercambio, lo que permite que las organizaciones que generan contenidos obtengan rendimientos adicionales sobre sus inversiones. Finalmente, potencia el desarrollo de herramientas para la creación y gestión de contenidos estandarizados.

En la actualidad, diversos organismos participan activamente en el desarrollo de propuestas de estandarización. Se trata de un proceso complejo, debido a los muchos aspectos a considerar dentro de un proceso general de *e-learning*. Existen en la literatura sobre el tema dos revisiones de referencia, la realizada por Anido en 2002 [AFC+02] y la de Duval en 2004 [Duv04]; todas abordan la mayoría de los esfuerzos realizados o en curso. No obstante, el trabajo más exhaustivo y actualizado al respecto es el del observatorio sobre estándares de tecnologías educativas del CEN[8]. Utilizando como fuente de datos los elementos anteriormente citados, se muestra a continuación una breve reseña sobre las iniciativas más relevantes en el ámbito de la estandarización de los objetos de aprendizaje.

2.4.2. Metadatos

En lo referente a los metadatos específicamente orientados a la descripción de objetos de aprendizaje, IEEE LOM —basado en Dublin Core y cuyos antecedentes se encuentran en una especificación anterior de Ariadne— es el de más amplia difusión. No obstante, existen otras propuestas, como GEM, una extensión de Dublin Core orientada a la descripción de materiales educativos desarrollada por el Departamento de Educación norteamericano, o EdNA, propuesta de carácter similar a la anterior en el ámbito de la educación en Australia. Por otra parte, existen interesantes aportaciones para estandarizar la búsqueda de información de metadatos, como el protocolo de recuperación de metadatos en repositorios de objetos de aprendizaje de la Open Archives Initiative[9] [LVNW03], o la especificación para la interoperabilidad de listas de recursos [Jac04], cuyo propósito es facilitar el intercambio de metadatos entre sistemas de cara a crear listas de recursos que permitan una mejor organización y

[8]Learning Technology Standards Observatory - http://www.cen-ltso.net
[9]The Open Archives Initiative - http://www.openarchives.org

localización de los mismos.

Acrónimo	**Propuesta**	**Responsable**	**Fecha**
DCMI	Dublin Core Metadata Iniciative	DC-Ed	21/10/02
CANCORE	Can Core Profile	CanMDI	14/02/02
EdNA MD	EdNA Metadata	EdNA	25/10/01
GEM MD	GEM Metadata	GEM	14/03/02
GEMSTONES	Gestalt Extensions to MSES	GESTALT	26/04/99
IMS MD	IMS Metadata	IMS	01/10/01
ARIADNE MD	Ariadne Metadata	ARIADNE	01/02/02
LOM	Learning Object Metadata	LTSC	12/06/02

Tabla 2.2: Trabajos relacionados con las especificaciones y estándares de metadatos orientados a la descripción de los objetos de aprendizaje

Para conocer con mejor detalle el desarrollo de esta investigación, a continuación se describen las especificaciones DCMI Dublin Core e IEEE LOM.

DCMI Dublin Core

Dublin Core es un estándar abierto de metadatos utilizado para describir de manera sencilla cualquier tipo de recurso. El estándar define 15 elementos establecidos por un grupo internacional interdisciplinar experto en metadatos. La gramática es sencilla, formada por “elementos” y “cualificadores”.

Los elementos Dublin Core poseen nombres descriptivos que pretenden transmitir un significado semántico a los mismos. Cada elemento es opcional y puede repetirse. Además, los elementos pueden aparecer en cualquier orden. Se clasifican en:

- Elementos relacionados principalmente con el contenido del recurso.
- Elementos relacionados principalmente con el recurso cuando es visto como una propiedad intelectual.

- Elementos relacionados principalmente con la instanciación del recurso.

En la tabla resumen 2.3 se especifican los elementos Dublin Core clasificados.

Contenido	**Propiedad Intelectual**	**Instanciación**
Title	Creator	Date
Subject	Publisher	Type
Description	Contributor	Format
Source	Rights	Identifier
Language		
Relation		
Coverage		

Tabla 2.3: Elementos Dublin Core

En Agosto de 1999, el DCAC (Dublin Core Advisory Committee) fundó el Grupo de Trabajo de Educación de Dublin Core con el fin de desarrollar y realizar una propuesta para la utilización de los metadatos Dublin Core en la descripción de recursos educativos. Básicamente, su tarea es proponer ampliaciones al conjunto Dublin Core para describir este tipo particular de recursos, tomando LOM como base.

IEEE LOM

Propuesto por el comité de estándares de e-learning del IEEE [LTS02], establece un conjunto de atributos necesarios para localizar, administrar y evaluar un objeto de aprendizaje. La estructura básica del esquema de LOM se divide en nueve categorías:

1. **General:** categoría que agrupa la información general que describe a un objeto de aprendizaje. Engloba atributos básicos de localización y administración (identificador, título, lenguaje, descripción, palabras clave, contexto de aplicación, estructura, nivel de agregación).

2. **Lifecycle:** agrupa atributos relacionados con la evolución y el estado actual del objeto de aprendizaje. También describe quién ha realizado las ampliaciones o cambios en el objeto.

3. **Meta-Metadata:** grupo de atributos que describe la información relativa al propio registro de metadatos LOM, tales como autor de los metadatos, fecha de creación o formato, entre otros.

4. **Technical:** conjunto de atributos relativos a los requisitos y características técnicas necesarias.

5. **Educational:** grupo de atributos que describen las características pedagógicas del objeto de aprendizaje, tales como el tipo de interactividad (Interactivity Type), el tipo de recurso (Learning Resource Type) o la densidad semántica entre otros.

6. **Rights:** categoría que agrupa atributos relativos a los derechos de propiedad intelectual y licencia de uso del learning object.

7. **Relation:** grupo de atributos que describen las relaciones del objeto de aprendizaje con otros objetos de aprendizaje. Las relaciones están basadas en la especificación Dublin Core Qualifiers [DCM00]. La figura 2.1 muestra un esquema de los distintos tipos de relación posibles:

    - Relaciones de composición: permiten expresar la composición de un recurso compuesto.
    - Relaciones de formato: indican las relaciones de formatos de presentación de un recurso.
    - Relaciones de referencia: indican los recursos que dependen de forma indirecta en la comprensión y creación de un recurso.

- Relaciones de base: indican si un recurso está basado en otro.
- Relaciones de requerimiento: utilizadas para indicar las relaciones de necesidad entre distintos recursos.
- Relaciones de versión: utilizada para expresar relaciones de adaptación y evolución de versiones entre recursos.

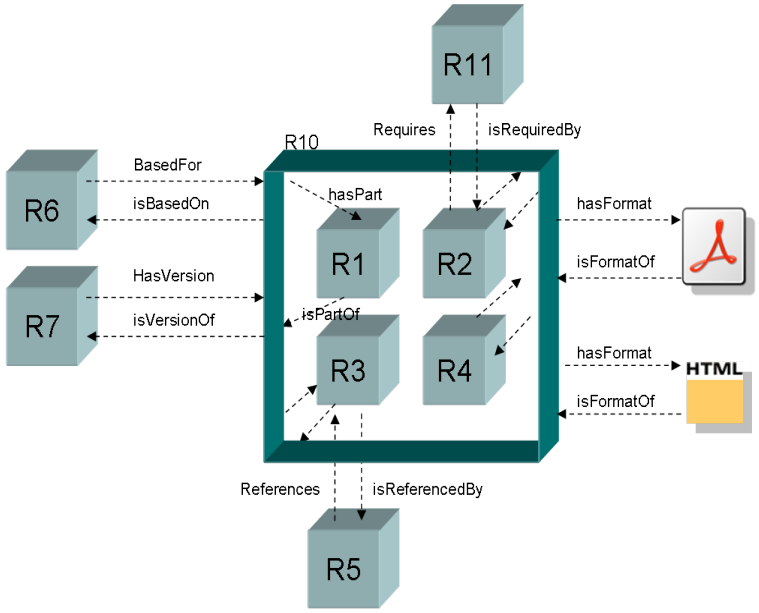

Figura 2.1: Tipos de relaciones LOM

8. **Annotation:** proporciona un conjunto de atributos que aportan comentarios sobre el uso del objeto de aprendizaje, además de quién y cuándo lo ha comentado.

9. **Classification:** describe el objeto de aprendizaje en relación a un sistema particular de clasificación.

2.4.3. Empaquetado y organización de recursos educativos

Un factor clave en el proceso de intercambio de agregaciones de recursos educativos entre diferentes sistemas es la preservación de las relaciones existentes entre las distintas unidades que componen la agregación. Así, es indispensable la definición de modelos de datos que permitan la representación de la estructura de las agregaciones de recursos educativos, con el fin de facilitar el intercambio de cursos completos o partes de los mismos. La recomendación más destacada en este campo es la propuesta por el consorcio IMS: la especificación IMS Content Packaging, cuyo elemento clave es el paquete.

Un paquete representa una agregación de recursos educativos que es tratado como una entidad única. Esta agregación puede incluir un curso independiente, una o varias partes de un curso, o incluso una colección de cursos. Además del consorcio IMS, otras instituciones han estado trabajando en este campo. Por ejemplo, la primera tarea de la iniciativa ADL en este área fue la adaptación del formato para la definición de estructuras de cursos desarrollado por el AICC a XML. La última versión oficial del modelo de referencia SCORM [ADL03](descrito al final de la sección) ha adoptado la propuesta IMS. Así, el SCORM Content Aggregation Model (CAM) incluye una versión extendida del IMS Packaging Model, incorporando, entre otras aportaciones menos significativas, la posibilidad de definir "prerrequisitos de acceso". Los prerrequisitos soportan la definición de comportamientos dinámicos sencillos en las organizaciones de recursos, por medio del establecimiento de un conjunto de condiciones de acceso a cada "item"dependiendo del estado del alumno en los otros "ítems"de la agregación. De este modo, SCORM proporciona capacidades sencillas de secuenciado y navegación condicional. Sin embargo, estas capacidades son muy limitadas. El consorcio IMS ha publicado un modelo de secuenciado más versátil denominado IMS Simple Sequencing Specification. Esta especificación define un método

para representar el comportamiento pretendido por el creador en una sesión de aprendizaje, es decir, la secuencia de entrega de objetos educativos al alumno. El proceso general de secuenciado simple se describe como una combinación de varios modelos de comportamiento: navegación, salida, acomodación, selección y aleatorización, secuenciado y entrega.

Acrónimo	**Propuesta**	**Responsable**	**Fecha**
CMI	CMI Guidelines for Interoperability	AICC	02/04/01
IMS CP	IMS Content Packaging	IMS	01/07/03
SCORM-CAM	Content Aggregation Model	ADL	01/10/01
AICC-CS	AICC Course Structure	AICC	02/04/01
AICC-P	AICC Packaging	AICC	19/09/06
IMS SS	IMS Simple Sequencing	IMS	20/03/03
IEEE P1484.6	IEEE Course Sequencing	IEEE	20/03/03

Tabla 2.4: Trabajos relacionados con las especificaciones y estándares de metadatos para el empaquetado y organización de recursos educativos

IMS Content Packaging Specification

Los paquetes IMS se componen de dos elementos. El primero es el "manifiesto", un documento XML en el que se describen los contenidos encapsulados y su organización. El segundo son los propios contenidos educativos, descritos en el manifiesto, tales como páginas Web, ficheros de texto, objetos de evaluación o cualquier otro tipo de material de datos. Cuando estos elementos se encapsulan en un fichero único (e.j. un fichero comprimido .zip, .jar, o .cab), el archivo resultante se denomina *fichero de intercambio de paquete*, figura 2.2.

Dentro del manifiesto existe el sub-elemento "metadatos" utilizado para describir en general el contenido empaquetado bajo un esquema de metadatos de objetos de

aprendizaje. Ejemplos válidos pueden ser IEEE LOM, IMS-MD, LTSN, UKCMF o algún otro perfil de aplicación de los citados. Gracias a la descripción realizada bajo estos esquemas, el objeto de aprendizaje empaquetado puede ser localizado por sistemas de búsqueda externos. Por ejemplo, un objeto descrito bajo el perfil de aplicación LOM-LTSN [FT01] puede ser identificado y clasificado dentro de un árbol jerárquico gracias a la categoría "classification" del esquema de metadatos utilizado, facilitando así la localización y clasificación de contenidos para su posterior búsqueda y navegación.

Otro de los componentes fundamentales del manifiesto es el sub-elemento "organizaciones". Este elemento se utiliza para especificar una o varias organizaciones alternativas para los recursos incluidos en el paquete. Cada organización define las relaciones estáticas existentes entre los recursos de la agregación como un árbol jerárquico, tal como se representa en la figura 2.2, donde cada elemento (item) se corresponde bien con un recurso educativo o bien con una agregación de elementos (items) de menor nivel (como por ejemplo la rama "Resolución de Problemas" de la figura 2.2).

Además, en el sub-elemento "recursos" dentro del manifiesto, se incluyen las referencias a todos los recursos utilizados. La especificación define dos tipos posibles de referencias: locales y globales. Las referencias locales son utilizadas para localizar únicamente los recursos almacenados dentro del paquete. La localización es posible gracias al uso de identificadores internos (e.j. RESOURCE-10221). Por otro lado, las referencias globales permiten el uso de recursos externos al paquete. En este tipo de referencia, el direccionamiento es posible gracias al uso de URL's. Cada recurso en IMS Content Packaging (IMS-CP) puede contener una o varias referencias a varios ficheros. En el caso de una página Web, si en ésta existen varias imágenes, en el recurso aparecerán las referencias al fichero html y a las imágenes contenidas.

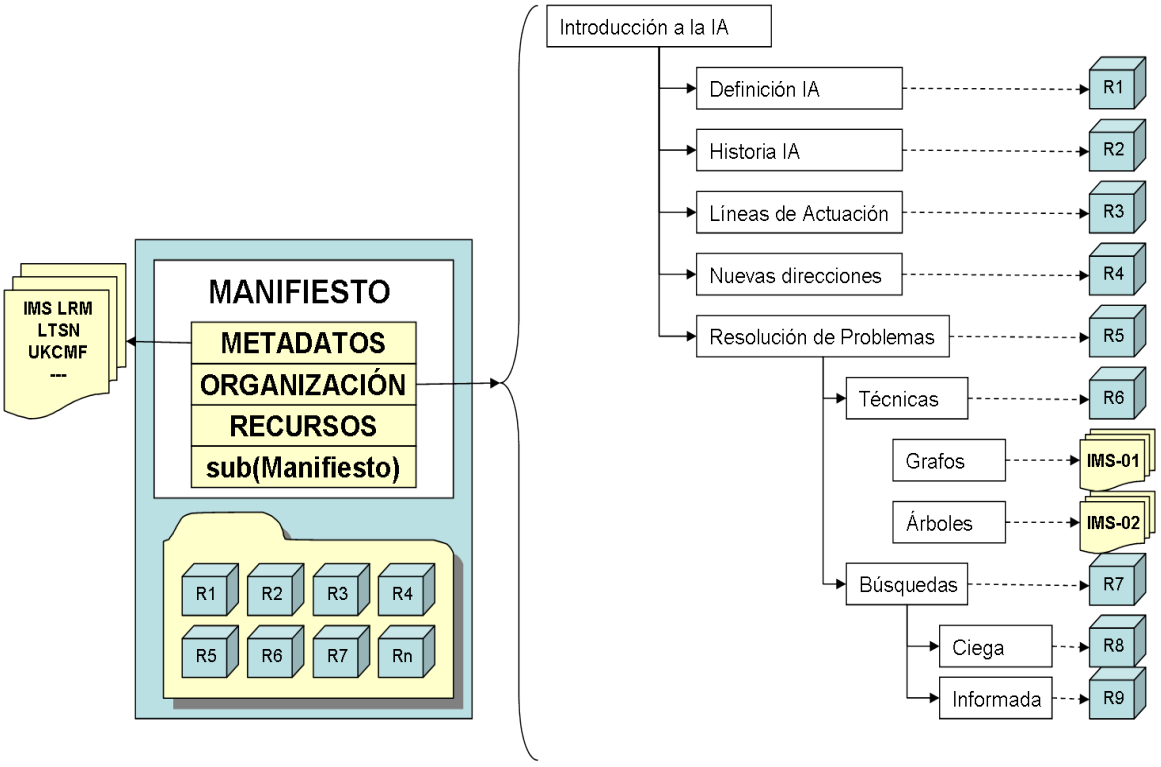

Figura 2.2: Estructura del fichero de intercambio de paquete IMS

IMS-CP posee una amplia aceptación gracias a la inclusión en la especificación agrupada SCORM-CP [ADL03], así en los sistemas de ejecución de contenidos de aprendizaje conformes con esta especificación, tales como Moodle o Atutor, es posible el intercambio de cursos. También algunas herramientas de construcción de metadatos, como RELOAD[10] o la herramienta "Authorware" de ADOBE[11], utilizan los esquemas de IMS-CP para la construcción del manifiesto del paquete didáctico.

[10]Reusable eLearning Object Authoring and Delivering (RELOAD) - http://www.reload.ac.uk/
[11]Authorware - http://www.adobe.com/products/authorware/

IMS Simple Sequencing

El modelo simple de secuenciación de actividades de aprendizaje, aporta un esquema base utilizado para especificar la navegación entre las distintas actividades de aprendizaje existentes en un objeto de aprendizaje. Este esquema permite crear un lenguaje lógico (escrito en XML), capaz de ser procesado por un LMS con la finalidad de habilitar los mecanismos de ejecución y seguimiento sobre el grado de compleción de las actividades definidas por el diseñador del objeto de aprendizaje.

La composición organizada de un objeto de aprendizaje ya ha sido expuesta en las secciones anteriores. De modo general toda actividad definida en el elemento "organización" de un manifiesto (tanto IMS como SCORM) es secuenciada dentro de un entorno de ejecución.

El orden básico parte del árbol de actividades, basado en la estructura base de organización del objeto de aprendizaje. En el árbol se especifica la composición de las actividades. Los nodos padre definen la agrupación de una serie de actividades (sus sucesores inmediatos). La forma más simple de secuenciado es la composición por recorrido pre-orden. Este recorrido puede ser modificado si se introducen en las agrupaciones determinadas reglas de secuenciado creadas por el diseñador del objeto de aprendizaje. A partir de este árbol de actividades es posible establecer la información de secuenciado.

En el árbol de actividades de la figura 2.3 se muestra un ejemplo simple de secuenciación. El orden del árbol es expresado con las conexiones en negro, el orden de secuencia en rojo y finalmente el orden de consecución de los objetivos en verde. Como se puede apreciar, en cada actividad elemental (nodo hoja, como por ejemplo la actividad A1.2) podemos especificar el objetivo asociado al terminarla, o una serie

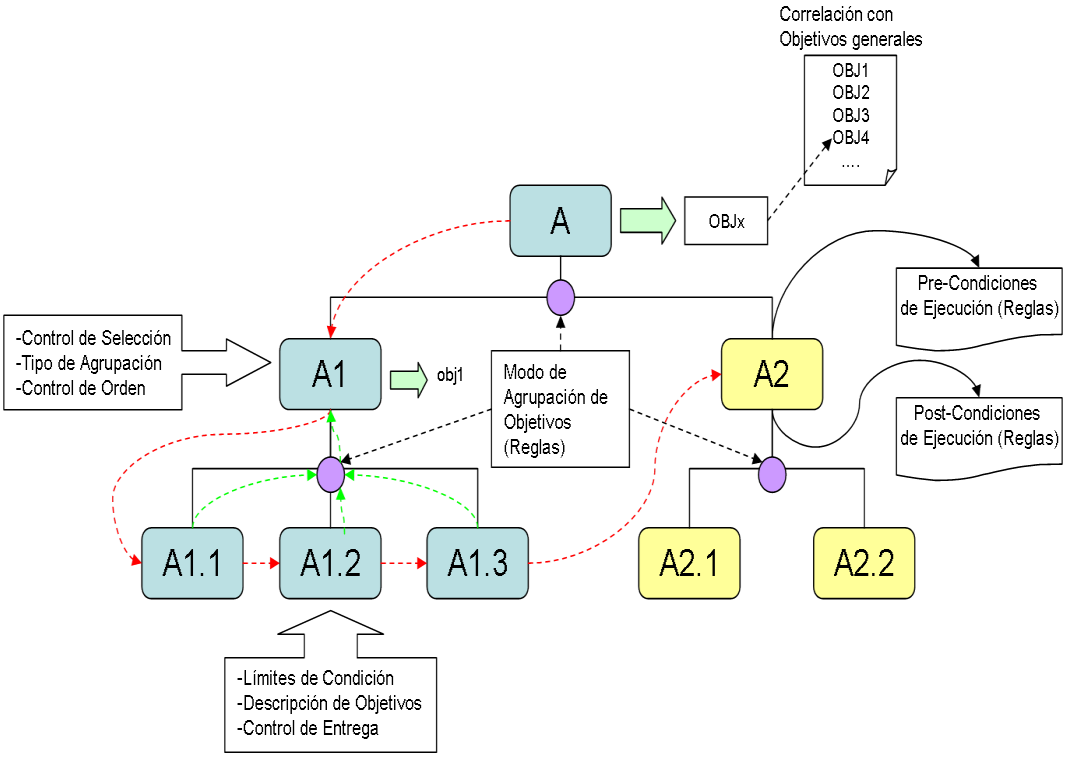

Figura 2.3: Árbol de Actividades

de condiciones de ejecución (número de intentos, tiempo estricto necesario o el comienzo de la tarea). Cada actividad será realizada por un usuario, que modificará el estado de la actividad dependiendo del progreso realizado. En la raíz de un grupo de actividades (raíz del cluster o cualquier nodo padre, como por ejemplo la actividad A2), puede especificarse el modo de navegación y el modo de selección existente entre las actividades hijas, así como las reglas de agrupación de los objetivos conseguidos en las tareas hijas para la consecución de un objetivo de mayor nivel [12]. Los objetivos pueden ser correlacionados con otros más generales utilizados en una definición más

[12]En la figura 2.3, pueden elegirse diferentes modos de agrupación: por media de objetivo superior a un umbral predefinido, únicamente por completar una de las tareas hijas o la necesidad de todas, entre otras posibles especificadas en el estándar

genérica para otros objetos de aprendizaje[13]. A la hora de ejecutar una actividad se analizan las precondiciones para ver si es necesario incluir la tarea en el flujo de ejecución o no, así como los efectos producidos al completarse la ejecución.

2.4.4. Contenido

En lo referente al contenido de los materiales educativos, es importante referenciar los denominados EML —Educational Modelling Languages— [RvRK+02], notaciones semánticas para diseñar unidades educativas. Uno de los más relevantes y maduros es el de la Universidad Abierta de Holanda [Kop01], base de la especificación para el diseño de materiales educativos de IMS.

Otro esfuerzo significativo lo constituye la especificación para la interoperabilidad de tests y cuestionarios [Lay04], que describe la estructura básica de los citados elementos de evaluación, principalmente para facilitar el intercambio de cuestionarios y evaluaciones entre distintos LMS.

IMS Learning Design

Los diseños para el aprendizaje son un tipo concreto de objetos para el aprendizaje en los cuales se determina una secuencia y definición de actividades para un propósito educativo concreto [KOA03a]. Un diseño para el aprendizaje tendrá por tanto que tener algunos elementos esenciales básicos tales como:

- Objetivos pedagógicos que se pretenden cumplir.

[13]La importancia de la definición de unos objetivos generales radica en la posibilidad de crear sistemas capaces de ofrecer contenidos de aprendizaje en función de los perfiles deseados, de tal forma que un sistema LMS pueda generar una secuenciación correcta en función del perfil del cliente. Según su curriculum puede haber completado algunos objetivos que no son necesarios re-aprender de nuevo. Por ello es necesario una visión más global a la hora de definir estos mismos.

Acrónimo	**Propuesta**	**Responsable**	**Fecha**
OUNL-EML	Educational Modeling Language	OUNL	01/06/01
IMS LD	Learning Design	IMS	20/01/03
PALO	PALO Language	UNED	01/01/02
CDF	Course Description Language	ARIADNE	No-published
IMS QTI & IMS Question	Test Interoperability	IMS	26/03/03

Tabla 2.5: Trabajos relacionados con las especificaciones y estándares de metadatos para la descripción del contenido de los materiales educativos

- Secuencia de actividades y subactividades.
- Recursos a utilizar en cada una de las (sub-)actividades.
- Perfil y rol de los participantes en las (sub-)actividades.

El diseño pedagógico es un problema de racionalidad abierta, en el que las opciones no están predeterminadas o conforman un espacio determinado. Por ejemplo, si se quiere que unos estudiantes aprendan a programar por primera vez, podrían darse múltiples alternativas: actividades o ejercicios individuales, guiados por el profesor, por parejas, en grupos más grandes, etc. Y también la secuencia de las actividades admite múltiples variantes. Dado que en el ámbito de la programación de ordenadores el estudio de los vectores y de las cadenas de caracteres está estrechamente relacionado, ¿es mejor primero estudiar los vectores y luego las cadenas, o al contrario?. Todas estas decisiones se pueden tomar intuitivamente o pueden basarse bien en principios pedagógicos generales, o en conocimiento pedagógico existente. En cualquier caso, es importante resaltar que la descripción del diseño proporciona información útil para los entornos de ejecución de los sistemas gestores del aprendizaje. Por ello existen especificaciones de lenguajes que permiten describir los elementos esenciales del diseño (anteriormente enunciadas, ver tabla 2.5). Un caso concreto aceptado en múltiples plataformas de aprendizaje es IMS LD.

IMS LD [KOA03b] permite modelar diseños para el aprendizaje. El lenguaje o modelo que proporciona para los mismos se muestra en la figura 2.4, que ha sido extraída de la propia especificación. En la figura se pueden apreciar los siguientes elementos fundamentales:

- La estructura de actividades: un método es un diseño para el aprendizaje, y este se estructura en piezas (plays en inglés) y representaciones (acts, en inglés) tomando en ambos casos el sentido teatral de estos términos. Las piezas y las representaciones son por tanto sub-actividades, con la característica de que las piezas son actividades que pueden ser simultáneas (concurrentes), mientras que las representaciones son actividades en secuencia. Finalmente las actividades forman la subestructura de las piezas, pero con una estructura según partes de un papel (denominadas role-parts en inglés).

- Los objetivos (y también los prerrequisitos de los mismos) se asocian al método completo.

- Los roles (que se asociarán a personas concretas cuando se esté realizando la actividad) permiten una definición de propiedades para los mismos. El concepto de "parte de un papel" representa "la participación de un rol en una actividad", de modo que para la misma actividad, diferentes roles pueden tener participaciones diferenciadas.

- Los recursos se representan como objetos de aprendizaje y servicios (estos últimos representan cualquier servicio, tales como un chat o un foro) y se asocian a las actividades mediante un entorno.

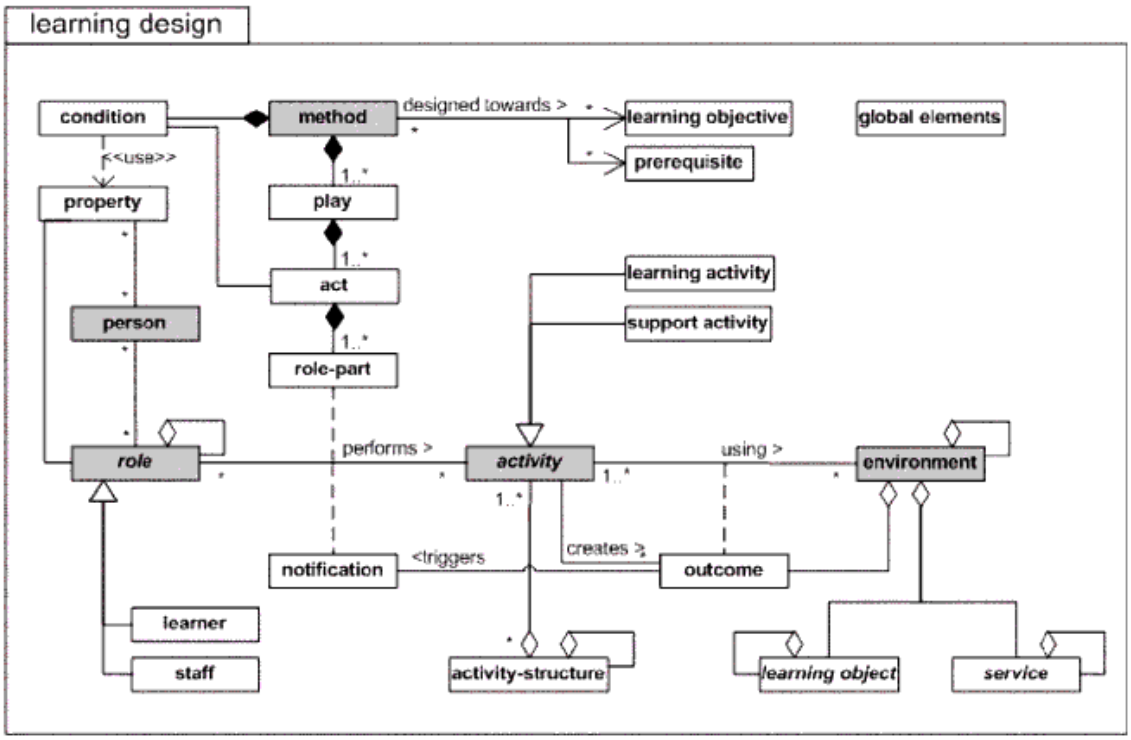

Figura 2.4: IMS Learning Desing - relaciones entre clases

Es muy importante resaltar que además de los beneficios derivados de tener un modelo común de descripción de actividades, IMS LD permite que se registre la interacción durante el aprendizaje con sistemas que "entiendan" IMS LD. Esto abre nuevas oportunidades al estudio de los resultados de diferentes estrategias pedagógicas (Sicilia, 2006), y permitirá eventualmente la identificación de patrones de interacción de los aprendices (Koper, 2004).

Ante esta amplia descripción, numerosas instituciones y grupos de desarrolladores de software libre han puesto a disposición de la comunidad de usuarios herramientas que permiten manejar este amplio conjunto de metadatos. Entre los ejemplos más

destacados podemos nombrar: RELOAD IMS-LD[14] Editor o LAMS[15].

IMS Question and Test Interoperability

La especificación IMS-QTI está enfocada a la interoperabilidad de los cuestionarios, evaluaciones y preguntas almacenados en los sistemas LMS. Contempla una estructura básica que describe la forma de representar preguntas individuales o elementos (assesment item) y gestionar evaluaciones o exámenes completos (assessment). La especificación facilita el acceso de sistemas electrónicos de enseñanza a este tipo de almacenes de datos útiles y reutilizables en otros sistemas o escenarios de enseñanza.

Por otro lado, la especificación propone un sistema coherente para que los sistemas puedan informar de cuál es el resultado de una evaluación, además de normalizar el modo en el que almacenan y estructuran los cuestionarios creados por diferentes herramientas de autor. Esto permite, por ejemplo, el uso de las mismas preguntas en diversos LMS o en sistemas de evaluación electrónica, o la integración en un único LMS de preguntas o exámenes desarrollados con distintas herramientas.

El estándar QTI está ampliamente extendido entre las herramientas de autor. Ejemplos notables son RESPONDUS[16], AlfaNET QTI Tools[17] y Moodle.

2.4.5. Manejo e intercambio de información entre sistemas LMS

En lo correspondiente a la especificación, manejo e intercambio de información asociada a los alumnos, existen interesantes trabajos en curso. *Public and Private Information for Learners* [Far01], pretende la especificación de registros portables

[14]Reload Editor, Web - http://www.reload.ac.uk/
[15]LAMS International, Web - http://www.lamsinternational.com/
[16]Herramienta de Autor IMS-QTI RESPONDUS: http://www.respondus.com/
[17]Herramientas IMS-QTI AlfaNET: http://rtd.softwareag.es/alfanetqtitools/

para el alumno, simplificando así el intercambio de datos entre sistemas cooperantes. Esta especificación fue desarrollada inicialmente por LTSC, pero a finales de 2001 fue recogida por ISO/IEC JTC1 SC36[18]. Otro esfuerzo importante es el *Learning Information Package* de IMS [STR01], una especificación orientada al intercambio de información sobre alumnos entre sistemas basados en Internet.

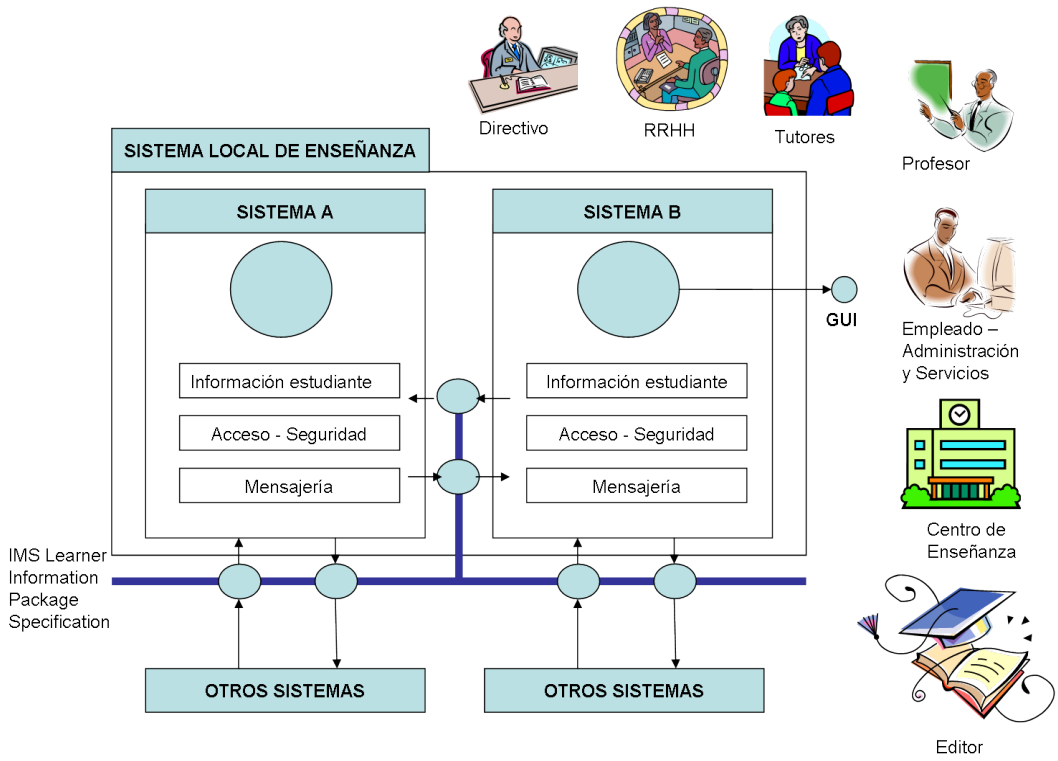

Figura 2.5: IMS LIP: Representación de componentes de un sistema de e-learning

2.4.6. Competencias

En lo que se refiere a la definición y estandarización de competencias existen dos esfuerzos principales. *Competency Definitions* de LTSC [Arc04] es un trabajo en

[18]Information Technology for Learning, Education, and Training - http://jtc1sc36.org

Acrónimo	Propuesta	Responsable	Fecha
IMS-LIP	Learner Information Package	IMS	17/02/05
IMS-EIM	Enterprise Information Model	IMS	01/05/02
IMS-ES	Enterprise Services	IMS	11/06/04
SC36/WG3	ISO/IEC JTC1 SC36 WG3	ISO	No publicado

Tabla 2.6: Trabajos relacionados con las especificaciones y estándares de metadatos orientadas al manejo e intercambio de información entre sistemas LMS

curso cuyo objetivo es definir un modelo de datos para las definiciones de competencia, creando identificadores únicos para los llamados "objetivos didácticos" que faciliten la referencia no ambigua desde los recursos didácticos. Un trabajo similar es IMS-RDCEO —Reusable Definition of Competency or Educational Objective— que persigue la definición de competencias que aparecen como prerrequisitos para la realización de experiencias educativas o como objetivo didáctico de las mismas [CO02].

Acrónimo	Propuesta	Responsable	Fecha
LTSC-CD	Competency Definitions	LTSC	17/02/05
IMS-RDCEO	Reusable Definition of Competency or Educational Objective	IMS	25/10/02
CEN	European Model for Learner Competencies	CEN	No-published

Tabla 2.7: Trabajos relacionados con las especificaciones y estándares de metadatos orientadas a la definición de competencias

2.4.7. Interacción entre sistemas

En cuanto al soporte para la interacción entre los contenidos en forma de objetos de aprendizaje y los sistemas que los manejan, existen tres trabajos importantes. El modelo estándar de datos para la comunicación de contenidos de LTSC [Lew04],

describe la información que se puede transmitir desde y hacia un objeto de aprendizaje por parte de los LMS cuando un usuario está interactuando con el objeto de aprendizaje, bien para transmitir contenidos sobre las puntuaciones del alumno,etc. o bien para recabar información sobre el alumno que el objeto necesita para funcionar. Un esfuerzo similar aparece en la especificación *CMI Guidelines for interoperability* de AICC [Hyd01], donde se definen los procedimientos y responsabilidades del mecanismo de comunicación entre contenidos y LMS. Por último, la normalización de la arquitectura de los sistemas de *e-learning* está reflejada en el borrador de estándar LTSA [FT01], que especifica a alto nivel una arquitectura en cinco capas (aunque sólo la capa 3, componentes del sistema, es normativa) junto con el diseño de los sistemas y los componentes de los mismos.

Acrónimo	**Propuesta**	**Responsable**	**Fecha**
LTSC-COC	Content Object Communication	LTSC	21/03/2002
CMI-GI	Guidelines for interoperability, versión 4	AICC	16/08/2004
LTSC-LTSA	Learning Technology System Architecture	LTSC	30/11/2001

Tabla 2.8: Trabajos relacionados con las especificaciones y estándares de metadatos orientadas a la interoperabilidad de contenidos

Respecto a la interoperabilidad entre los repositorios de objetos de aprendizaje existen tres trabajos importantes:

- IMS-DRI proporciona las recomendaciones para la interoperabilidad de las funciones más comunes de los repositorios. En la especificación se propone el uso de estándares y tecnologías ya existentes aprovechando las características aportadas por cada uno. Así por ejemplo en las búsquedas se recomienda XPath

[W3C99], o en los métodos de agregación y recogida de resultados el modelo OAI (Open Archive Iniciative) [LVNW03].

- Otro esfuerzo considerable a tener en cuenta es el de la iniciativa de interoperabilidad del CEN/ISSS apoyada por PROLEARN, cuyo objetivo es el de proporcionar una especificación de una interfaz simple de acceso — SQI (Simple Query Interface [SQI05])— . El proyecto europeo ELENA[19] es un claro ejemplo de uso de está especificación, en el que se crea una red de repositorios de información útil para el aprendizaje (tales como Amazon, LASON, Clix , HCD Online y SeminarShop.at entre otros) conectados a través de esta interfaz. SQI es parte de LORI (Learning Object Repository Interface). LORI [LOR05] es una arquitectura de integración por capas que define los servicios necesarios para conseguir la interoperabilidad entre repositorios, como por ejemplo servicios de autentificación, administración de la sesión y servicios de aplicación (consultas o petición de contenidos).

- Por último, uno de los esfuerzos más importantes desarrollados en los últimos años por por la iniciativa ADL, la organización para las iniciativas de investigación nacional CNRI y el laboratorio de arquitectura de sistemas de aprendizaje LSAL. CORDRA (Content Object Repository Discovery and Registration/Resolution Architecture) [20] es un modelo de arquitectura distribuida abierta para repositorios. En las especificaciones de este modelo se establecen una serie de servicios para el descubrimiento, compartición y reutilización de recursos con contenido didácticos. CORDRA se complementa con SCORM aportando las especificaciones de federación entre distintos repositorios utilizando diversos agentes software coordinados (CORDRA Registry, CORDRA Catalog, Federation Repository y Content Repository).

[19]Proyecto de investigación de la IST-European Commision: http://www.elena-project.org/
[20]CORDRA - http://cordra.net/docs/

Acrónimo	Propuesta	Responsable	Fecha
IMS-DRI	Digital Repository Interface	IMS	30/01/2003
SQI	Simple Query Interface	CEN/ISSS & PROLEARN Network	20/04/2005
CORDRA	Content Object Repository Discovery and Registration/Resolution Architecture	ADL, CNRI & LSAL	29/12/2004

Tabla 2.9: Trabajos relacionados con las especificaciones y estándares de metadatos orientadas a la interoperabilidad de repositorios

IMS Digital Repositories Specification

La especificación Digital Repositories [IMS03b] tiene como objetivo la elaboración de recomendaciones que permitan la interoperabilidad entre diferentes repositorios digitales. El propósito es poder acceder a cualquier almacén de recursos educativos para obtenerlos sin necesidad de conocer cuál es la organización o estructura de dicho almacén. En esta recuperación, los metadatos son el elemento principal para la identificación de los recursos.

En este esquema (figura 2.6) se pueden observar las tres entidades definidas por esta especificación para abordar el problema (marcadas en verde, azul, y amarillo):

- Roles: Learner, Creator, Infoseeker y Agent.
- Componentes para la administración de los recursos.
- Servicios.

Las líneas rojas indican las interacciones entre los principales componentes de la arquitectura.

A continuación se exponen los diferentes roles definidos en la especificación:

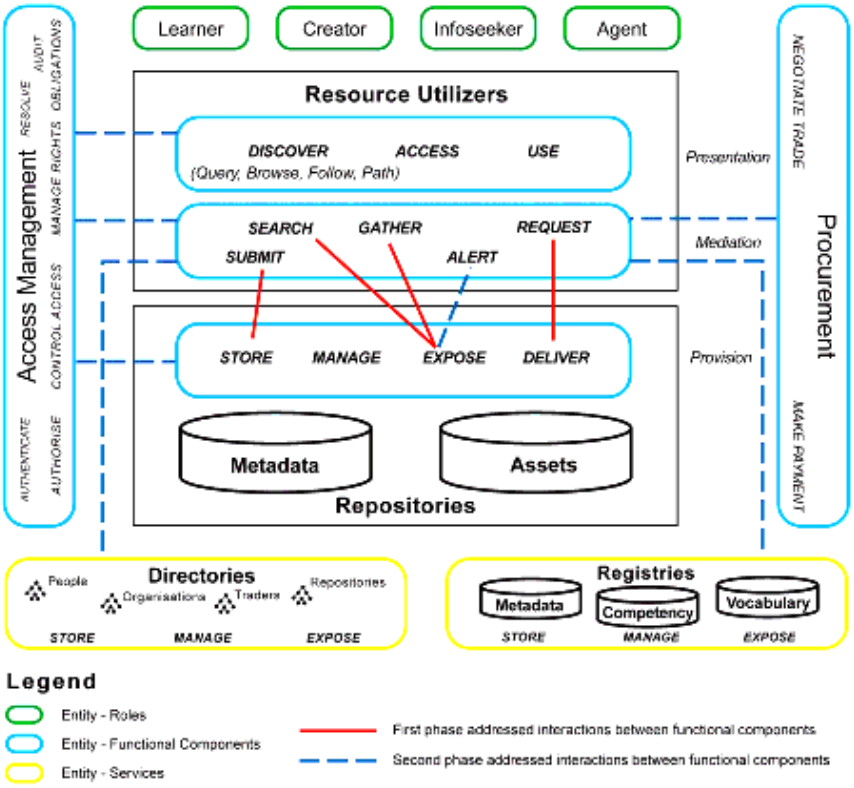

Figura 2.6: IMSDRI: Esquema de componentes y relaciones

- Learner (alumno). Se define como una persona que sigue un curso o que está dentro de un proceso de aprendizaje. Generalmente un alumno usará los diferentes Servicios Web que un sistema de enseñanza pueda ofrecer y necesitará los diferentes recursos educativos para poder llevar a cabo su formación. Una vez que un alumno abandona la aplicación de enseñanza, su rol cambiará al de Infoseeker (buscador de información).

- Creator (creador). Es una persona encargada de la creación de los objetos educativos, de los cursos, del secuenciamiento de aprendizaje, o de los programas

de las asignaturas.

- Infoseeker (buscador de información). Este rol lo desempeñan aquellas personas que buscan información en los repositorios a través de los diferentes servicios de búsqueda. Tanto un alumno como un creador pasan a ser buscadores de información en el momento que buscan dentro de los diferentes recursos de aprendizaje que hay dentro de un almacén. Es importante tener en cuenta que aquellos que se engloban dentro de este rol no tienen porqué estar dentro del proceso de aprendizaje.

- Agent (agente). Es una aplicación inteligente capaz de emular el comportamiento de cualquiera de los otros roles. Una vez realizada su tarea, un agente procesará los resultados obtenidos bien de manera automática, bien mediante la intervención de alumnos, creadores o buscadores de información.

Gracias a esta especificación, se consigue acceder a los contenidos almacenados en repositorios desde sistemas de enseñanza (LMS), sistemas de gestión de contenidos educativos (LCMS), portales de búsqueda de contenidos, y desde cualquier agente software que sea compatible con IMS RDI.

La especificación introduce un componente intermedio que podría implementar las siguientes funciones para facilitar las búsquedas:

- Un traductor, que sea capaz de traducir una petición de búsqueda de un formato a otro para hacerla así comprensible a todos los repositorios.

- Un conversor, encargado de hacer que todos los metadatos incluidos en cada objeto de aprendizaje sean útiles para facilitar las búsquedas.

- Un encargado de pasar una petición de búsqueda a los diferentes almacenes y de gestionar las respuestas.

Las funcionalidades descritas en la especificación según las líneas rojas de la figura 2.6 se detallan a continuación:

Search/Expose: esta funcionalidad hace referencia a la búsqueda de metadatos asociados con los contenidos almacenados en los repositorios. Nos encontramos ante un dominio muy extenso y heterogéneo en cuanto a las diferentes tecnologías de almacenamiento que existen, se propone una capa intermedia encargada de mediar entre las peticiones y los diferentes formatos en los que se encuentran almacenados los metadatos.

Gather/Expose: esta funcionalidad proporciona la forma de escribir los meta datos que van a servir para las búsquedas, la forma de agruparlos para facilitar los sondeos futuros y la manera en que se tienen que agregar para formar nuevos repositorios. Esta funcionalidad interactúa con el repositorio de dos maneras diferentes. La primera consiste en solicitar metadatos del repositorio (pull), mientras que en la segunda ofrece al almacén metadatos para que sean almacenados (push).

Alert/Expose: la especificación RDI contempla esta funcionalidad como un posible componente de un repositorio digital o un servicio intermedio encargado de mandar correos electrónicos.

Submit/Store: esta funcionalidad hace referencia a la forma de almacenar un objeto en un almacén y la forma que tomará una vez almacenado para hacer posible su recuperación. El lugar desde el cual se coge el objeto para su almacenamiento puede ser otro repositorio, un sistema de enseñanza, el disco duro del desarrollador, o cualquier punto de la red. Bajo la especificación se incluye:

- Añadir al protocolo mecanismos de cifrado para asegurar la integridad de los contenidos.

- No ofrecer acceso directo al servidor mediante FTP por los problemas de seguridad que esto conlleva.
- Tener en cuenta que FTP no ofrece ningún mecanismo que asegure que un objeto ha sido copiado en su totalidad de un punto de la red al almacén.

Request/Deliver: La función "*Request*" es la petición de acceso a un recurso que realiza un usuario del sistema una vez lo ha localizado gracias a los metadatos que lleva asociados. "*Deliver*" se refiere a la respuesta que le da el repositorio, otorgando o negando el acceso al recurso.

Gracias al aporte de esta especificación es posible crear aplicaciones capaces de realizar consultas distribuidas entre distintos repositorios, y de ofrecer respuestas con mayor oferta de recursos. La propuesta de construcción del repositorio de esta tesis hace uso de esta especificación para enriquecer las búsquedas (en el capítulo 4 se explicará con mayor detalle).

2.4.8. SCORM

SCORM [ADL03] es un modelo común de objetos de aprendizaje basado en componentes, cuyo principal objetivo es permitir la compartición de contenidos educativos estándar entre diferentes sistemas *e-learning*. SCORM aspira a convertirse en un modelo final, único para el diseño y manejo de los objetos de aprendizaje hacia el que converjan el resto de iniciativas, y así, engloba varios estándares y especificaciones de entre las mencionadas anteriormente, como LOM, las especificaciones de IMS sobre secuenciación y diseño de contenidos, o el EML de la Universidad Abierta de Holanda (OUNL).

SCORM es una colección integrada y armonizada de las especificaciones y estándares descritos en diferentes ámbitos de aplicación dentro de la siguiente serie de libros

técnicos: “SCORM Content Aggregation Model”, “SCORM Run-Time Environment” y “SCORM Sequencing and Navigation”.

Modelo de agregación de contenidos

El objetivo del modelo de agregación de contenidos de SCORM es proveer un medio común de componer “contenidos docentes” desde diversas fuentes compartibles y reusables (SCO - Sharable Content Object). Define cómo un objeto de aprendizaje puede ser identificado, descrito y agregado dentro de un curso o una parte de un curso, y cómo puede ser compartido por diversos gestores de aprendizaje (LMS Learning Management Systems) o por diversos repositorios.

Cada SCO queda compuesto por múltiples recursos (páginas html, applets, flash, imágenes, texto plano, etc..)[21] y estructurado en función de un manifiesto de metadatos.

Entorno de ejecución

El objetivo del entorno operativo o de ejecución de SCORM es proporcionar un medio para la interoperatividad entre los objetos compartibles de contenidos, SCO, y los sistemas de gestión de aprendizaje (LMS). Un requisito de SCORM es que el contenido pueda intercambiarse entre múltiples LMS sin tener en cuenta las herramientas que usen para crear o utilizar los contenidos. Para que esto sea posible, debe existir un método común para ejecutar un contenido en la plataforma de usuario final, un método común para que los contenidos se comuniquen con el LMS y elementos de datos predefinidos que sean intercambiables entre el LMS y el contenido durante su ejecución.

[21] En el estándar SCORM aparecen como ASSETS.

Secuenciación y navegación

SCORM almacena la secuenciación de los contenidos didácticos apoyándose en diferentes estándares de navegación como (IMS Simple Sequencing) o el AICC CMI001 v3.5. Como ya se ha expresando anteriormente en la descripción de IMS-SS, la secuenciación permite indicar el modo de ejecución de los contenidos a lo largo de una actividad de enseñanza. Los motores de ejecución de SCORM obtienen la información de secuenciación del paquete a ejecutar, de esta forma el motor de ejecución sigue estrictamente la secuencia de contenidos establecida por el creador de contenidos.

2.5. Repositorios de objetos de aprendizaje

Los repositorios de objetos de aprendizaje describen los distintos recursos didácticos existentes en la Web, almacenando registros de metadatos asociados a los objetos descritos y garantizando una búsqueda mucho más estructurada del conocimiento existente. Creadores de objetos de aprendizaje y estudiantes pueden obtener las referencias a estos objetos como resultado de una búsqueda más detallada que hace uso de los metadatos almacenados en el repositorio. En los repositorios actuales, la mejora en los resultados sobre búsquedas específicas para objetos de aprendizaje no es la única ventaja. Un elemento importante que justifica por sí mismo la existencia de este tipo de repositorios, lo constituyen las revisiones que ciertos expertos realizan (y publican) a los objetos.

En la figura 2.7 se muestra el esquema básico de interacción entre los distintos actores de un repositorio de objetos de aprendizaje dentro de una red privada (intranet de empresa, universidad o centro educativo). Como se puede apreciar en este esquema, tanto el objeto de aprendizaje como sus metadatos son almacenados en el mismo repositorio. De igual forma funcionan los sistemas estructurados LCMS (Learning

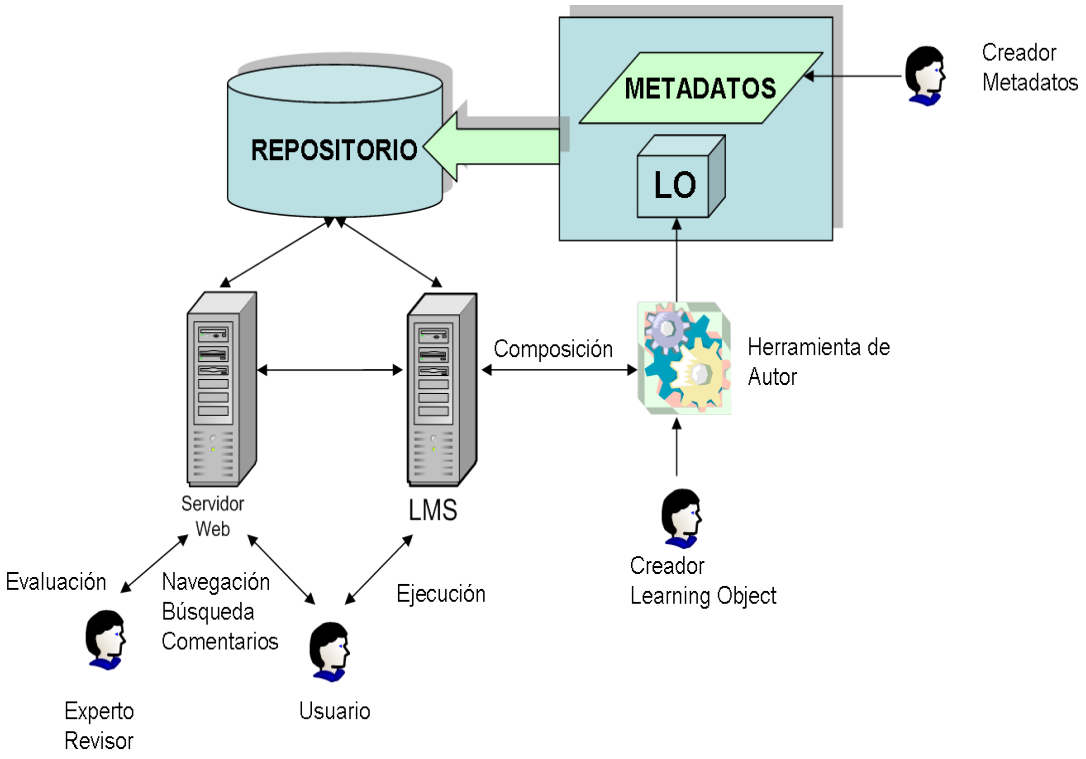

Figura 2.7: Esquema general de un repositorio de objetos de aprendizaje.

Content Management Systems), almacenando los recursos educativos empaquetados con sus metadatos, o bien en un único fichero (comprimido en zip según alguna especificación de almacenamiento tal como LCMS SCORM [ADL03]), o en ficheros separados: por un lado el paquete agrupado de recursos del objeto de aprendizaje y por otro el fichero descriptor de metadatos (en XML estructurado). El LCMS puede ser integrado en un sistema (Learning Management System) LMS, o los dos pueden ser conectados por una interfaz. En el esquema también se representa la interacción de un LMS con el repositorio de objetos (que puede ser administrado por un LCMS) y sus usuarios a través de un servidor web. La mayoría de las aplicaciones LMS presentan una interfaz web para la gestión y operación del contenido, de modo que la interacción con el LMS permite ejecutar las acciones enviadas por los usuarios del

repositorio. Hay que tener en cuenta que en este esquema de operación, el contenido del repositorio queda acotado dentro de la red corporativa.

Este tipo de sistemas dan soporte a toda la teoría de reutilización de objetos de aprendizaje, ya que es posible componer nuevos objetos utilizando recursos de otras personas ya existentes en el repositorio. La atomicidad de los objetos de aprendizaje siempre pasa por la unidad de recurso elemental utilizado en una actividad de enseñanza, así por ejemplo un texto, una imagen, una animación, un video o un applet que sirvan para enseñar son objetos de aprendizaje. Para entender mejor esta idea se expone como ejemplo un applet capaz de elaborar la representación gráfica de una función a partir de su expresión matemática. Este recurso puede ser utilizado para enseñar a generar e interpretar gráficas de funciones o contrastar determinados resultados de algunos ejercicios. Cualquier persona que necesite este recurso para crear un objeto de aprendizaje más complejo puede localizarlo en el repositorio de la universidad o institución y referenciarlo.

Si nos trasladamos al modelo de Internet, las posibilidades son infinitas pues existen miles de recursos didácticos sin ningún fin comercial, sólo educativo. Los nuevos repositorios de objetos de aprendizaje explotan la filosofía de compartición de recursos para conseguir un mayor grado de reutilización y para compensar la duplicidad innecesaria de recursos. Este nuevo escenario, representado en la figura 2.8, aporta una nueva concepción del ciclo de vida de los objetos de aprendizaje gracias a la intervención de diferentes roles independientes:

- **Creador del objeto de aprendizaje**: institución o persona encargada de crear el contenido mediante diferentes herramientas de autor.

- **Creador de metadatos**: organización o persona encargada de generar un conjunto de metadatos sobre el objeto de aprendizaje según un modelo y formato establecido.

- **Usuario final**: el usuario del repositorio que busca y usa el objeto de aprendizaje, puede ser un profesor o estudiante. Además puede realizar comentarios sobre las experiencias como usuario.

- **Evaluador**: experto en el dominio, responsable de realizar la evaluación del objeto de aprendizaje. Sus evaluaciones son una valiosa aportación al repositorio.

Con los comentarios y evaluaciones hechos por los profesionales del dominio, el creador del objeto de aprendizaje puede mejorar el contenido para incrementar el grado de efectividad requerido en sus objetivos. Este tipo de repositorios permiten por tanto conectar a los diferentes roles, creando así una nueva sinergia entre distintos profesionales con el objetivo de mejorar de la calidad en la enseñanza y educación.

Es importante resaltar que para el propósito actual de esta tesis, y si bien no es el caso del 100 % de los repositorios actuales, se considerará que los repositorios de objetos de aprendizaje almacenan metadatos y no el contenido de los objetos. Supondremos que los recursos del objeto estarán localizados en otro repositorio especializado en el almacenamiento de la información más comúnmente utilizada dentro del objeto de aprendizaje (animaciones flash, applets, páginas html, imágenes, etcétera).

Los repositorios de este tipo, entre los que se destaca CAREO[22], NLN[23] o MERLOT[24], al no almacenar el contenido de los objetos de aprendizaje, aportan una gran

[22]Campus Alberta Repository of Education Objects - http://careo.ucalgary.ca/
[23]National Learning Network - http://www.nln.ac.uk/
[24]Multimedia Educational Resource for Learning and Online Teaching - http://www.merlot.org/

variedad de materiales educativos y herramientas para compartir y explorar el conocimiento. Estos repositorios son catálogos de diversos materiales de e-learning, tales como animaciones, ejercicios, simulaciones, tutoriales, etc. También almacenan los resultados de los revisores, los creadores de los metadatos y diversos comentarios de los usuarios que aportan calidad al conjunto del repositorio.

Al proporcionar un soporte para albergar los metadatos, estos repositorios desempeñan un papel importante de cara al futuro. No sólo los humanos pueden consultar y buscar información, sino también cualquier tipo de aplicación, agente o sistema software externo (ver figura 2.8). Sin embargo, será difícil procesar correctamente la información existente en los registros de metadatos, a menos que estos cumplan unos niveles mínimos de compleción y calidad. Como se verá en las siguientes secciones, uno de los principales problemas de estos repositorios es el de carecer de un modelo conceptual que establezca qué es exactamente un objeto de aprendizaje y qué descriptores de metadatos asociados a cada una de las diferentes conceptualizaciones hay. Sin un acuerdo explícito y universal sobre el modelo de metadatos a utilizar, ni la completa certeza de que los registros de metadatos están completos, se hace difícil la implementación de ciertos niveles de automatización en estos repositorios.

A lo largo de esta sección se describen primero las tecnologías utilizadas en la construcción de repositorios, seguidamente se presenta un análisis de los repositorios de objetos de aprendizaje de mayor relevancia en la actualidad y finalmente se enuncian conclusiones relevantes para el desarrollo de la presente investigación.

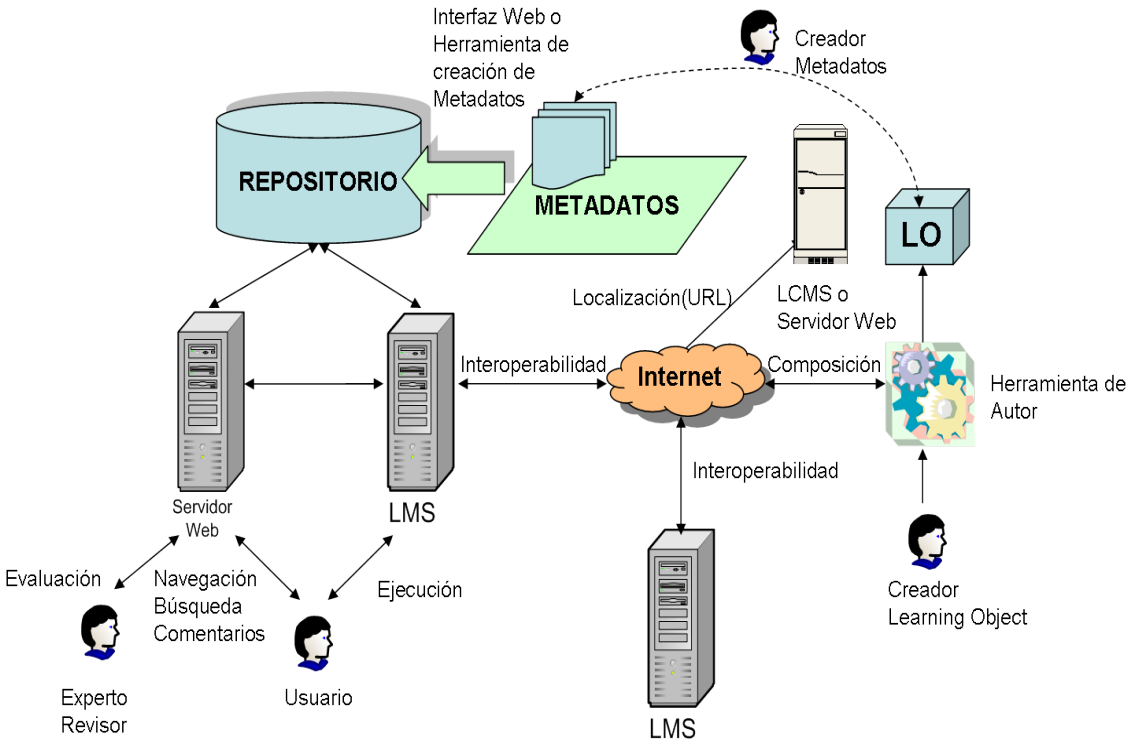

Figura 2.8: Nuevo esquema de un repositorio Web de objetos de aprendizaje.

2.5.1. Tecnologías utilizadas para la construcción de repositorios

Dentro de esta sección se exponen las tecnologías más relevantes utilizadas dentro de la construcción de los repositorios de objetos de aprendizaje. Casi todos los repositorios analizados exponen una interfaz web para el acceso a la funcionalidad expuesta, así como otras interfaces de acceso basadas en los nuevos protocolos de interoperabilidad entre aplicaciones (como SOAP [W3C03]). Se ha decidido incluir esta sección para aclarar y exponer las distintas tecnologías que potencialmente podrían ser utilizadas para la construcción del repositorio que se describe en la sección 4.2 del capítulo 4.

En cuanto al entorno de ejecución de los repositorios, existen dos arquitecturas posibles:

- Repositorio como módulo integrado dentro de una aplicación web: las interfaces exhiben un conjunto de páginas web dinámicas que acceden de forma interna a los servicios del repositorio. PHP o CGI son las tecnologías que más se utilizan para la implementación de los repositorios con este enfoque.

- Repositorio como módulo totalmente independiente: arquitectura que permite separar las interfaces de los servicios del repositorio. Desde el punto de vista de la ingeniería del software es el mejor método, ya que consigue desacoplar las distintas capas de una aplicación. Por ejemplo, pueden crearse interfaces de acceso desde distintos dispositivos para acceder a la funcionalidad ofrecida. Para construir internamente la funcionalidad del servidor, se pueden utilizar las últimas tecnologías de programación: dotNET o Java, aunque enfoques como el de MERLOT utilizan CGI's para la implementación de funcionalidades. Los más modernos aportan servicios de acceso normalizados (ej. IMS DRI) en arquitecturas modernas distribuidas (Servicios Web).

El almacén de datos en los repositorios actuales suele ser un sistema gestor de bases de datos, utilizando el modelo relacional para conceptualizar los datos mediante un esquema.

A continuación se expone un breve análisis de las distintas tecnologías consideradas en esta investigación.

Repositorio como aplicación web

Desde el punto de vista de la ingeniería web, un repositorio de objetos de aprendizaje es una aplicación alojada en un servidor de aplicaciones web, que posee una serie de interfaces y módulos funcionales generalmente bajo una arquitectura MVC.

Por tanto, actualmente las herramientas de desarrollo pueden variar en función del tipo de servidor que aloje la aplicación:

- Servidor Apache: es uno de los servidores web más populares, permite ser instalado tanto en sistemas Windows como Unix/Linux. Su arquitectura basada en módulos funcionales, permite ejecutar diferentes secuencias de comandos PHP, PERL o CGI's. Cabe mencionar la integración de Apache con PHP, uno de los lenguajes de programación web más utilizados [Mag]. Gran parte de las herramientas orientadas a la gestión de contenidos didácticos, como ATutor o Moodle han sido desarrolladas bajo este lenguaje. De entre todas las características a favor destaca su fin de lenguaje libre no atado a licencias restrictivas y compiladores propietarios.

- Microsoft - dotNET: durante los últimos años Microsoft ha desarrollado una nueva plataforma de ejecución de aplicaciones. Al mismo estilo que SUN, las aplicaciones son ejecutadas sobre el CLR (Common Language Runtime) que interpreta el código intermedio generado tras la compilación y ensamblado de los lenguajes que cumplan la especificación marcada por el CLS (Common Language Specification). A diferencia de Java, aún no existe ninguna implementación de Microsoft del framework .net para Unix/Linux. En su lugar, existen iniciativas como MONO que intentan dar soporte a las especificaciones elaboradas para .net. Microsoft ha conseguido reflotar su mercado de desarrollo y explotación debido a la amplia aceptación de ASP.NET [Mag]. Con el desarrollo de Visual Studio, Microsoft ha conseguido un entorno de desarrollo y un lenguaje web (ASP.NET) con la capacidad de desarrollar aplicaciones web como aplicaciones de escritorio. ASP.NET permite crear componentes web reutilizables de forma sencilla y rápida, bajo el entorno de ejecución aportado por el servidor de aplicaciones Internet Information Server (IIS).

- Sun MicroSystems - J2EE: la plataforma J2EE de Sun MicroSystems proporciona un amplio conjunto de especificaciones, bibliotecas y servidores. La plataforma utiliza la máquina virtual de Java, por lo que cualquier aplicación desarrollada puede ser ejecutada en cualquier sistema operativo. En el desarrollo web la comunidad de desarrolladores de Java proporciona múltiples herramientas, de entre las más importantes cabe mencionar como programas servidor Glassfish[25], Tomcat[26] o JBoss[27]. J2EE proporciona dos tecnologías para implementar el patrón MVC: Struts[28] y Java Server Faces (JSF)[29]. Struts permite que el desarrollador se concentre en el diseño de aplicaciones complejas como una serie simple de componentes del Modelo y de la Vista intercomunicados por un Control centralizado. Por otro lado, la tecnología JSF proporciona un nuevo enfoque orientado a la idea de eventos en la interfaz, similar al de la programación de aplicaciones escritorio (idea similar a ASP.NET). Además JSF permite crear nuevos componentes web capaces de ser reutilizados en varias aplicaciones web.

Servicio Repositorio

Un repositorio de objetos de aprendizaje puede implementar varios servicios de acceso externo para habilitar la interoperabilidad con otras aplicaciones o repositorios. La implementación de la funcionalidad del repositorio queda encapsulada dentro del conjunto de clases de las que se compone el componente. El modo de implementación puede ser propietario o basado en un estándar (ejemplo IMS/DRI). A continuación se exponen los diferentes tipos de implementación de estos servicios:

[25]Glassfish Community - https://glassfish.dev.java.net/
[26]Tomcat - http://tomcat.apache.org/
[27]JBoss - http://labs.jboss.com/
[28]Struts - http://struts.apache.org/
[29]Tecnología JSF - http://java.sun.com/javaee/javaserverfaces/

- Servicios Web: un servicio web expone una interfaz de acceso acorde a una especificación formal expuesta en un lenguaje (WSDL [W3C01]). Pueden ser utilizados independientemente de si existen cortafuegos o NAT (Network Address Translation). Esto es posible gracias a la invocación al protocolo de comunicación HTTP / SOAP [W3C03] y a la posibilidad de utilizar el puerto 80. Desde el punto de vista de la arquitectura del repositorio, para habilitar los servicios de interoperabilidad IMS/DRI la especificación de Servicios Web cubre los requisitos. Al ser una tecnología con una serie de estándares publicados por el W3C (WSDL, SOAP o UDDI [OAS02]), es posible realizar una aplicación cliente que utilice la funcionalidad expuesta por el repositorio (búsquedas o inserciones por ejemplo).

- Componente Enterprise Java Bean(EJB): un componente EJB [Net06] contiene código escrito en Java con una interfaz definida, administrado dentro de un servidor de componentes. Por lo tanto, cualquier biblioteca que implemente la especificación de acceso a EJB's puede acceder a ese componente. Desde el punto de vista de la arquitectura de un repositorio de objetos de aprendizaje, se puede definir un componente EJB que exponga la funcionalidad pública del repositorio. El protocolo de acceso al componente soporta traducción de llamadas por compatibilidad de Servicios Web y CORBA IIOP.

- Componente DCOM: la tecnología propietaria DCOM [Mic06] permite crear componentes con una interfaz definida para ser ejecutados bajo el servidor de componentes del sistema operativo Microsoft Windows. La interfaz de la funcionalidad expuesta por el repositorio puede ser definida con la tecnología DCOM, creando un componente con los métodos de acceso públicos y albergándolo en el servidor de componentes. Uno de los problemas acentuados al trabajar con esta tecnología propietaria son las restricciones dadas al implementar aplicaciones

que deseen acceder al componente.

- Objeto CORBA: la tecnología CORBA [OMG01]proporciona un conjunto de librerías de comunicación y administración de objetos remotos. La especificación define un protocolo genérico GIOP sobre el cual se definen otros protocolos secundarios en función del tipo de red o bus de acceso, por ejemplo IIOP es el protocolo definido para redes TCP/IP. Desde este punto de vista, se puede crear un objeto CORBA que implemente las funciones básicas de acceso al sistema y quede albergardo en un proceso adaptador de objetos (POA - Portable Object Adapter).

- Protocolos propietarios: también se pueden definir distintos protocolos de acceso utilizando diversas librerías de comunicación. En general, las librerías que implementan la funciones de comunicación TCP/IP permiten la construcción de cualquier tipo de protocolo bajo estas redes. Como ejemplo el paquete java.net de la api de Java o System.Net de la librería de clases bases del framework .net. La implementación de la interfaz pública del repositorio será mucho más complicada, pues a diferencia de las tecnologías mencionadas anteriormente, aquí es necesario generar un proceso servidor que gestione las múltiples peticiones realizadas por los clientes. Además, los clientes deberán implementar la funcionalidad de acceso al proceso servidor.

Persistencia

La gran cantidad de datos almacenados dentro de un repositorio de objetos de aprendizaje requiere de un sistema gestor que permita administrar los diferentes registros de metadatos y los recursos de los que se compone un objeto de aprendizaje (en el caso en el que el repositorio contenga el objeto).

Como posteriormente se comentará, la mayoría de los repositorios utilizan un esquema propietario o en determinadas circunstancias se ciñen a un estándar (por ejemplo IEEE LOM). Al no contemplar múltiples representaciones de un objeto de aprendizaje (teniendo en cuenta el uso de múltiples estándares), la implementación del modelo de datos puede ser realizada como la traducción simple del esquema de metadatos utilizado por el repositorio, en un modelo relacional capaz de ser creado en una base de datos. El resultado tras la traducción puede ser una o varias tablas con las relaciones necesarias que den soporte al esquema de metadatos.

A nivel técnico, para implementar un modelo relacional es necesario un sistema gestor de bases de datos (SGBD). A nivel comercial cabe destacar para grandes conjuntos de datos el uso de ORACLE o Microsoft SQL Server. En cambio, si se desea optar por una solución menos costosa puede utilizarse el SGBD MySQL o PostgreSQL.

Otras soluciones más novedosas utilizan el concepto de persistencia de objetos, utilizando bases de datos relacionales u orientadas a objetos como almacén de los registros de metadatos. De esta forma, los registros de metadatos quedan definidos por un conjunto de clases que poseen capacidades especiales de gestión del sistema de persistencia. Cuando se realiza alguna operación en el objeto (albergado en la memoria del servidor) que contiene el conjunto de metadatos asociados al esquema del repositorio, automáticamente se activan los mecanismos de gestión para mantener consistente el contenido de la base de datos orientada a objetos [Bar96]. Este tipo de sistemas aportan una mayor flexibilidad en la gestión de los objetos. Cabe destacar como bibliotecas de persistencia sobre modelos relacionales de bases de datos Hibernate y JDO. Como bases de datos totalmente orientadas a objetos: Versant[30], db4o[31]

[30]Versant: http://www.versant.com/
[31]db4o Server: http://www.db4o.com/

o objectDB[32].

En la siguiente sección se describen las características principales de los repositorios más relevantes tenidos en cuenta en esta investigación.

2.5.2. Análisis de los repositorios actuales

En esta sección se plantea un análisis de los repositorios más relevantes de objetos de aprendizaje seleccionados respecto al número de usuarios, número de metadatos almacenados y menciones a los mismos en literatura científica sobre el tema. Para ello el análisis se ha centrando en las siguientes características que pueden verse reflejadas en modo de resumen en la tabla 2.10:

- **Acceso libre (AL):** característica referida al modo de acceso al contenido de los recursos del repositorio. Pueden existir repositorios únicamente privados para una red interna (PRI), o con todo el contenido público (PUB), con contenido privado y público (MX).

- **Conectividad con herramientas de autor (CON):** en esta característica se reflejan las herramientas de autor con las que, mediante la existencia de algún proyecto software, se permita crear un canal de comunicación entre el cliente y el servidor.

- **Conectividad con otros repositorios (ELOR):** se indican los repositorios con los que está conectado[33] para la compartición de recursos didácticos.

- **API para clientes (API):** se refleja la existencia o no de un software intermedio que permita construir aplicaciones que utilicen los recursos del repositorio.

[32]http://www.objectdb.com/

[33]Federados en alguna red o de acceso libre bajo una especificación tal como IMS-DRI.

- **Esquema de metadatos (EM):** esta característica muestra si se utiliza un estándar de metadatos asociado a los registros almacenados en el repositorio (referido al almacenamiento o exportación de datos).
- **Creación de metadatos (CM):** se indica si un usuario registrado puede crear nuevos registros (USU) o si por el contrario los proveedores de contenidos del repositorio son los únicos responsables de los registros proporcionados (PROV).
- **Comentarios de los usuarios (CMU):** se refleja si los usuarios pueden o no realizar comentarios sobre los objetos de aprendizaje almacenados o referenciados (según el tipo de repositorio).
- **Evaluación por expertos (PREV):** se muestra si el repositorio permite introducir evaluaciones realizadas por expertos.
- **Búsqueda (BUS):** en esta característica se refleja el tipo de búsqueda que permite realizar el repositorio: simple (S) referida a la búsqueda sencilla sobre el nombre o descripción del objeto de aprendizaje, o avanzada (A) sobre atributos concretos de los registros.
- **Navegación (NAV):** esta característica indica si el repositorio posee alguna organización taxonómica navegable de su contenido.
- **Compleción (C):** característica que refleja el grado medio de compleción de los registros de metadatos albergados en el repositorio. Valores posibles definidos: bueno respecto a la compleción de los registros en un esquema amplio de definición (B), variable en un esquema completo (V) y deficiente por un esquema pobre de definición (M).
- **Apoyo institucional externo (AIE):** refiere las instituciones externas que apoyan, utilizan o avalan el proyecto.

A continuación se describen los repositorios más relevantes y se hace hincapié en cuáles de las características enumeradas incluyen.

CAREO

Campus Alberta Repository of Educational Objects (CAREO), es un proyecto de construcción de un repositorio de objetos de aprendizaje multidisciplinar para la búsqueda y localización de recursos, apoyado por CANARIE (Canadian Network for the Advancement of Research in Industry), BELLE (Broadband Enabled Lifelong Learning Enviroment) y la Universidad de Alberta en Canada. Actualmente es un pilar de referencia para el resto de repositorios, alberga registros de metadatos (basados en LOM con algunas entradas del perfil de aplicación CANCORE) que permiten localizar y buscar objetos de aprendizaje en Internet (ver figura 2.9). Además, CAREO ofrece la posibilidad de crear registros de metadatos a los usuarios. No existen los roles de evaluadores, ni encuestas para obtener el grado de calidad de los objetos referenciados.

Para el acceso externo al repositorio, independiente de la interfaz web, se propuso el proyecto ALOHA para que distintas herramientas y repositorios pudiesen acceder a los contenidos de CAREO. Los resultados de este proyecto derivaron en un cliente Java que permite interactuar con el sistema de almacenamiento de CAREO. La API permite programar de forma transparente el acceso al repositorio a través de XML-RPC [Win95], aunque recientes trabajos ponen de manifiesto futuras versiones sobre SOAP [W3C03]. Con ALOHA es posible insertar nuevos registros de metadatos dentro del repositorio utilizando una interfaz sencilla. Es un ejemplo de la versatilidad del acceso necesaria para estos repositorios, tanto para crear como para localizar recursos de forma autónoma y automática por parte de cualquier sistema LMS o programa externo.

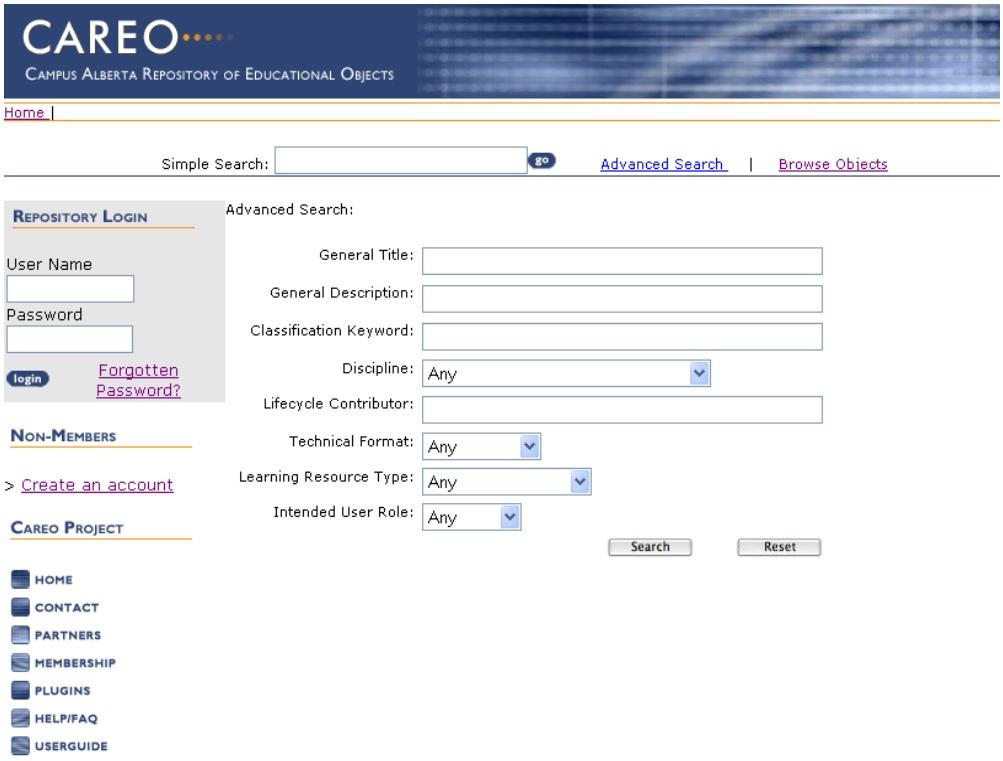

Figura 2.9: Careo: Interfaz de búsqueda

MERLOT

Multimedia Educational Resources for Learning and Online Teaching (MERLOT), es un repositorio que ofrece recursos libres con contenidos educativos multidisciplinares para mejorar la enseñanza y el aprendizaje en la educación superior. Contiene miles de enlaces a materiales educativos, multitud de evaluaciones realizadas por usuarios individuales así como numerosos resultados de las encuestas realizadas a expertos. Permite que los usuarios contribuyan con sus materiales educativos. MERLOT también es una comunidad de usuarios con el interés común de mejorar la forma de enseñar y compartir las experiencias de enseñanza. Posee un fuerte fondo pedagógico avalado por expertos en la educación superior y corporativa. Los miembros asociados de MERLOT dan un fuerte soporte a este proyecto.

La interfaz proporciona múltiples opciones de navegación y búsqueda de información. La figura 2.10 muestra una interfaz de MERLOT, en concreto, el árbol de temas que permite navegar por las distintas materias, así como un motor de búsqueda (parte superior derecha) que permite el acceso directo a los recursos. Además, cada objeto de aprendizaje posee asociado un nivel de calidad según las evaluaciones realizadas por expertos (en inglés "Peer Reviews"). En las interfaces (figuras 2.10 y 2.11) aparece una media general de la evaluación realizada por los expertos, en función de la calidad del contenido, la efectividad y la facilidad de uso. También se pueden consultar los comentarios realizados por los usuarios finales y las tareas de trabajo (en inglés "assignment") asociadas. Muchos profesores plantean posibles escenarios de aprendizaje dentro de una clase utilizando el objeto de aprendizaje, en cuyo caso se puede decir que marcan el modo de "uso" del objeto.

Un análisis de los objetos albergados en MERLOT, muestra que diferentes objetos incluyen diferentes grados de compleción e incluso diferentes campos de metadatos. La calidad de los registros de metadatos depende de los siguientes factores:

- La información proporcionada en los metadatos está relacionada con la experiencia y capacidades del creador del registro y del tiempo necesario para añadirlo.
- Las capacidades de edición o herramientas proporcionadas por el repositorio.
- El nivel de conocimiento del creador del registro sobre los estándares de metadatos de objetos de aprendizaje.
- El modelo conceptual del repositorio. Qué entiende el creador del registro que es un objeto de aprendizaje, y qué estructura de información asociada a los metadatos debe tener.

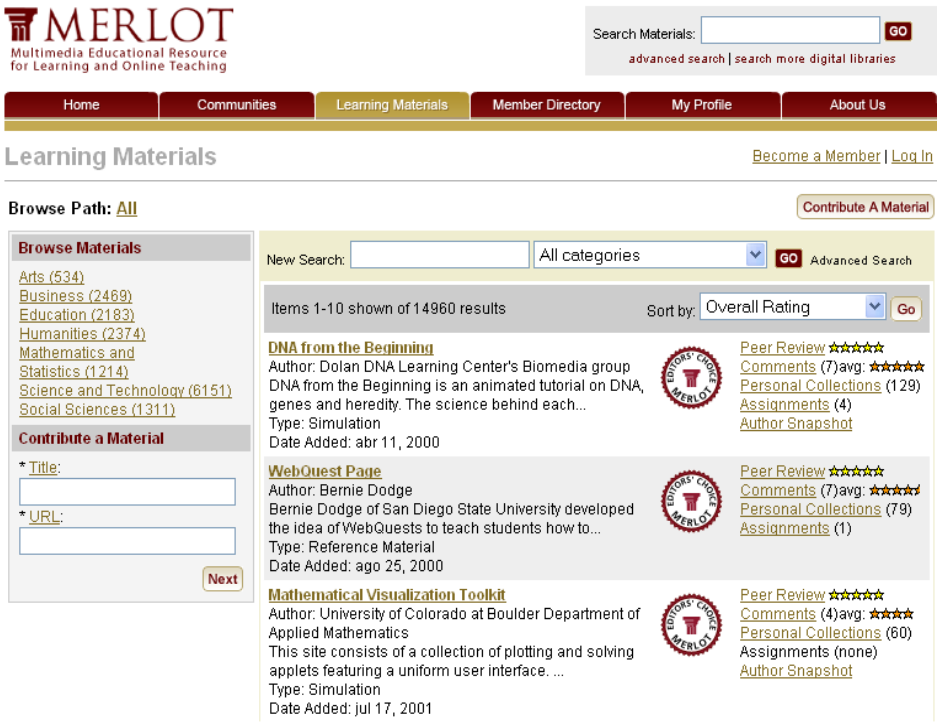

Figura 2.10: MERLOT, capacidades de navegación y búsqueda

La figura 2.11 muestra la información de metadatos asociada al objeto de aprendizaje *"Laboratorio Virtual de Química"*, incluida su localización mediante un hipervínculo (ver el atributo "*location*" en la figura 2.11). En este caso, se proporciona la información del objeto en unos pocos campos de metadatos, la mayoría en forma de descripciones textuales en lenguaje natural. Este hecho demuestra la dificultad de procesar automáticamente la información albergada dentro del repositorio: una aplicación software con la capacidad de realizar determinadas tareas automáticas, tales como la composición de nuevos objetos o la recomendación de objetos específicos para conseguir unos determinados objetivos pedagógicos, difícilmente sería capaz de procesar adecuadamente la información sobre los contenidos y los objetivos de aprendizaje del objeto incluidos, por ejemplo, en el campo *"descripción"*.

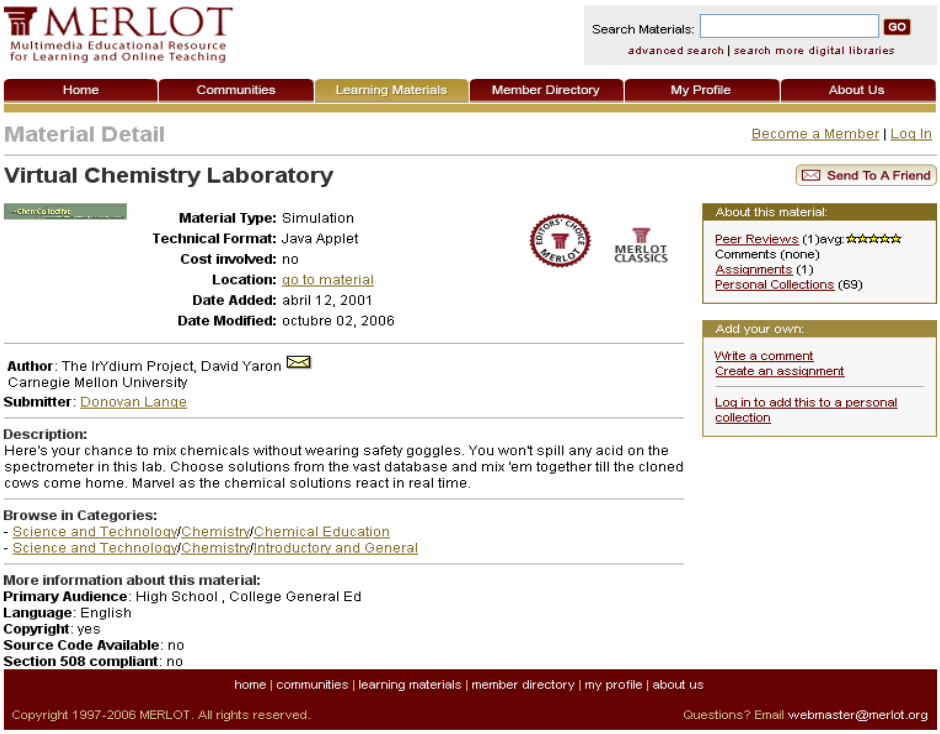

Figura 2.11: MERLOT, objeto de aprendizaje

NLN

National Learning Network (NLN), es una red destinada a fomentar el uso de las tecnologías de e-learning en Inglaterra. NLN aporta cientos de materiales didácticos de gran calidad, además de estar apoyada por el LSC (Learning and Skills Council) y sus socios BECTA (British Educational Communications and Technology Agency), JISC (Joint Information Systems Commitee), LSN (Learning and Skills Network), NILTA y UKERNA-JANET (Red de universidades, Colegios, Centros de investigación y comunidades de e-learning del Reino Unido).

La figura 2.12 muestra los metadatos del objeto de aprendizaje *"Dieta, Ejercicios y Recreación"*. Al igual que en MERLOT el objeto está descrito en lenguaje natural,

por lo que resulta igualmente difícil realizar tareas automáticas con agentes software externos. Además de la información mostrada en la figura 2.12, cada objeto de aprendizaje en el repositorio de materiales didácticos NLN tiene asociado un registro de metadatos llamado "Tutor documentation", donde se incluye información adicional sobre el propósito, los objetivos didácticos, el tiempo de estudio aproximado, el tipo de unidad didáctica o los prerequisitos de conocimiento, entre otros. Esta información, aunque es útil, no está formalizada dentro de un registro de metadatos de acuerdo con los estándares existentes (como LOM), y en consecuencia no proporciona una ayuda adicional en el esfuerzo para estandarizar y unificar los atributos requeridos para describir adecuadamente el objeto de aprendizaje dentro del repositorio.

A diferencia de otros repositorios, NLN posee un motor de ejecución de los objetos de aprendizaje. En la figura 2.13 se puede ver la apariencia del objeto de aprendizaje "Dieta, Ejercicios y Recreación" bajo el entorno de ejecución.

FREE

Federal Resources for Educational Excellence (FREE), es un repositorio de recursos e-learning apoyado por el departamento de educación de los EEUU. Más de treinta agencias federales se organizaron en 1997 con el objetivo de crear un sistema que permitiese una localización fácil y organizada de recursos utilizados en la enseñanza y el aprendizaje. El mantenimiento lo realiza un grupo de trabajo, encargado de actualizar y aportar nuevos recursos. A pesar de poseer una gran cantidad y calidad en el contenido de los objetos almacenados, existen diferencias relevantes con otros repositorios:

- Los usuarios no pueden crear nuevos registros de metadatos, sólo pueden localizar los recursos navegando por un árbol de categorías generales o mediante una sencilla búsqueda por descripción (ver figura 2.14).

- Tampoco hay ningún sistema de evaluación de la calidad de los contenidos por

Figura 2.12: NLN, registro de Metadatos

parte de los usuarios, sino que estos se ven supeditados a las directrices del grupo de trabajo de FREE.

- Los registros no siguen un estándar de metadatos para objetos de aprendizaje, únicamente almacenan la dirección de localización (URL) y una breve descripción en lenguaje natural, aunque estén clasificados por áreas de conocimiento. Este hecho limita las capacidades de búsqueda directa e interoperabilidad con otros sistemas (otros repositorios o herramientas de autor).

- El acceso se realiza únicamente por un cliente Web, por lo que resulta difícil

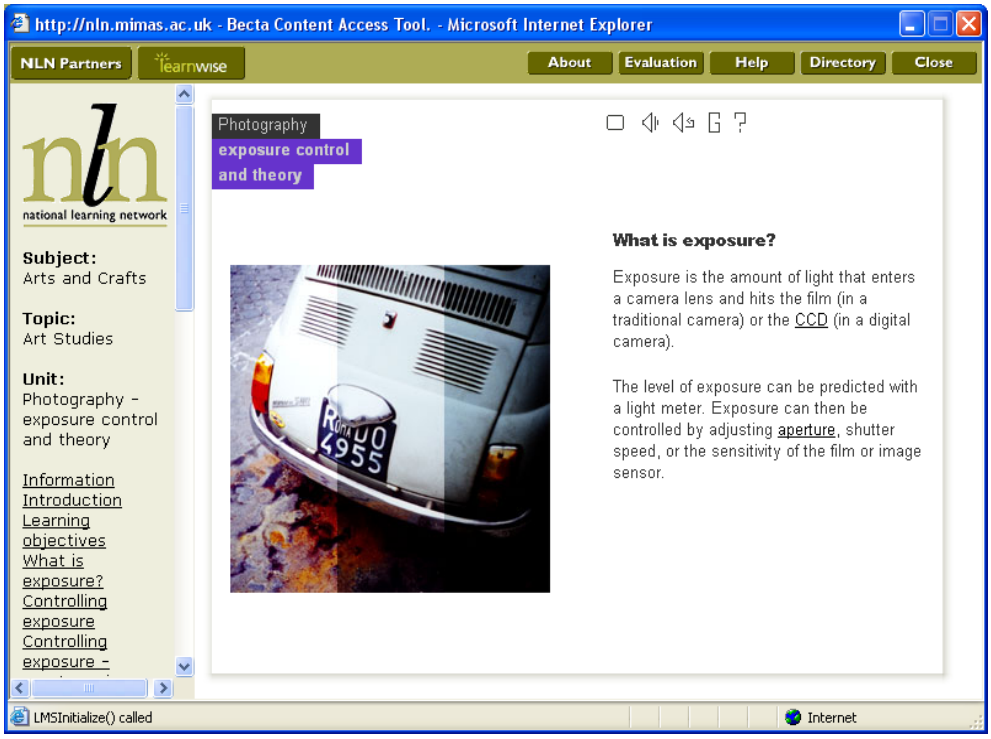

Figura 2.13: NLN, objeto de aprendizaje

crear módulos de programación que permitan acceder y utilizar el contenido del repositorio.

Gracias a la filosofía de FREE cualquier usuario de Internet puede localizar recursos de gran valor didáctico avalados por multitud de agencias o instituciones importantes de EEUU (NASA, Galería Nacional de Arte, la Fundación Nacional de la Ciencia, entre otros). En contraposición la tecnología ofrecida posee multitud de restricciones que limitan la interoperabilidad con otros sistemas, de tal forma que su uso se ve limitado a la localización de recursos por navegación o búsqueda simple a través de una interfaz Web.

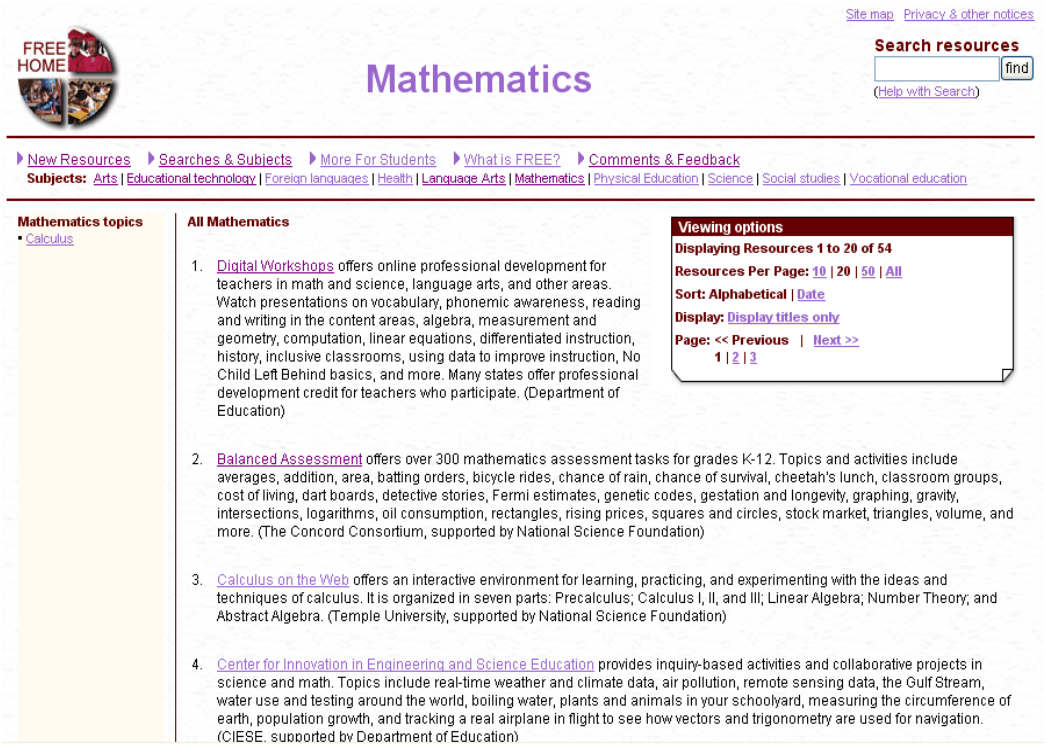

Figura 2.14: FREE, búsqueda y navegación

MCLI (Maricopa Center for Learning and Instruction)

Organismo dedicado a aportar recursos, programas y servicios para mejorar la calidad en la enseñanza mediante trabajo colaborativo entre distintas facultades, administraciones y amplios grupos de trabajo de Arizona. Uno de los proyectos del MCLI es el MLE (Maricopa Learning Exchange), un repositorio con más de 1400 "paquetes". Cada paquete representa un registro de metadatos (ver figura 2.15), que incluye un conjunto de atributos básicos para su descripción y localización. Entre estos atributos se incluye el título, disciplinas a la que pertenece, descripción, creador, enlaces a recursos y otros sitios Web, comentarios e información adicional.

MLE es un repositorio en el que sólo los usuarios que trabajen dentro de los centros

maricopa learning exchange

Packing Slip

SOME RIGHTS RESERVED creative commons

This work is licensed under a Creative Commons License

item	**Nutrition Now Unit 1 PowerPoint (key concepts)**
contact	Laura May (Mesa Community College) lauramay@mail.mc.maricopa.edu *1 of 26 packages by Laura May*
credits	"Nutrition Now" 3rd edition, Judith E. Brown, published by Wadsworth
college(s)	Mesa Community College
discipline(s)	Nutrition
summary	This 25 slide presentation and notes goes with Unit 1 of "Nutrition Now" 3rd edition, published by Wadsworth.
details	This presentation covers the meaning and definition of nutrition and briefly covers ten basic concepts of nutrition. These ten concepts are: 1. Food is a basic need of humans 2. Food provides energy, nutrients and other substances needed for growth and health. 3. Health problems related to nutrition originate in cells. 4. Poor nutrition can result from inadequate and excessive levels of nutrient intake. 5. Humans have adaptive mechanisms for managing fluctuations in nutrient intake. 6. Malnutrition can result from poor diets, disease states, genetic factors and combinations of these causes. 7. Some groups of people are at higher risk of becoming inadequately nourished than others. 8. Poor nutrition can influence the development of certain chronic diseases. 9. Adequacy, variety, and balance are key characteristics of a healthy diet. 10. There are no "good" foods or "bad" foods. ***Note!*** *As a professional courtesy to the owner of this package, if you use some aspect of this package or have some thoughts about it, please share your feedback via the comments form below.*

Figura 2.15: MCLI - MLE Packing Slip

del distrito de Maricopa (profesores a tiempo completo, parcial, administrativos o investigadores) pueden aportar nuevos contenidos. Cada paquete puede ser público (visible para el resto de usuarios de Internet) o privado (únicamente los usuarios registrados del repositorio pueden verlo) según desee el creador. Respecto al uso de estándares de e-learning, en el MLE no se describe ningún apartado sobre la estructura interna, ni tampoco existen protocolos de intercambio de datos. Parte de los campos necesarios para la creación de un paquete no se corresponden con los descritos en los estándares de metadatos de los objetos de aprendizaje.

NSDL

Nacional Science Digital Library (NSDL), biblioteca digital fundada por la Fundación de Ciencia Nacional de EEUU, con el objetivo de proporcionar un acceso rápido y organizado a recursos y herramientas de alta calidad existentes en la Web. Los "pathway" (o caminos de búsqueda) son proyectos encargados de proporcionar mecanismos de búsqueda distribuida entre diversos repositorios orientados en una disciplina. Entre los conectores más importantes están los proporcionados para la Biología (BioSciEdNet), la Informática (Fundación Shodor), Ingenierías (TeachEngineering de la universidad de Colorado), materiales para la ciencia (MathDL), Matemáticas (Asociación de Matemáticas de América), recursos multimedia (Teacher's Domain) y además recursos sobre Física y Astronómía (ComPADRE - Asociación de profesores de Física y Astronomía).
Como puede apreciarse en la figura 2.5.2, la colección de atributos utilizados en los registros de metadatos son limitados, orientados a la descripción y localización básica del recurso. Los usuarios sólo pueden consultar, los metadatos son gestionados por los organismos de gestión y proveedores de la NSDL.

CLOE

Co-operative Learning Object Exchange (CLOE) es un proyecto de colaboración entre las universidades e institutos de Ontario, para la creación, compartición y reutilización de recursos de aprendizaje multimedia. A la vez, CLOE es una organización fundada y establecida en Canadá, posee afiliaciones con MERLOT y GLOBE[34], así como otros organismos y organizaciones internacionales.

CLOE proporciona los mecanismos de compartición de recursos de aprendizaje multimedia a través de su repositorio. Cada institución crea y comparte sus recursos

[34]Global Learning and Observations to Benefit the Environment - http://www.globe.gov

Figura 2.16: NDSL, registros de metadatos

con otras, con el fin de crear otros recursos para utilizarlos en determinadas actividades didácticas. La figura 2.17 muestra la apariencia de la interfaz del repositorio.

MIT OCW

El proyecto Open Course Ware (OCW) apoyado por el Instituto Tecnológico de Massachusetts y la fundación William y Flora Hewlett, tiene como principal objetivo poner a disposición de profesores y alumnos multitud de recursos educativos que puedan ayudar a las tareas de enseñanza o autoaprendizaje. Actualmente posee 1400 cursos publicados, con las expectativas futuras de seguir creciendo gracias al apoyo de las distintas facultades del MIT. Con este repositorio, se brinda la oportunidad de conocer y disfrutar de los materiales didácticos de uno de los más prestigiosos institutos tecnológicos del mundo.

El usuario no necesita registrarse para poder obtener cualquier tipo de material, ni

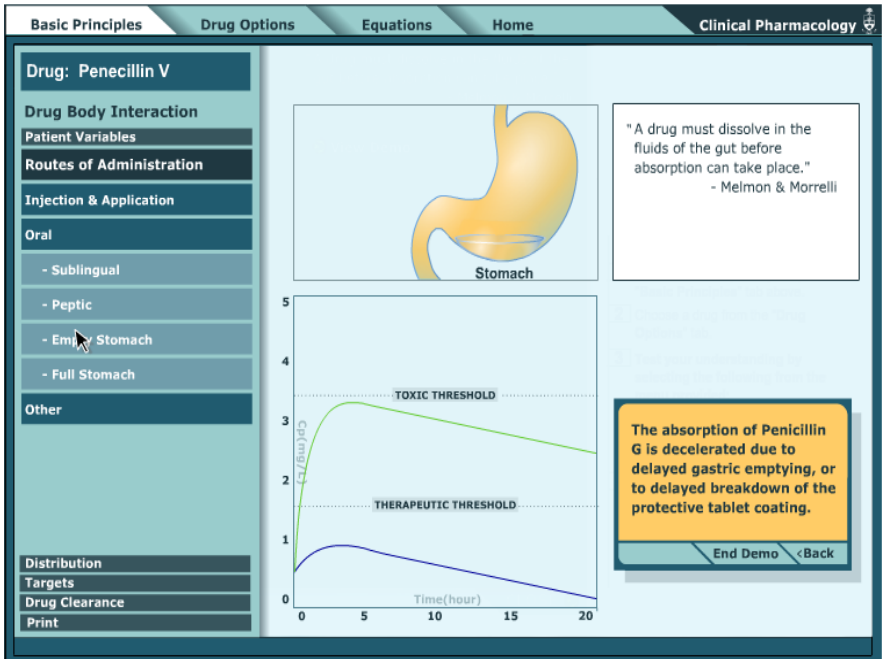

Figura 2.17: CLOE Learning Object

tampoco interactua con el repositorio aportando sus propios materiales. Este repositorio aporta un gran valor a la comunidad científica como recurso de autoaprendizaje y una visión global del beneficio de liberar los materiales didácticos en este tipo de repositorios.

El repositorio ofrece un mecanismo de búsqueda simple y otro de búsqueda avanzada detallando las características de búsqueda (frase exacta, al menos una de las palabras, sin las palabras especificadas) y el tipo de recurso a localizar. Los resultados ofrecidos son múltiples, desde recursos individuales hasta cursos completos con contenidos normalizados en un esquema de metadatos (ver figura 2.18). También ofrece la posibilidad de navegar para localizar los recursos organizados por departamentos y disciplinas.

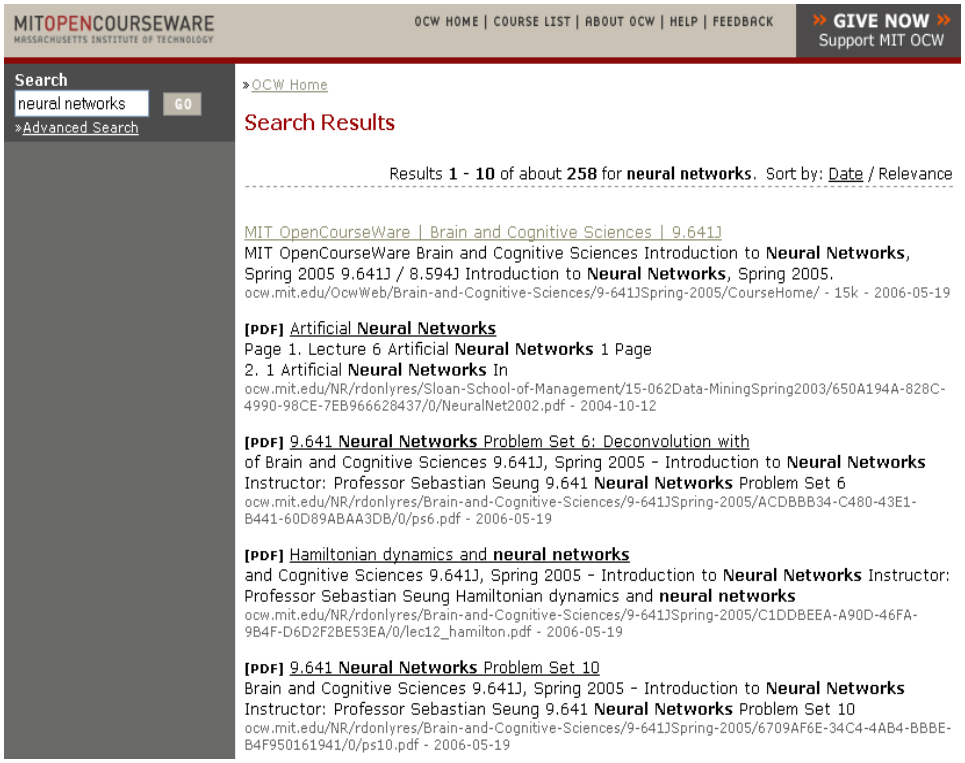

Figura 2.18: MIT OpenCourseWare

GEM

Este proyecto proporciona un repositorio como puerta de acceso libre a multitud de materiales y herramientas didácticas. GEM (Gateway to Educational Materials) está apoyado por NEA (National Education Association), una importante asociación relacionada con la educación en EEUU, centrada en el apoyo las nuevas tecnologías para la enseñanza en el sector público. GEM recoge el esfuerzo conjunto de los organismos dedicados a mejorar la enseñanza pública.

Los usuarios pueden registrarse y ser miembros de la red GEM, además algunos pueden adquirir privilegios para realizar evaluación de materiales. Los recursos están descritos en un estándar propio que incluye campos que indican el nivel pedagógico al cual están orientados (ver figura 2.19).

Por otro lado GEM facilita un árbol de categorías estructurado, cuya taxonomía interna permite realizar navegaciones óptimas dentro de las múltiples categorías de los objetos de aprendizaje que alberga. La búsqueda es simple y se realiza por tres atributos diferentes: título, descripción o palabras clave. Respecto a la integración con herramientas de autor, no se conoce ningún desarrollo sobre el contenido del repositorio, así como ninguna API que permita interactuar con aplicaciones cliente.

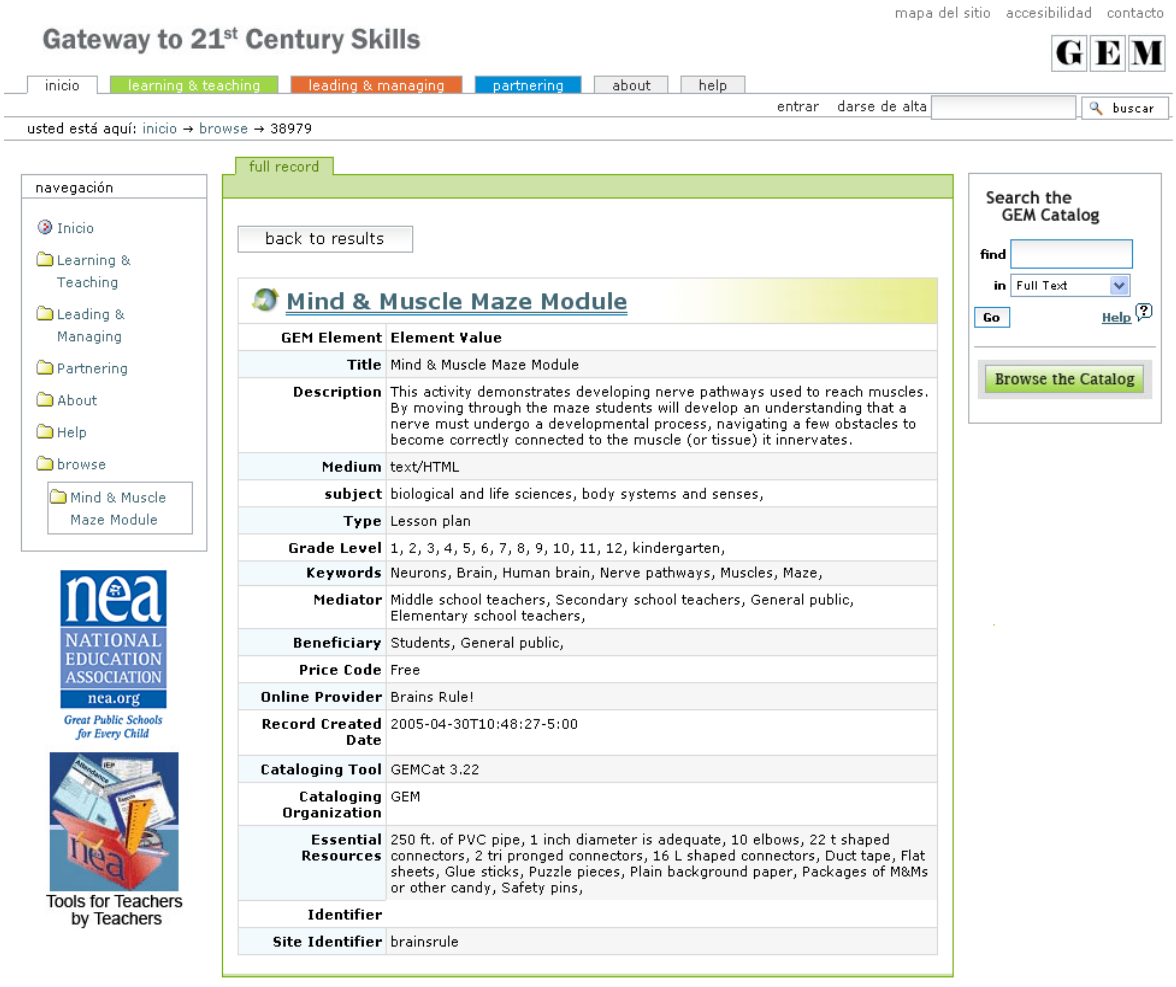

Figura 2.19: GEM - Metadatos de un objeto de aprendizaje

2.5.3. Conclusiones del análisis

El análisis realizado pone de relieve las diferencias existentes entre los distintos repositorios, que a modo resumen se muestran en la tabla 2.10.

REPO	AL	CON	ELOR	API	EM	CM	CMU	PREV	BUS	NAV	C	AIE
CAREO	PUB	SI	NO	SI	SI	USU	NO	NO	A	SI	V	SI
MERLOT	PUB	SI	SI	SI	NO	USU	SI	SI	A	SI	V	SI
NLN	PUB	SI	NO	NO	NO	PROV	NO	NO	A	SI	B	SI
FREE	PUB	NO	NO	NO	NO	PROV	NO	NO	S	SI	M	SI
MCLI	MX	NO	NO	NO	NO	USU	SI	NO	A	SI	V	SI
NSDL	MX	NO	NO	NO	NO	PROV	NO	NO	A	SI	V	SI
CLOE	PRI	NO	NO	NO	SI	USU	NO	NO	A	SI	V	SI
MIT OCW	PUB	NO	NO	NO	NO	PROV	NO	NO	A	SI	B	SI
GEM	PUB	NO	NO	NO	NO	PROV	NO	SI	S	SI	B	SI

Tabla 2.10: Características de los Repositorios

Casi todos los repositorios ofrecen contenidos públicos para la mejora de la enseñanza, muchos orientados al propio desarrollo del sistema educativo del país o comunidad que le da soporte. La naturaleza de acceso libre desde Internet ofrece la posibilidad de poder obtener y compartir múltiples recursos didácticos. Respecto al contenido y la calidad del mismo hay diversos enfoques:

- Aquellos repositorios que permiten crear nuevos registros de metadatos de recursos didácticos, como CAREO o MERLOT, presentan numerosas variantes en la compleción del registro y la calidad del contenido. Respecto a la compleción se han encontrado múltiples carencias en los registros de metadatos, siendo en algunos casos muchos de ellos incompletos, incluyendo únicamente una breve descripción. Respecto al contenido, en estos repositorios la calidad depende mucho del creador del objeto de aprendizaje, si bien con las evaluaciones realizadas por los expertos se puede garantizar la calidad del contenido. El formato de los registros de metadatos varía también, ya que no se utilizan los estándares de metadatos de forma uniforme.

- En cambio en los repositorios en los que un usuario registrado no puede crear metadatos, el formato y el contenido del mismo es uniforme.

- Otro caso particular es el de los repositorios que permiten la creación de nuevos registros a usuarios registrados e identificados por un organismo administrativo, tales como CLOE, GEM o MCLI. En este caso, sobre un análisis de los objetos almacenados existen discrepancias leves de completitud en los registros, así como en el contenido, aunque este queda garantizado por algún servicio de mantenimiento del repositorio.

En relación al esquema de metadatos, la característica EM (Esquema de Metadatos) refleja que 7 de los 9 repositorios utiliza un esquema distinto de metadatos. Esto es debido a que cada repositorio ha decidido utilizar una conceptualización propia del término "objeto de aprendizaje'. El estudio de McGreal [McG04] establece que la terminología actual considera que un objeto de aprendizaje puede ser desde cualquier cosa digital o no digital hasta algo concreto que tenga un propósito didáctico. Al final de este estudio, se propone una nueva definición basada sobre concordancias entre todas las definiciones, que pueden ser la base de una nueva generación de repositorios más flexibles que soporten las diferentes conceptualizaciones.

Respecto a la búsqueda y navegación, podemos concluir que todos los buscadores poseen un esquema de clasificación que permite navegar e identificar de forma rápida cualquier tipo de objeto. Casi todos los repositorios analizados permiten realizar búsquedas avanzadas (BUS - 7 de 9), pero la mayoría no permite la conectividad con otros repositorios (CON - sólo 1 de 9) ni el acceso a las funcionalidades del mismo desde una aplicación cliente utilizando una biblioteca de programación (API - sólo 2 de 9), ni la conectividad con herramientas de autor (CON - sólo 3 de 9). Cuando un repositorio proporciona una serie de funciones públicas accesibles desde cualquier

aplicación (utilizando una biblioteca de programación específica) o desde otros repositorios (interconectividad) existe un consenso en la necesidad de incluir información en los metadatos, junto con el objeto o en un registro que enlace al objeto. Además, la información debería ajustarse a un estándar, preferiblemente IEEE LOM [LTS02].

Por otro lado, merece la pena resaltar que gran parte de la taxonomía de clasificación utilizada en estos repositorios difiere en términos de vocabulario. Como ejemplo, en el repositorio NLN existe una rama denominada Humanities, en cambio en el repositorio del OCW del MIT la rama similar se denomina "Humanistic Studies". El uso de bases de conocimiento formales, como se verá más adelante, permitiría determinar conceptos únicos que clasificarían de forma uniforme múltiples recursos sin ninguna ambigüedad.

Como puede apreciarse, el análisis anterior revela importantes deficiencias en los repositorios actuales. Dichas deficiencias sugieren la creación de un nuevo modelo que dé soporte a múltiples conceptualizaciones bajo un esquema formal que permita definir la semántica de los metadatos. En el próximo capítulo de esta investigación se muestra una descripción avanzada del problema a abordar.

Capítulo 3

Planteamiento del problema

Yo empiezo con una idea y después hago algo con ella.

Pablo Picasso

A lo largo de este capítulo se realiza un análisis de las distintas carencias detectadas en los repositorios y en las especificaciones existentes de los objetos de aprendizaje, a partir del estudio presentado en el capítulo anterior. El análisis se centra en el desarrollo de las deficiencias encontradas, concretamente se estudiarán:

1. La existencia de la distintas conceptualizaciones del término "objeto de aprendizaje", que provocan problemas de entendimiento entre los distintos actores que interactúan con el repositorio.

2. El uso de registros de metadatos heterogéneos dentro de un repositorio, que dificulta la interacción y reutilización de los contenidos entre sistemas que soporten conceptualizaciones basadas en esquemas diferentes.

3. La carencia de una especificación formal para la información almacenada dentro de los registros de metadatos, que pone de manifiesto las limitaciones existentes a la hora de realizar un tratamiento automatizado sobre los datos.

4. Los problemas de interoperabilidad y comunicación entre diferentes repositorios

y aplicaciones externas, que limitan las capacidades de operación sobre un único tipo de repositorio, sin aprovechar el contenido de otros que utilicen otros esquemas e interfaces diferentes.

A continuación se detalla cada una de las deficiencias, planteando los puntos débiles en los que se centrará el desarrollo de los próximos capítulos, donde se elabora la propuesta de solución de la presente investigación.

3.1. Las distintas acepciones del termino objeto de aprendizaje: el problema de la falta de flexibilidad

Del estudio realizado en el capítulo anterior referente al modelo de almacenamiento de los registros de metadatos en un repositorios de objetos de aprendizaje se derivan las siguientes conclusiones:

- Los repositorios actuales están orientados a almacenar y procesar metadatos asociados a recursos digitales didácticos. En algunos ámbitos de la enseñanza puede darse la necesidad de incluir en estos repositorios registros de metadatos de objetos existentes físicamente en algún lugar de un centro docente utilizados en alguna actividad de enseñanza (como por ejemplo libros, apuntes o instrumentos utilizados en la ejecución de una práctica), por consiguiente estos repositorios no pueden procesar todos los tipos de recursos utilizados en los diferentes escenarios y actividades de la enseñanza.

- Los repositorios no poseen un esquema que permita unificar todas las acepciones del concepto “objeto de aprendizaje”. Según las conclusiones presentadas en la sección 2.5.3 del capítulo anterior, la mayoría de los repositorios analizados

permiten incluir registros de metadatos basados en un esquema propietario o en un estándar específico para un determinado tipo de objeto de aprendizaje. Esta situación es debida a la carencia de un modelo flexible que establezca todas las posibles conceptualizaciones del término "objeto de aprendizaje" y defina los descriptores de metadatos asociados a cada conceptualización diferente.

Como ejemplo puede apreciarse la diferencia existente entre los campos utilizados en un registro de metadatos de un libro de la biblioteca nacional (ver figura 3.1) y los campos utilizados en los registros de metadatos de los repositorios analizados (ver figuras de los formularios de entrada de objetos de aprendizaje 2.11, 2.13 o 2.19 y las conclusiones del análisis en la sección 2.5.3). Además de existir diferencias entre los esquemas utilizados por los repositorios, se percibe la carencia de gestionar y procesar metainformación utilizada en otro tipo de ámbitos, como es en el presente caso de catalogación de libros. En la figura 3.1 pueden apreciarse campos de metadatos utilizados para la clasificación del libro (campo CDU - Clasificación Decimal Universal [Mci07], de identificación como el ISBN, el LCCN y la clave interna del sistema, o relativos al aspecto físico del mismo entre otros. En los repositorios analizados no existe un mecanismo automático que permita procesar este tipo de metainformación relevante, ya que su ámbito queda limitado únicamente a la catalogación de los recursos digitales existentes en Internet o dentro de una intranet privada.

Es interesante por tanto aclarar que un objeto de aprendizaje no es sólo un recurso o un curso digital, sino que también puede serlo un conjunto de folios que contienen unos apuntes tomados en una clase de un profesor, un libro o una práctica de química junto con todos los recursos físicos necesarios (una probeta, una serie de elementos químicos y las instrucciones). Como puede apreciarse, en función del ámbito y el dominio de conocimiento los objetos de aprendizaje pueden adquirir determinadas

Figura 3.1: Registro de metadatos del libro el Çantar de Mío Cid".

acepciones y necesitar para su descripción de diferentes modelos de metadatos. Si nos ajustamos a la definición dada en la sección 2.2.1, cada objeto de aprendizaje almacenado en un repositorio debe tener asociado uno o varios registros de metadatos que lo describan para el posterior proceso del mismo en operaciones automáticas o semiautomáticas de búsqueda o composición. Para definir un modelo genérico que dé soporte a la definición anterior se ha de contemplar:

- La posibilidad de coexistencia de todas las posibles acepciones del término "objeto de aprendizaje" en un único modelo.
- El modo de gestión y proceso de operaciones dada la heterogeneidad de los registros de metadatos almacenados en un repositorio bajo un modelo flexible.

Estudios recientes sobre la unificación de las distintas conceptualizaciones de la definición de *objeto de aprendizaje*, sugieren la posibilidad de la coexistencia de todas las múltiples acepciones del término en una única definición [McG04]. De acuerdo con el estudio de McGreal sobre las distintas caracterizaciones de un objeto de aprendizaje, coexisten 5 definiciones. De la más general a la más especifica, son las siguientes:

1. Cualquier cosa.
2. Cualquier cosa digital, con o sin propósito educativo.
3. Cualquier cosa con propósito educativo.
4. Objetos digitales que tengan un propósito educativo.
5. Objetos digitales que tengan definidos de forma especial su propósito educativo.

- De acuerdo con la primera definición (representada en la figura 3.2, como [1] LearningObject-AsAnything), los objetos de aprendizaje son cosas que pueden ser utilizadas en eventos de aprendizaje o están provistos de descripciones que especifican los posibles usos didácticos. Por ejemplo un libro, un conjunto de recursos utilizados en una práctica de laboratorio o incluso un cassette que contiene una sección de audio relevante para una actividad de aprendizaje.
- La segunda definición ([2] LearningObject-AsAnythingDigital,figura 3.2) introduce el concepto de "*objeto digital*", incluyendo el significado de representación

digital utilizada en el aprendizaje. En esta definición quedan englobados todos los objetos de aprendizaje que poseen un formato digital, ejemplos de instancias concretas son archivos digitales de audio, videoconferencias, documentos electrónicos o aplicaciones.

- La tercera definición ([3] LearningObject-AsAnythingWithEducationalPurpose, figura 3.2) restringe los objetos de aprendizaje a aquellos que posean un propósito educativo, cuyo diseño irá orientado a este propósito. Las definiciones [2] y [3] proporcionan características que son esenciales en todas las especificaciones tecnológicas de e-learning, como SCORM [ADL03] o IMS LD [KOA03b]. El propósito del aprendizaje puede describirse en un registro de metadatos, sin necesidad de seguir una especificación formal como LOM [LTS02]. Como ejemplos concretos, apuntes de una clase o libros didácticos de apoyo a la enseñanza.

- Las últimas definiciones (cuarta y quinta) puede agruparse en una única definición, ([4] LearningObject-OtherSpecificAccounts, figura 3.2), se define un objeto de aprendizaje como una entidad digital con propósito didáctico que posee un esquema específico en sus registros y contenidos. En esta definición los registros de metadatos de los objetos de aprendizaje están obligados a seguir una especificación formal, tanto en su descripción del propósito didáctico como en su contenido para la ejecución en otros sistemas. Como ejemplo concreto, una lección de un curso que incluye información sobre su secuencia y ejecución dentro de un sistema LMS, así como la estructura y referencias a aplicaciones digitales con un propósito didáctico bien definido (applets o animaciones flash orientados a la enseñanza), videoconferencias de clases magistrales, documentos electrónicos con propósito didáctico o referencias a libros didácticos de apoyo a la enseñanza entre otros.

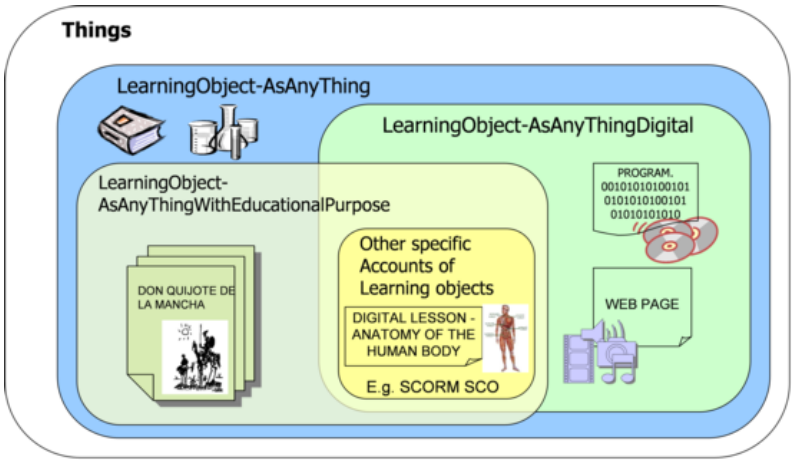

Figura 3.2: Representación del solapamiento de las diferentes conceptualizaciones de los objetos de aprendizaje

La propuesta de McGreal sugiere la creación de un nuevo modelo flexible de almacenamiento de metadatos para objetos de aprendizaje que permita incluir todas las posibles acepciones del término objeto de aprendizaje. A nivel digital si un repositorio implementa el modelo descrito podrían catalogarse de manera sencilla cualquier tipo de recursos utilizado en alguna actividad de aprendizaje.

En un repositorio con el modelo sugerido se podría incluir tanto un registro de metadatos que incluya el diseño de la actividad de aprendizaje IMS-LD [KOA03a], como también un registro que asocie los metadatos relativos al empaquetamiento de un curso en formato digital SCORM [ADL03]/AICC [Hyd01], o incluso en un nivel más general se podría incluir un registro de metadatos en formato EXIF [Sta05] o IPTC/XMP [IPT05] referente a una descripción de una foto de un cuadro almacenado como recurso digital.

3.2. Carencias de los registros de metadatos de los repositorios: el problema de la heterogeneidad

Escribiendo información en registros de metadatos sobre el contenido de los objetos de aprendizaje, se facilitan varios procesos, tales como el almacenamiento, la búsqueda y la recuperación desde repositorios distribuidos, así como la composición de nuevos materiales didácticos (entre otros). Aceptando las especificaciones de metadatos y los estándares, los objetos de aprendizaje son más operables y reutilizables, si bien existen diversos inconvenientes que conviene reseñar:

- En primer lugar, la información almacenada en gran parte de los repositorios (7 de 9 según las conclusiones presentadas en la sección 2.5.3 del capítulo anterior) no está definida en estándares internacionales como IEEE LOM [LTS02], por lo tanto, no está orientada a ser procesada por agentes externos. Este hecho, destaca la carencia de programar aplicaciones con capacidad de interactuar formalmente con el repositorio. (Ejemplo: recuperar los registros que tengan "renacimiento" en el campo *coverage*).

- Además, aquellos en los que se utiliza un estándar, no contienen información uniforme (en el sentido de que no siempre están completos los mismos campos de metadatos) tomando como muestra todos los registros existentes dentro del repositorio. Por lo tanto, existen registros con distintos grados de compleción en función de los criterios de inserción seguidos por el creador de metadatos, y de los datos existentes sobre el objeto. Por ejemplo, un registro puede especificar las relaciones con otros objetos (uso de la categoría *7. Relation* de IEEE LOM) y otro registro no incluirla.

- Por último, en los repositorios actuales no existe un consenso respecto al uso

de los términos utilizados como identificador de las categorías de los diferentes dominios de conocimiento de un objeto de aprendizaje, ni tampoco existen directivas de organización de los diferentes niveles de clasificación. Así por ejemplo, MERLOT utiliza como categoría raíz (del directorio inicial) *Matemáticas y estadística*, mientras que NLN en el mismo nivel utiliza la categoría *Ciencia y Matemáticas*, así como otros ejemplos indicados en las conclusiones del análisis (ver sección 2.5.3 del capítulo anterior). Además una categoría puede identificase con distintas palabras dada la variedad terminológica existente en torno a un mismo significado.

Respecto al primer inconveniente puede encontrarse una solución utilizando los estándares descritos en la sección 2.4 del capítulo anterior. El problema se da en la gran diversidad de los mismos existente por la definición tan amplia que posee el concepto objeto de aprendizaje. De nuevo apuntamos hacia una solución que proporcione un esquema flexible de almacenamiento para incluir la amplia gama de esquemas existente en el ámbito de los objetos de aprendizaje.

El segundo inconveniente requiere de mecanismos que permitan obtener el esquema sobre el cual se ha creado el registro de metadatos y los campos que ha utilizado. De esta manera, dentro de un modelo flexible de almacenamiento pueden procesarse aquellos registros que cumplan con los requisitos mínimos de compleción necesarios para una operación.

Finalmente, el último problema mencionado limita las capacidades de búsqueda y navegación dada la importancia de la clasificación de los registros de metadatos albergados en los repositorio. La existencia de un vocabulario general universal para todos los posibles términos utilizados en las diferentes categorías de conocimiento, capaz de ser procesado por una máquina, aportaría la base de una de las posibles

propuestas de solución general a este problema.

3.3. La carencia de formalismos de representación: el problema de la interpretación de los metadatos por parte de agentes software externos

Todos los inconvenientes expuestos en la sección anterior derivan a un problema de interpretación de los elementos existentes en un repositorio por parte de agentes software. El conocimiento que poseen e intercambian los agentes de software se encuentra formalizado en estructuras cognitivas que adoptan la forma de ontologías, por lo tanto, la única forma de interpretar el conocimiento existente en los metadatos pasa por el uso de una ontología que dé soporte a un modelo formal de conceptualización. [WJ94]

Los estándares actuales son de propósito descriptivo, dan información sobre los contenidos o el formato del objeto de aprendizaje, pero no disponen de una semántica de ejecución para los sistemas gestores de contenido didáctico (LMS), ni de un modelo formal que aporte significado dentro del contenido de los registros de metadatos.

Este problema es debido a la situación general existente en el procesamiento de la información escrita en lenguaje natural (el que los seres humanos utilizan para comunicarse), por lo tanto no indica una posible solución tan clara como la anterior. Un lenguaje natural posee una riqueza semántica que le proporciona ese carácter tan expresivo, se desarrolla con el enriquecimiento progresivo de la cultura de la población que lo utiliza, y además es muy difícil obtener una formulación completa del mismo.

Los repositorios analizados mantienen entradas descriptivas de texto como son el título, la descripción, las palabras claves o el contexto. El análisis del significado de la

información en un texto escrito en lenguaje natural provoca multitud de problemas y ambigüedades, muchas de ellas estudiadas en el área de investigación del procesamiento del lenguaje natural[1]. Actualmente los procesos de búsqueda de información analizan los textos escritos en lenguaje natural creando estructuras especiales que facilitan la obtención de resultados próximos a los deseados, como los índices de clasificación ([CGRU96],[W.03], [M.05]) o de afinidad y relación de tipo PageRank de Google [BP98]. Los resultados dejan de ser completamente satisfactorios en entornos de trabajo dónde se necesita comprender el significado introducido en los campos para obtener unos resultados.

Para entender mejor esta situación, se plantea el siguiente escenario: un usuario que desea introducir un registro de metadatos sobre un objeto de aprendizaje del dominio de las Matemáticas. El campo de metadatos referente a la descripción puede ser el siguiente:

> *Este objeto de aprendizaje tiene como objetivo introducir los fundamentos básicos de la geometría descriptiva, especialmente orientados al sistema diédrico. Utiliza una serie de applets que permiten representar gráficamente diferentes tipos de teoremas y problemas geométricos. Esta herramienta esta orientada a los estudiantes de esta disciplina con nivel universitario.*

Sobre el texto anterior se pueden aplicar diferentes tipos de métodos de búsqueda:

- Lineal: aplica un análisis de patrones similares por fuerza fruta, o con algoritmos de complejidad computacional inferior como Knuth-Morris-Pratt (KMP) o Boyer-Moore.
- Índice-clasificación: el texto se clasifica en una categoría bien de forma manual o utilizando algoritmos de clasificación de textos como Naive Bayes, SVM

[1]Del inglés - Natural Language Processing o NLP.

(Support Vector Machines), árboles de decisión ID3, C.45 o J.48, Rocchio, estomatológico por aprendizaje, AdaBoost, EM, SOM u otro tipo de algoritmos basados en técnicas de clasificación por incertidumbre aplicando conceptos de lógica difusa y probabilidad. El proceso de búsqueda utiliza este índice para obtener los resultados de forma rápida.

- Índice-relación: analiza las relaciones entre los documentos estableciendo una serie de pesos que definen la relevancia del documento según el número de referencias entre ellos. Este tipo de técnica es utilizada por algunos buscadores de Internet, como el algoritmo PageRank de Google.

- PLN: analiza el contenido del texto desde el aspecto *morfológico* - estructura y composición de las palabras, *sintáctico* - gramática, *semántico* - significado de las palabras y de las oraciones y *pragmático* - eliminación de ambigüedades por análisis del significado global. Este tipo de análisis puede proporcionar mecanismos de búsqueda más inteligentes utilizando bases de conocimiento para la deducción e inferencia de conceptos relacionados con la intención de búsqueda del usuario. El PLN requiere de una alta complejidad de cálculo, aún sigue en proceso de investigación y en muchos casos no obtiene resultados muy satisfactorios debido a la gran variedad y complejidad de los lenguajes humanos.

La sintaxis de un lenguaje natural se puede modelar mediante la utilización de un lenguaje formal. Esta característica elimina la capacidad de expresividad dada en un alto grado por un lenguaje natural, pero a la vez incrementa la capacidad de computación del mismo. Si un usuario experimentado (creador de metadatos) pudiese introducir información estructurada y formalizada en algún otro lenguaje más cercano a las máquinas, utilizando por ejemplo aserciones escritas en lógica de primer orden o descriptiva obtendríamos un resultado como el siguiente:

```
1 es−un(LOR01,LearningObject−AsAnythingWithEducationalPurpose).
2 relacionado_con−disciplina(LOR01,geometría−descriptiva).
3 relación−principal−con(LOR01,sistema−diédrico).
4 tiene−incluido(LOR01,programas,applets).
5 requiere−conocimientos−de(LOR01,nivel−universitario).
```

Este tipo de descripción, además de facilitar el proceso de búsqueda sugiere la posibilidad de incluir nuevos mecanismos de inferencia que permitan realizar búsquedas mucho más inteligentes. Con el ejemplo anterior, LOR01 es el identificador del objeto de aprendizaje y el inicio de cada sentencia es un predicado escrito en lógica de primer orden. Si deseásemos obtener objetos de aprendizaje relacionados con la geometría obtendríamos como resultado este objeto, ya que la geometría descriptiva es un área de conocimiento de la geometría general. Para ello es necesario tener almacenadas las relaciones existentes entre los diferentes conceptos utilizados, como por ejemplo: *relacionado_con-disciplina(geometría-descriptiva,geometría).*

3.4. Problemas de interoperabilidad y comunicación

Al no poseer ninguna restricción en el uso de las especificaciones y la falta de implementación de los mecanismos de intercambio propuestos por alguna de las especificaciones como IMS-DRI o la interfaz SQI de LORI, estos repositorios quedan aislados en un sistema cerrado, cuya única ventana de acceso es la interfaz web ofrecida desde el servidor de aplicaciones. Debido a esta situación y a las deficiencias de los repositorios explicadas en las secciones de este capítulo se pueden apreciar los siguientes problemas:

- Las funciones de intercambio de registros de metadatos y objetos entre los repositorios analizados son imposibles de realizar, ya que no existe ningún acuerdo ni recomendación implementada que regule este tipo de transacciones.

- Realizar aplicaciones que interactúen con estos repositorios resulta una labor complicada y la mayoría de los casos imposible. Por lo que las herramientas de autor que permiten componer cursos didácticos, como las ofrecidas por Moodle, Atutor, Adobe AuthorWare, LAMS o RELOAD no ofrecen la posibilidad de aprovechar y enlazar las referencias a los recursos que proporcionan estos repositorios. Esta situación limita la reutilización y uso de los recursos didácticos ofrecidos por los repositorios.

- Por otro lado, se limita la capacidad de proceso automático en relación a la composición automática de recursos descrita en la tesis de investigación elaborada por Sánchez Alonso [Alo05]. Por ejemplo, en el escenario de seleccionar entre dos objetos de aprendizaje (referenciados cada uno por un repositorio diferente) que consiguen los mismos objetivos didácticos, el mejor de ellos en función de las precondiciones del cliente.

- Tampoco se habilita un escenario para establecer un mecanismos de búsqueda distribuida entre varios repositorios. Por lo tanto, si un recurso no existe en un repositorio, de forma automática no podremos conocer si puede existir en otro.

- La automatización de los propios procesos de interoperabilidad se ve reducida ya que no se implementan mecanismos de descubrimiento de los servicios aportados por un repositorio. Arquitecturas como CORDRA proponen la implementación de este tipo de infraestructura. Por lo tanto, con los repositorios analizados una aplicación no puede descubrir qué repositorios hay en una red ni qué operaciones puede realizar con ellos.

- Las distintas conceptualizaciones dadas en el término “objeto de aprendizaje” también aportan problemas de interoperabilidad debido a las deficiencias anteriormente explicadas en las secciones 3.2 , 3.3. Si en un mismo repositorio

pueden darse distintos tipos de objetos de aprendizaje (como ya se describió en el modelo de McGreal), en diferentes repositorios la diversidad aumenta, por lo que el tratamiento de la información de manera distribuida en ambos se complica.

Dados los problemas anteriormente enunciados, la aportación de un posible modelo flexible indicado en al sección 3.2 y el uso de los estándares de interoperabilidad (IMS-DRI) utilizados en arquitecturas distribuidas como CORDRA aportaría una posible solución general.

3.5. Resumen

En este capítulo se han clasificado y descrito los problemas encontrados en el análisis realizado en la sección 2.5.3. En cada problema se han expuesto diferentes estudios y técnicas que sugieren la posibilidad de elaborar una solución enfocada a eliminar las deficiencias encontradas. A lo largo del próximo capítulo se describe una propuesta original de diseño de un repositorio semántico de objetos de aprendizaje, orientado a resolver y mejorar los problemas de flexibilidad, heterogeneidad, comunicación e interoperabilidad y tratamiento automático de los metadatos albergados en un repositorio de objetos de aprendizaje.

Capítulo 4

Diseño de la solución: Repositorio Semántico de Objetos de Aprendizaje

Words, as every one now knows, "mean" nothing by themselves, although the belief that they did, was once equally universal. It is only when a thinker makes use of them that they stand for anything, or, in one sense, have 'meaning.'

C. K. Ogden & I. A. Richards

Las definiciones discutidas en el capítulo anterior pueden coexistir. De hecho, el análisis de tal diversidad de conceptualizaciones y la taxonomía de las mismas, será la base de un modelo conceptual neutro que debe proporcionar una serie de funcionalidades adaptadas a cada conceptualización particular de "objeto de aprendizaje" y no necesariamente restringidas a una única definición. La flexibilidad de este nuevo esquema nos permitirá almacenar en un repositorio cualquier tipo de información sobre un objeto de aprendizaje (normalizada o no), y si se desea (lo cual depende de la concepción particular del creador) con la especificación del propósito educativo del mismo.

Este enfoque puede servir como pilar de un nuevo modelo de repositorio de objetos de aprendizaje, diseñado según un sólido modelo semántico que incluya todas las definiciones del análisis de McGreal [McG04] con el objetivo fundamental de soportar los diferentes tipos de objetos de aprendizaje. Los clientes del repositorio (usuario finales, agentes software y sistemas de gestión de contenidos didácticos), podrían, entre otras funcionalidades, añadir, recuperar, modificar y buscar objetos de aprendizaje independientemente de la conceptualización utilizada por el creador de registros de metadatos.

El diseño del repositorio está basado en un conjunto de tecnologías entre las que cabe destacar las ofrecidas por la Web semántica y las ya explicadas en el capítulo 2. Con el objetivo de entender el modelo propuesto, en la primera sección del capítulo se expone una visión global del conjunto de tecnologías ofrecidas por la Web semántica. En la siguiente sección se describe la propuesta del diseño del repositorio semántico de objetos de aprendizaje, incluyendo la representación semántica de metadatos de objetos de aprendizaje, su ámbito de aplicación, y el modelo formal que le da soporte. Seguidamente se propone una primera aproximación de la arquitectura del repositorio, donde se exponen una serie de problemas encontrados que cambian el enfoque inicial planteado orientándolo a la versión final del diseño del repositorio descrita en la última sección.

4.1. Web semántica

La Web semántica es un nuevo paradigma que propone un conjunto de lenguajes, meta-lenguajes, procesos y herramientas enfocado a resolver las deficiencias del modelo actual de Internet. Actualmente, los datos no se encuentran separados de las etiquetas de formato dentro del código de representación de una página web (HTML).

Además, la información no posee ningún mecanismo que facilite entender su significado. Estas restricciones, por parte de las aplicaciones, limitan la capacidad de utilizar de forma automática la información existente en la "*Web*". Tim Berners Lee en [BLHL01] define la Web semántica como:

> *"Una extensión de la Web actual, en la cual se define el significado de la información, permitiendo a los ordenadores y las personas trabajar en cooperación de mejor forma."*

La Web semántica proporciona un marco de trabajo común que permite reutilizar y compartir la información entre aplicaciones, empresas y humanos. Gracias a esta tecnología, es posible programar agentes autónomos con capacidad de extraer, intercambiar y procesar la información existente en Internet. Un ejemplo de la potencia del nuevo software que podría programarse una vez estuvieran maduras estas tecnologías, puede ser un agente de bolsa autónomo con capacidad de consultar la bolsas de todo el mundo sin realizar ningún contrato ni acuerdo entre ellas. Este agente sería capaz de entender los datos existentes en los distintos servidores públicos de la bolsa de cada país, gracias a una representación del conocimiento que define la semántica de la información.

En la figura 4.1 se representa, estructurado en capas, los elementos más importantes que conforman la Web semántica.

Los componentes básicos de la Web semántica son el lenguaje XML y el metalenguaje XML Schema Definition:

- **XML** [1]**:** aporta la sintaxis superficial para los documentos estructurados, pero sin dotarlos de ninguna restricción sobre el significado.

[1] **http://www.w3.org/XML/**

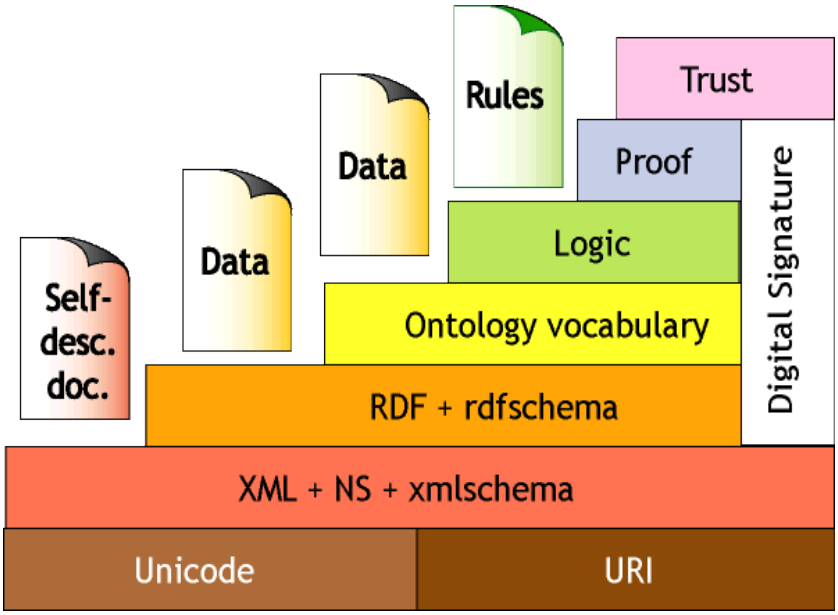

Figura 4.1: Capas de la Web semántica

- **NS** [2]**:** en XML proporciona un método simple de referenciar a un elemento y sus atributos.
- **XML Schema (XSD**[3]**):** es un lenguaje utilizado para definir la estructura de los documentos XML.

En el siguiente nivel se encuentran las especificaciones del vocabulario RDF, y el metalenguaje RDF Schema:

- **RDF**[4]**:** es un lenguaje de propósito general que permite describir la información existente en la web.
- **RDFS**[5]**:** lenguaje creado para la definición de la estructura de los documentos

[2]**XML Namespaces - http://www.w3.org/TR/REC-xml-names/**
[3]**XML Schema Definition Language - http://www.w3.org/XML/Schema**
[4]**Resource Description Framework Specification - http://www.w3.org/RDF/**
[5]**RDF Schema Specification - http://www.w3.org/TR/2000/CR-rdf-schema-20000327/**

RDF. Forma parte de la especificación RDF [W3C04b], a la que pueden añadirse metadatos definidos en otros estándares como Dublin Core [DCM03]. Gracias a los esquemas definidos en RDFS pueden crearse nuevos escenarios que permiten facilitar el trabajo cooperativo entre las aplicaciones y el proceso automático de la información de la Web llevado a cabo por agentes inteligentes. Por ejemplo, puede publicarse en un portal de noticias un fichero XML bajo el esquema RDF RSS[6], para que cualquier otra página web mediante un módulo que interprete RSS pueda recoger el fichero XML que sigue la estructura RSS e interprete y muestre las noticias ofrecidas por el proveedor.

En un nivel superior, "*Ontology Vocabulary*", se encuentran los modelos conceptuales definidos por medio de lenguajes de ontologías[7] como DAML, OIL o OWL. A continuación se describen de forma breve por orden cronológico de aparición:

- **DAML[8]:** evolución de RDF, proporciona un esquema básico para crear una ontología que permita a una máquina realizar inferencias simples.

- **DAML + OIL[9]:** DAML proporciona un marco básico para la descripción de ontologías, combinado con el modelo OIL se aportan la representación y las reglas de inferencia usadas sobre los modelos conceptuales escritos en DAML.

- **OWL[10]:** lenguaje desarrollado para definir una ontología que pueda utilizar un conjunto de aplicaciones que trabajen con conocimiento de un dominio concreto. OWL es un lenguaje que empieza a consolidarse en una buena posición como

[6] http://www.rssboard.org/rss-specification

[7] Aunque se verá en la sección 4.2.2 en detalle, una ontología es una especificación formal de un modelo conceptual compartido de un dominio concreto

[8] **DARPA Agent Markup Language - http://www.daml.org/**

[9] **DAML + OIL Ontology Interface Language: http://www.w3.org/TR/daml+oil-model**

[10] **http://www.w3.org/TR/owl-features/**

estándar en Internet debido al amplio conjunto de herramientas desarrolladas y a la amplia aceptación dentro de las instituciones. Existen múltiples ejemplos de ontologías descritas en OWL, por ejemplo la ofrecida por el proyecto Friend of a Friend FOAF para la descripción de personas y sus relaciones, la ontología del National Cancer Institute (NCI) o la ontología OGC creada para los sistemas GIS de información geográfica.

En la capa "*Logic*" se encuentran un conjunto de especificaciones de lenguajes de reglas y restricciones, y un conjunto de herramientas que implementan un determinado conocimiento procedimental para deducir nuevo conocimiento o verificar la consistencia de una ontología. Sobre los modelos conceptuales pueden definirse reglas lógicas bajo lenguajes como SWRL [W3C04a], que pueden ser procesadas por un motor de inferencia capaz de interpretar el lenguaje de descripción de las reglas y el modelo conceptual sobre el cual han sido definidas. En modelos que poseen definidas restricciones lógicas en los elementos utilizados en la descripción del esquema del modelo, pueden realizarse operaciones que verifiquen la integridad y consistencia del mismo. A continuación se describen de forma breve los elementos que pueden encontrarse en esta capa:

- **Herramientas de verificación del modelo:** sobre el modelo conceptual definido en los lenguajes del nivel anterior, se pueden describir determinadas restricciones lógicas y reglas. Por ejemplo, en el lenguaje OWL se ha definido un vocabulario específico ("*OWL-DL*") para incluir los diferentes tipos de restricciones definidas en la lógica descripciones [BCM$^+$02]. Operaciones de verificación de la consistencia del modelo en función de este tipo de restricciones definidas son llevadas a cabo por aplicaciones llamadas razonadores, como Bossam, Racer, Pellet o FaCET entre otros.

- **Descripción de reglas lógicas - SWRL[11]:** es una propuesta de especificación de un lenguaje formal de reglas lógicas. Este tipo de lenguajes permiten completar el modelo para ser procesado por un motor de inferencia. En este nivel, sobre un modelo conceptual y sus reglas pueden realizarse determinadas consultas a un motor de inferencia y obtener los resultados y la explicación de deducción.

- **Motores de inferencia:** con el conocimiento descrito en base al modelo conceptual definido por el esquema de una ontología y un conjunto de reglas descritas utilizando los elementos de la ontología se pueden ejecutar procesos de inferencia para conseguir deducir nuevo conocimiento o procesar alguna consulta realizada por un humano o aplicación externa. Implementaciones reales de este tipo de sistemas para SWRL son el SWRL Rule Engine del módulo SWRLTAB de la herramienta Protégé, Bossam, Hoolet, Pellet, KAON2, Racer o SweetRules, entre otras herramientas con formatos de reglas propios[12].

Finalmente en la última capa (niveles "*Trust*" y "*Proof*") se encuentran un conjunto de especificaciones orientadas a garantizar las operaciones de autenticidad y veracidad de los documentos creados a partir de las especificaciones descritas anteriormente. Este es el nivel menos consolidado por el momento, intenta solucionar preguntas de este tipo:

> "In stark reality, the simplest way to put it is: if one person says that x is blue, and another says that x is not blue, doesn't the whole Semantic Web fall apart?"

En el nivel "*Proof*" se está trabajando en lenguajes que permitan probar si una sentencia es correcta o no con la información descrita en un sitio web determinado.

[11] **SWRL - http://www.w3.org/Submission/SWRL/**
[12] http://en.wikipedia.org/wiki/Semantic_Reasoner

Para ello se trabaja en la dirección de verificar el contenido completo de un sitio web con diversos ficheros de información, trabajando con "*checksum*" como identificadores unívocos de las aserciones tratadas.

En la figura 4.1 se representa "*Digital Signature*" como elemento vertical común a las capas "*RDF-RDFS*", "*Ontology*", "*Logic*" y "*Proof*", que engloba las especificaciones aportadas por XML Security[13] para el uso de la firma digital en los diferentes documentos creados en la Web semántica.

4.1.1. Evolución

La evolución que ha tenido la Web semántica se muestra en un breve esquema, en la figura 4.2. En esta investigación se emplean técnicas de Web semántica para formalizar un modelo conceptual aceptado a nivel global y con capacidad de ser procesado por cualquier tipo de agente. El uso de estas técnicas permitirá solucionar los problemas de conceptualización existentes en los repositorios de aprendizaje.

4.1.2. Representación del conocimiento en la Web

Uno de los problemas encontrados en el modelo actual de Internet, es la necesidad de expresar el conocimiento humano en un lenguaje capaz de ser procesado por una máquina. Desde el nacimiento del término "inteligencia artificial" en la conferencia de Dartmouth[14] en el año 56 pasando por la época de pesimismo iniciada con el informe ALPAC, se han propuesto multitud de proyectos de interpretación de nuestro

[13]XML Security - http://www.w3.org/2004/Talks/0520-hh-xmlsec/

[14]Reunidos en la conferencia los máximos representantes, Marvin Minsky, John McCarthy, Nathaniel Rochester y Claude Shannon.

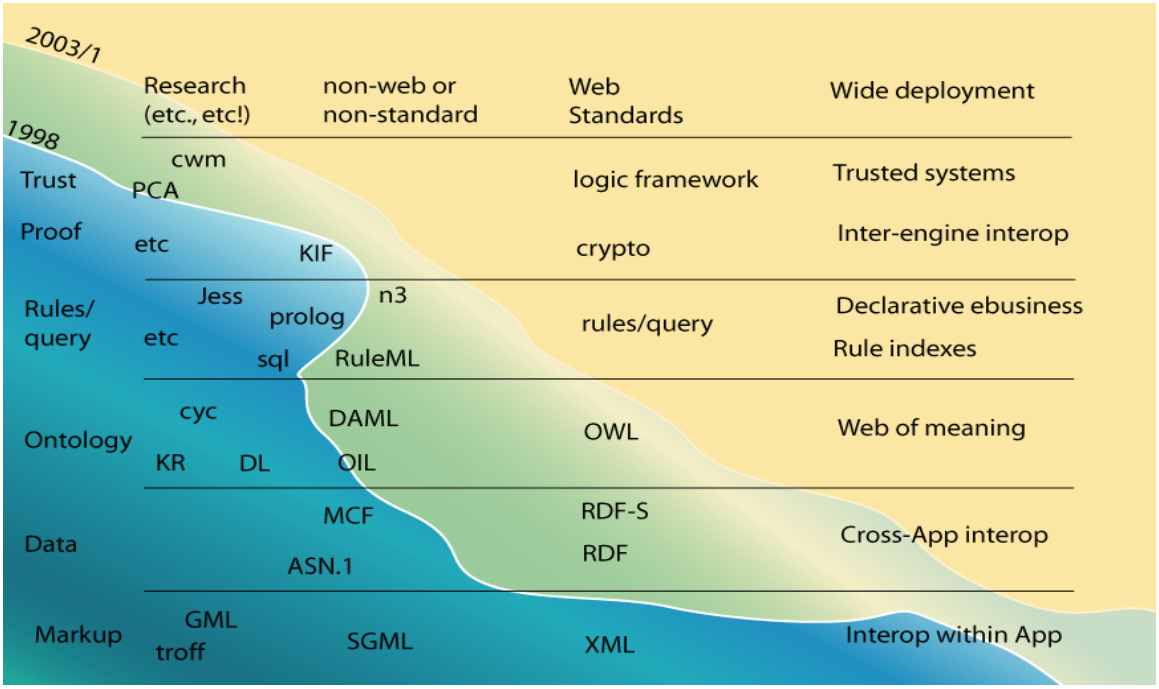

Figura 4.2: Evolución de la Web semántica

lenguaje natural. Los resultados cosechados en los proyectos traductor Ruso-Inglés[15], STUDENT[16] y otros, ayudaron a cambiar el enfoque del problema usando una concepción más reducida del lenguaje bajo una representación formal del mismo. Gracias a estos proyectos y a toda la evolución del pensamiento lógico, existen diferentes métodos de representación formal del conocimiento.

[15]Inicios teóricos de Warren Weaver(1949), pionero en la traducción automática, los problemas de comunicación se podían analizar en tres niveles: técnico, semántico (referido al significado e interpretación del mensaje) y pragmático (sobre las consecuencias de la comunicación en el comportamiento de las personas).

[16]STUDENT - Daniel Bobrow(1965), uno de los primeros Sistemas Expertos Basados en Reglas para la resolución de problemas algebraicos a partir del lenguaje natural. El comentario de Marvin Minsky realizado en el artículo "WHY PEOPLE THINK COMPUTERS CAN'T" explica parte de los problemas de interpretación: "Evita STUDENT los *verdaderos significados* utilizando trucos (más, suma, añade,etc.. ? +) O... ¿es quizá que lo que llamamos significados son en realidad grandes colecciones de trucos? Lo mejor que podemos hacer es ser razonablemente cuidadosos a la hora de interpretar el lenguaje, ya que posee muchas paradojas, dejemos que nuestras máquinas también sean razonablemente cuidadosas."

Marvin Minsky (Director del laboratorio de IA del Instituto Tecnológico de Massachusetts) en su libro "The Society of Mind" [Min86] expresa la complejidad de nuestro ser en distintos planos, en especial el social, así como la complejidad de crear la conciencia mientras no sepamos nosotros mismos lo que es. En su obra "The Emotion Machine" [Min06] intenta formular a nivel abstracto la arquitectura de una máquina con emociones. De todas sus publicaciones cabe destacar especialmente [Min88] la pregunta *¿Por qué la gente piensa que los ordenadores no pueden pensar?*, es una pregunta con una respuesta muy sencilla, porque carecen de sentido común. Empresas como "Cyc" o el proyecto OpenMind (Red de conceptos ConceptNet) abren una nueva línea de investigación sobre el razonamiento con sentido común[17]. Dentro de estos proyectos se han elaborado grandes bases de conocimiento descritas en reglas lógicas (expresividad en lógica mejorada de primer orden) capaces de ser procesadas por cualquier agente software y ampliadas por grupos de usuarios especializados. Estas nuevas ideas se han trasladado al mundo de Internet, en el que gracias a los avances realizados en materia de estándares del nuevo modelo Web, se proponen modelos lógicos de expresividad de conocimiento. El IEEE ha creado un grupo de trabajo denominado SUO[18] dedicado a analizar las grandes bases de conocimiento aportadas por estos proyectos, con el objetivo de proporcionar una ontología de nivel superior que sirva para expresar el conocimiento en Internet.

Los esfuerzos aportados por los grupos de trabajo de la Web semántica han dado origen a nuevas especificaciones de lenguajes capaces de expresar el significado de la información. Este enfoque ha aportado una nueva visión sobre la automatización de procesos compartiendo el conocimiento expresado en la web. El objetivo para el

[17]En inglés "Common Sense Reasoning"
[18]http://suo.ieee.org/

nuevo Internet, no es solo mostrar información sino compartirla y procesarla de forma automática.

A lo largo de los siguientes apartados se explican de forma breve los conceptos fundamentales utilizados en la investigación. En primer lugar se introducen los diferentes tipos de conocimiento que pueden formalizarse, a continuación se explica la lógica de descripciones como modelo elegido por la especificación del lenguaje de ontologías y finalmente se describe el concepto de ontología utilizado en la creación de los modelos conceptuales de la Web semántica.

Tipos de Conocimiento

- Conocimiento declarativo: utilizado para expresar los "hechos" y las "reglas lógicas" de un universo del discurso[19]. Ejemplo: "La Almudena es una catedral" o "Todas las catedrales son edificios religiosos". Hay que entender que un ordenador no conoce nada si no se lo expresamos.

- Conocimiento procedimental: define los procesos de razonamiento que permiten razonar y generar nuevo conocimiento a partir del ya existente. Plasmado en un ejemplo, este tipo de conocimiento representa el proceso que me permite obtener el conocimiento inferido "La Almudena es un edificio religioso" a partir de los ejemplos expuestos.

Lógica de Descripciones

Las *lógicas de descripciones* [BCM+02] constituyen una familia de formalismos lógicos para la representación de conocimiento. Basadas en la lógica de marcos y en las redes semánticas, se utilizan para representar el conocimiento de un dominio de

[19]El concepto lógico "Universo del Discurso" es una abstracción utilizada para delimitar el campo de operación de los algoritmos de inferencia.

aplicación mediante la definición de los conceptos más relevantes, las relaciones entre los mismos y las propiedades de los individuos. Dentro de las lógicas de descripciones, una de las más representativas es el cuerpo $\mathcal{ALC}$, utilizado en el presente trabajo.

Las representaciones construidas con lógicas de descripciones permiten llevar a cabo razonamientos automáticos inferidos a partir del conocimiento representado explícitamente en un modelo. De igual modo soportan la inferencia de patrones, lo que permite estructurar y comprender el dominio de aplicación mediante la clasificación de conceptos e individuos, característica muy utilizada en sistemas inteligentes de procesamiento de información como los basados en ontologías.

Los sistemas basados en lógica de descripciones están formados por dos componentes básicos, *TBox* y *ABox*, tal y como se muestra en la Figura 4.3. Estos dos componentes comprenden la terminología (vocabulario del dominio de la aplicación) y las aserciones (individuos definidos según el vocabulario), respectivamente. La representación del dominio se lleva a cabo mediante la clasificación de conceptos, lo que determina las relaciones jerárquicas concepto/superconcepto dentro de una terminología. Por otra parte, la clasificación de individuos permite determinar si un individuo es una instancia de un concepto, tomando como base la descripción del individuo y la definición del concepto. Mediante las relaciones entre instancias será posible la aplicación de reglas que inserten hechos adicionales en la base de conocimiento.

Como cualquier otra lógica de descripciones, $\mathcal{ALC}$ –la lógica de descripciones proposicionalmente cerrada más reducida– basa su semántica en la utilización de conceptos, roles e individuos. Los *conceptos* (que habitualmente se representan con letras mayúsculas) son expresiones que reflejan las propiedades, descritas mediante roles, de un conjunto de individuos. Pueden ser de dos tipos: primitivos o construidos. Los

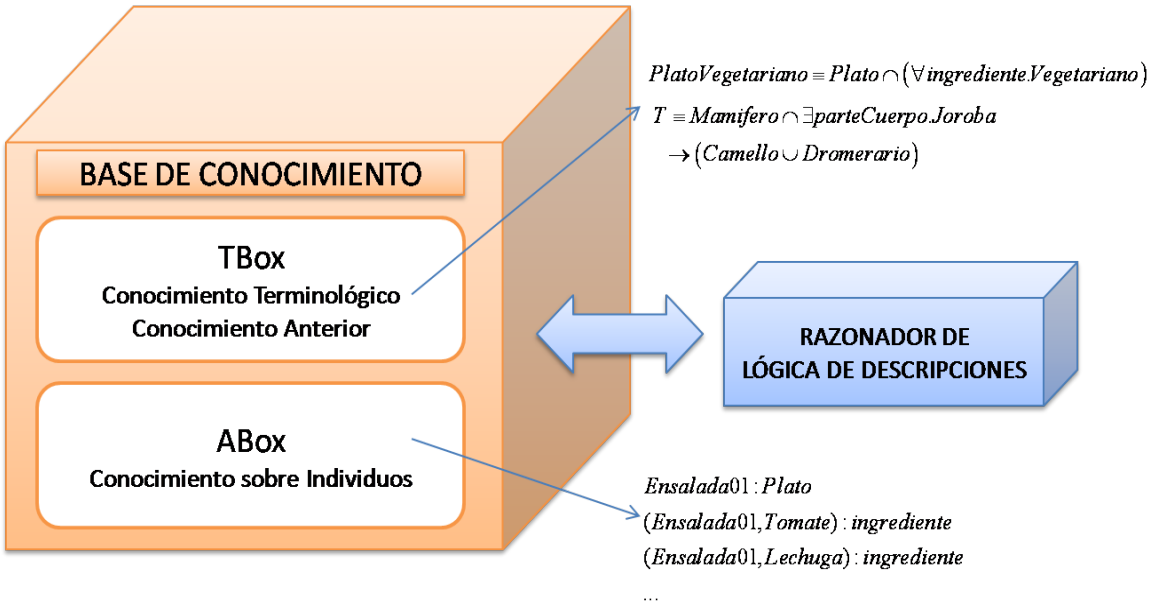

Figura 4.3: Arquitectura de un sistema de representación del conocimiento basado en lógica de descripciones

siguientes son conceptos primitivos:

$$\top \mid \text{(concepto universal)}$$
$$\bot \mid \text{(concepto nulo)}$$
$$A \mid \text{(concepto primitivo)}$$

Un *axioma terminológico* permite definir un nuevo concepto o especializar un concepto existente. Tanto los roles como los conceptos se pueden combinar mediante constructores para formar descripciones complejas, que conforman axiomas en el *TBox*. Para definir nuevos conceptos se utilizan los siguientes constructores:

$$C \sqcap D \mid \text{(conjunción de conceptos)}$$
$$C \sqcup D \mid \text{(unión de conceptos)}$$

$$\begin{aligned} \neg\ C &\mid \text{(negación de conceptos)} \\ \forall\ R.C &\mid \text{(cuantificación universal)} \\ \exists\ R.C &\mid \text{(cuantificación existencial)} \end{aligned}$$

donde C y D son conceptos y R es un rol.

La definición de conceptos se denota como $A \equiv C$, donde generalmente A es el nombre del concepto que está siendo definido y C un concepto complejo. Por ejemplo:

$$Elefante \equiv Mamífero \sqcap \exists\, parteDelCuerpo.Trompa \sqcap \forall\, color.Gris$$

La definición mediante especialización explícita se denota utilizando el símbolo de subsunción $A \sqsubseteq C$, donde A representa un concepto más específico que el concepto C:

$$Elefante \sqsubseteq Mamífero$$

Una *aserción* permite declarar individuos (que habitualmente se representan con letras minúsculas) como instancias de un concepto. También es posible establecer mediante aserciones que un par de individuos son una instancia de un determinado rol, y que por tanto se satisface la propiedad representada por el rol para ellos. La notación es la siguiente:

$$\begin{aligned} a : C &\mid (\ a \text{ es una instancia de } C) \\ (a, b) : R &\mid (\ (a, b) \text{ es una instancia de } R) \end{aligned}$$

También es muy habitual la siguiente notación:

$$\begin{aligned} C(a) &\mid \text{(Para representar la instanciación de un concepto)} \\ R(a, b) &\mid \text{(Para representar la instanciación de un rol)} \end{aligned}$$

Un conjunto finito de aserciones y axiomas terminológicos da lugar a una base de conocimiento. Dentro de cada base de conocimiento, como se ilustra en la Figura 4.3, el $TBox$ contiene las definiciones de conceptos y los axiomas que conforman la terminología o vocabulario del dominio de la aplicación, mientras que el $ABox$ contiene el conjunto de aserciones sobre conceptos y sobre roles que representan los individuos definidos según el vocabulario.

Inferencia y Razonamiento

Las aserciones, los axiomas de especialización y los axiomas de definición permiten la inferencia en la base de conocimiento, proporcionando mecanismos para la subsunción ($K \sqsubseteq P$) y la equivalencia ($K \equiv P$) de conceptos. Este tipo de relaciones proporcionan la información necesaria para conectar conceptos y así facilitar los servicios de inferencia.

Sobre una base de conocimiento que contiene un lenguaje descriptivo de un dominio concreto $\mathcal{ALC}$, si se define un nuevo concepto es necesario conocer si es consistente o contradictorio con el $TBox$. Esta propiedad se conoce como el concepto satisfacible (o respectivamente insatisfacible) con respecto al $TBox$. También puede ser necesario saber si un concepto es más general que otro, si son equivalentes o si son disjuntos. La formalización de estas propiedades es la siguiente, supongamos que $\urcorner$ es un $TBox$, C y D conceptos:

- C es satisfactible respecto a $\urcorner$ si existe un modelo I tal que $T(C)^I \neq \emptyset$.
- C es subsumido por P respecto a $\urcorner$ si para todo modelo I de $\urcorner$,$(C)^I \subseteq (D)^I$. Se escribe $\urcorner \models C$.

Al definir un $ABox$ referente a un $TBox$, las propiedades más importantes a verificar son las de la consistencia y la derivación de una instanciación a partir del $ABox$. Formalmente: Supongamos que $\urcorner$ es un $TBox$, A es un ABox, C un concepto y z un nombre de individuo:

- A es consistente con respecto a $\urcorner$ si existe una interpretación que es modelo de $\urcorner$ y de A.
- $z : C$ se deriva de $\urcorner$ y A si todo modelo I de $\urcorner$ cumple $(z)^I \in (C)^I(\urcorner, A \models z : C)$

Ontologías

La palabra ontología según el D.R.A.E. significa:

> *Parte de la metafísica que trata del ser en general y de sus propiedades trascendentales.*

Como puede apreciarse la definición aportada es demasiado general, no se han incluido las múltiples acepciones adoptadas por este término. Guarino y Giaretta en [GG95] proporcionan un interesante análisis sobre el termino ontología, incluyendo varias interpretaciones desde el punto de vista filosófico hasta las diferentes caracterizaciones dadas en la ingeniería del conocimiento. La interpretación del término dada por el D.R.A.E es la utilizada en el dominio de la filosofía, esta interpretación trata las cuestiones fundamentales del "*ser*" y las características y relaciones comunes dadas en la naturaleza física y metafísica de todo "*ser*".

Raul Corazzon aporta una definición respecto al concepto "*ontología formal*". En el estudio [Cor05] se describe el término como una conceptualización compartida de un dominio concreto, que proporciona criterios de distinción de los diferentes tipos de objetos (concretos y abstractos, existentes y no existentes, reales e ideales, independientes y dependientes) así como sus vínculos (relaciones, dependencias y predicados).

En el dominio de la ingeniería del conocimiento, la definición más aceptada del término ontología es la de Gruber [Gru93]:

"Una ontología es una especificación de una conceptualización"

Extendiendo la definición de Gruber a dominios particulares del conocimiento, una ontología puede entenderse como una especificación formal de una conceptualización compartida de un dominio concreto. Esta conceptualización ha de ser:

- Comprensible por las máquinas.
- Comprensible entre un grupo de personas que poseen conocimientos de un dominio concreto.
- Basada en conceptos.
- Definida con descripciones generales de conceptos incluyendo todas las restricciones, relaciones e individuales del dominio tratado.

Las ontologías se emplean en todo tipo de aplicaciones informáticas en las que resulta necesario definir concretamente el conjunto de entidades relevantes en el campo de aplicación determinado, así como las interacciones entre las mismas. Una ontología puede utilizarse para:

- Compartir estructuras de información comunes entre diversos agentes (humanos / software).
- Reutilizar el conocimiento de un dominio concreto.
- Hacer suposiciones explícitas sobre un dominio concreto.

- Separar el dominio de conocimiento del conocimiento procedimental.
- Analizar el conocimiento del dominio.

La estructura de una ontología incluye: conceptos (términos); propiedades que describen las clases y subclases; restricciones o características de una propiedad (tipo de valores que adoptarán, cantidad, etc.); taxonomía de los conceptos o relaciones formales entre conceptos; instancias, que representan objetos determinados dentro de un concepto y axiomas, que permiten inferir mediante reglas el conocimiento que no está explícitamente indicado en la taxonomía de conceptos.

En términos prácticos, el desarrollo de una ontología es un proceso creativo que consiste en una secuencia de acciones básicas [NM01]:

1. Definir las clases de la ontología.
2. Establecer una taxonomía entre los términos.
3. Definir las propiedades y describir los valores permitidos para cada una de ellas.
4. Definir las instancias individuales de la clase definida.

El proceso de conceptualización requiere de una aceptación consensuada por un conjunto significativo de personas con conocimientos expecializados de un dominio concreto. Este proceso puede beneficiarse del modo de construcción iterativo e incremental aportado por las metodologías modernas de construcción del software [JBR99]. La creación de una ontología siguiendo los pasos aportados por Noy y McGuinness [NM01] con un modelo iterativo de construcción dentro de un dominio global, debe por tanto seguir el siguiente esquema:

1. Definir las clases de la ontología.

Elemento	Definición	Ejemplo
Clase	Describe conceptos del universo del discurso. Se refiere a una colección de objetos individuales. Tipos de clases: ▪ Clases hermanas: aquellas que están en el mismo nivel de generalización. ▪ Clases enumeradas: formadas por un conjunto cerrado de instancias. Continente es la clase de África, Asia, Europa, América y Oceanía. ▪ Clases disjuntas: aquellas que no tienen instancias en común. (ej: Vino blanco y Vino tinto)	*Animal*. *Persona*: deriva de *Animal*.

Propiedad	Describe una relación entre un tipo de objetos individuales. Los objetos en el modelo conceptual pertenecen a clases y son relacionados con otros objetos o valores mediante propiedades.	`tiene_mascota`
		`  Dominio: Persona`
		`  Rango: Animal`
		
		`es_mascota_de`
		` Inversa de:`
		`  tiene_mascota`

Individual	Es un objeto que pertenece a una instancia de una clase.	El individual Jesús es una instancia de la clase Profesor.
Restricciones	Utilizadas en la descripción de conceptos. Delimitan el universo del discurso. Las restricciones son expresadas en lógica descriptiva (subconjunto de la lógica de predicados).	$PlatoVegetariano$: $\forall tieneIngrediente\ (IngredienteVegetal)$

Tabla 4.1: Elementos principales utilizados en la definición de una ontología

2. Establecer una taxonomía entre los términos.

3. Definir las propiedades y describir los valores permitidos para cada una de ellas.

4. Definir las instancias individuales de una clase.

5. Si aún no está bien definida la ontología, iterar desde el punto 1 hasta el punto 4.

6. Publicar una versión de a ontología.

7. Esperar aceptación y mejoras del resto de la gente sobre nuestra ontología.

Una ontología puede representarse en diferentes lenguajes formales. Clasificados por la estructura del modelo conceptual, podemos encontrar lenguajes de expresión basados en marcos como OIL, F-LOGIC, OKBC o KM. La representación de ontologías basadas en un modelo conceptual de lógica de descripciones, podemos realizarla con el lenguaje KL-ONE o OWL. Por otro lado los lenguajes que permiten la definición de ontologías basadas en modelos conceptuales que requieren un mayor nivel de expresividad, como base tienen en cuenta el modelo y cálculo de predicados aportado por la lógica de primer orden, ejemplos de estos lenguajes son KIF o CycL.

Cada conceptualización de un dominio concreto incluye en algunos casos la definición de relaciones con conceptos generales. Al no poseer un modelo conceptual común del conocimiento general humano, en el proceso de creación de una ontología pueden incluirse interpretaciones particulares de un subconjunto de términos, relaciones y reglas del conocimiento general humano, por ejemplo “El Seat Ibiza es un Coche” (es_un SeatIbiza Coche). Esta situación genera problemas de interpretación por parte de los agentes inteligentes que deseen realizar determinadas inferencias con los conceptos de dominio general. Para solucionar esta situación, el conocimiento general humano

puede quedar definido en una ontología con el apoyo de un organismo público de entidad internacional. Este es el caso de las ontologías de alto nivel (OpenCyc, SUO IIF, SUO 4D, SUMO) propuestas al grupo de trabajo SUO del IEEE.

4.1.3. Ontologías de nivel superior

Son bases de conocimiento que incluyen la definición de conceptos, relaciones, propiedades, restricciones e individuales, así como la capacidad de razonamiento sobre estos elementos. Las ontologías de nivel superior[20] poseen conocimiento "común" o "general" (commonsense), y por tanto no se limitan a conocimiento específico. Una ontología particular puede definir un conocimiento específico relacionado con el conocimiento general establecido en una ontología de nivel superior.

El grupo de trabajo de las ontologías de nivel superior del IEEE, SUO[21], está estudiando varias propuestas:

- **OpenCyc**: es la ontología elegida en este proyecto de investigación. Se detalla en la sección 4.1.4.
- **IFF**[22]: codificado en SUO KIF sobre una arquitectura de tres metaniveles (Top, Upper, Lower). Proporciona la clave de integración entre diferentes ontologías, definiendo varias interfaces sobre un metamodelo conceptual (accesible desde OWL, CyCL o SUO KIF, por ejemplo).
- **SUO 4D**[23]: facilita la integración de modelos (según ISO/FDIS 15926-2), basada en la reutilización de conceptos.
- **SUMO**[24]: posee una amplia jerarquía de conceptos con 654 términos y 2351

[20]Del inglés: Upper Ontologies
[21]Standard Upper Ontology Working Group (SUO WG) - http://suo.ieee.org/
[22]IFF - http://suo.ieee.org/IFF/metalevel/lower/ontology/ontology/version20021205.htm
[23]SUO 4D - http://suo.ieee.org/SUO/SUO-4D/index.html
[24]SUMO - http://www.ontologyportal.org/

afirmaciones.

Uno de los proyectos más importantes sobre la ingeniería del conocimiento es el llevado a cabo por la empresa Cyc [25], fundada en 1994 para la investigación, desarrollo y comercialización de la inteligencia artificial. Cyc [Len91] es una de las mayores bases de conocimiento existentes, cuyo objetivo es la construcción de software inteligente gracias al sentido común aportado por una gran base de conocimiento que posee hechos, reglas y heurísticas para el razonamiento sobre los objetos y eventos de la vida diaria [Len95].

4.1.4. Tecnologías y lenguajes

A continuación se describen las principales tecnologías y lenguajes existentes utilizados en el diseño y construcción de la nueva Web.

Marco de Trabajo para la descripción de Recursos (RDF)

RDF es un lenguaje de propósito general utilizado para representar información. Posee las siguientes características:

- Define un modelo de metadatos.
- Destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web.
- Utiliza XML para intercambiar descripciones de recursos Web, pero los recursos descritos pueden ser de cualquier tipo, incluyendo recursos XML y no-XML.

El modelo de datos básico consta de tres tipos de objetos:

- Recursos: todas las cosas descritas por expresiones RDF se denominan recursos. Un recurso puede ser una página Web completa, parte de una página Web, una

[25] Cyc, Inc. - http://www.cyc.com/

colección completa de páginas o un objeto que no sea directamente accesible vía Web. Los recursos se designan siempre por URIs.

- Propiedades: una propiedad es un aspecto específico, característica, atributo, o relación utilizado para describir un recurso.
- Sentencias: una sentencia está compuesta de un sujeto, un predicado y un objeto.

Un modelo RDF es una colección desordenada de sentencias. Puede representarse en un grafo dirigido expresando una estructura de relaciones entre nodos. Los sujetos y los objetos son nodos, mientras que los predicados son los arcos (ver figura 4.4). El objeto de una sentencia (el valor de la propiedad) puede ser otro recurso o puede ser un literal; es decir, un recurso (especificado por un URI) o una cadena de caracteres u otros tipos de datos primitivos definidos por XML.

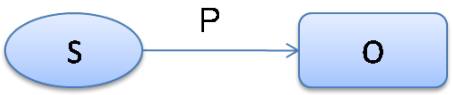

Figura 4.4: Esquema de una sentencia RDF

Sujeto--→ Predicado--→ Objeto
Jesus--→ (ha escrito)--→ portales XML.

RDF utiliza URI's para la identificación de los elementos utilizados en la construcción del modelo. El ejemplo anterior quedaría representado según la figura 4.5, donde "dc:" indica que se está utilizando el "Dublin Core Namespace".

Un modelo RDF puede representarse de diversas formas, entre las que cabe destacar aquellas representaciones que han sido ya implementadas por alguna herramienta

Figura 4.5: Sentencia RDF

o las que sigan una determinada especificación del W3C. En esta investigación se ha trabajado con las representaciones de grafos almacenadas en memoria y la especificación RDF-XML [W3C04b] o N3 [BL98] para almacenar de forma estándar los modelos RDF en soportes digitales. El ejemplo anterior serializado a XML posee la siguiente forma:

Listado 4.1: Ejemplo serialización RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description
      rdf:about="http://www.miservidor.com/PortalesXML.html">
      <dc:title>CONSTRUCCIÓN DE PORTALES WEB XML</dc:title>
      <dc:creator rdf:resource="urn:jesus.soto:upsam.net"/>
    </rdf:Description>
</rdf:RDF>
```

Como se puede ver, la serialización a XML de RDF [W3C04b] es poco intuitiva, por lo que no es posible leerla directamente. En el ejemplo anterior, <rdf:Description> no es una descripción, sino el inicio de una sentencia RDF.<dc:title> es una propiedad, que indica que el recurso descrito tiene un título. Todas las URI's deben ser especificadas como atributos <rdf:resource>. En cambio, existe otra forma de representar un modelo RDF con un lenguaje más fácil de interpretar por un humano, el lenguaje [BL98] de Tim Berners-Lee.

```
@prefix  dc:   <http://purl.org/dc/elements/1.1/> .
<http://www.miservidor.com/PortalesXML.html>
      dc:title        "CONSTRUCCIÓN DE PORTALES WEB XML"
      dc:creator      <urn:jesus.soto:upsam.net> .
```

Para que un agente software entienda N3, es necesario generar una transformación a RDF. Para ello existen programas como Jena (Java) o CWM (Python), pero procesos de traducción generan costes de rendimiento. Por ello, es recomendable trabajar con XML para el tratamiento automático de la información.

Como ejemplos útiles de aplicación de RDF cabe destacar *RDF Site Summary (RSS)* [26] utilizado para obtener resúmenes de los sitios web o *FOAF (Friend of a Friend)* (FOAF)[27] utilizado para la descripción de una persona y relaciones existentes con otras.

RDF es el pilar básico del modelo formal aportado en esta investigación. Con RDF sólo podemos definir relaciones semánticas, por ello, para definir los conceptos globales utilizados en la descripción de recursos, es necesario un lenguaje más formal. En la siguiente sección se presentan las características de OWL como lenguaje para el modelado de ontologías.

SPARQL

SPARQL es un lenguaje de consultas para documentos RDF que mantiene una cierta similitud con el lenguaje de consultas SQL para bases de datos. Gracias a esta iniciativa plasmada en el estándar [W3C07] la comunidad informática ha implementado múltiples bibliotecas de programación (Protégé search,ARQ Jena o Rascal) escritas en diversos lenguajes para el análisis y ejecución de consultas representadas en el lenguaje SPARQL sobre modelos RDF.

[26]RDF Site Summary (RSS) - http://web.resource.org/rss/1.0/spec
[27]Friend of a Friend (FOAF) - http://www-128.ibm.com/developerworks/xml/library/x-foaf.html

A continuación se expone una serie de ejemplos de búsqueda con SPARQL. En la primera consulta (listado de código 4.2), a partir de un archivo FOAF se obtiene el nombre y el e-mail de cada uno de los elementos individuales almacenados.

Listado 4.2: Consulta SPARQL realizada a un fichero FOAF

```
1  PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
2  SELECT ?name ?mbox WHERE{
3       ?x foaf:name ?name .
4       ?x foaf:mbox ?mbox
5  }
```

En el siguiente ejemplo (listado de código 4.3), se busca un recurso de tipo 'mpeg7:VideoType'cuyo creador es Álejandro Amenábar'.

Listado 4.3: Consulta SPARQL utilizando la ontología MPEG7

```
1  PREFIX mpeg7: http://rhizomik.upf.edu/ontologies/2005/03/Mpeg7
      -2001.owl
2  SELECT ?resource WHERE{
3         ?track rdf:type mpeg7:VideoType .
4         ?track mpeg7:Creator ?Author .
5         ?Author mpeg7:Name "Alejandro Amenábar" .
6         ?track mpeg7:uri ?resource .
7  }
```

El lenguaje permite realizar consultas sobre ficheros RDF, aunque también es posible realizar consultas sobre ontologías escritas con OWL. En este tipo de lenguaje es posible describir conceptos con un nivel alto de detalle (ver siguiente sección). El lenguaje SPARQL no permite realizar búsquedas con inferencias sencillas sobre conocimiento descrito con ontologías en OWL. Por ejemplo, si deseamos obtener todos los edificios religiosos de una ciudad, con una consulta SPARQL únicamente obtendríamos los datos asociados "directamente" a una propiedad, por lo que si existe una clara taxonomía entre los diferentes edificios religiosos (iglesias, catedrales, capillas, etc..) estos no serán tenidos en cuenta por SPARQL. En el artículo [SGRJ06a] se presenta un método de búsqueda de recursos con SPARQL a lo largo de múltiples

servidores multimedia.

Lenguaje de Ontologías Web (OWL)

OWL es un lenguaje de marcado para la publicación de ontologías en la WWW cuyo objetivo es facilitar un modelo construido sobre RDF y codificado en XML, que permita representar ontologías a partir de un vocabulario más amplio y una sintaxis más fuerte que la que permite RDF. Es utilizado para representar de forma explícita el significado de términos pertenecientes a un vocabulario y definir las relaciones que existen entre ellos. OWL tiene como punto de partida las experiencias previas realizadas con DAML-OIL.

OWL se divide en tres sublenguajes, OWL-Lite, OWL-DL y OWL-Full, cada uno de los cuales proporciona un conjunto definido sobre el que trabajar, siendo el más simple OWL-Lite y el más completo OWL-Full.

- OWL-Lite: expresa relaciones simples. (Clases / Subclases / Propiedades)
- OWL-DL: basado en lógica descriptiva. Permite el razonamiento automático.
- OWL-Full: mayor grado de expresividad. Definición de metaclases.

Una ontología descrita en OWL es un conjunto de información sobre clases y propiedades. Desde el punto de vista de nuestra investigación, necesitaremos demostrar la compleción del modelo conceptual del repositorio, verificando que todos los términos han sido definidos correctamente. El grado de expresividad de nuestro modelo sufre un compromiso fuerte con la computabilidad del mismo. Un modelo descrito completamente con OWL-Full aporta todas las definiciones e inferencias realizadas sobre el mismo, pues este es el mayor grado de expresividad aportado en la especificación de OWL. El problema de cara a nuestra investigación, es que OWL-Full requiere una capacidad de proceso demasiado elevada y por tanto innecesaria en el caso presente, ya que no necesitamos la compleción de todas las descripciones, muchas de

ellas pueden ser inferidas. Por otro lado, si utilizamos el menor grado de expresividad podemos encontrarnos con carencias de cara a la inferencia sobre el sistema, pues ni siquiera datos obvios no pueden ser inferidos dentro del modelo formal. Por ello, la especificación que cumple el grado de expresividad adecuado frente a los requisitos de computación es OWL-DL.

Bajo la semántica de OWL-DL es posible realizar demostraciones de consistencia y compleción. En las clases y propiedades se definen características y restricciones mediante la lógica descriptiva.

OpenCyc

OpenCyc[28] es la versión de código abierto de la base de conocimiento Cyc. OpenCyc representa el conocimiento general humano [WMB+05], contiene cientos de miles de términos y varios millones de aserciones sobre las relaciones de los mismos, un motor de inferencia, un navegador de la base de conocimiento y otras herramientas útiles.

La principal ventaja de OpenCyc sobre otros sistemas de representación del conocimiento, es el uso de un lenguaje formal "*CycL*" en el cual las conexiones entre conceptos y declaraciones están codificadas para ser procesadas por una máquina. El contenido de la base de conocimiento comprende una vasta taxonomía de conceptos y relaciones, además de una representación formal de las interconexiones. La figura 4.6, muestra las capas existentes en una base de conocimiento según OpenCyc:

- Capa 0 - Ontología de nivel superior (OpenCyc): representa las relaciones existentes entre los conceptos generales.
- Capa 1 - Teorías generales: son las teorías utilizadas en el proceso de razonamiento abstracto. Representan hechos generales sobre el espacio, el tiempo y la

[28]General knowledge base and commonsense reasoning engine - http://www.opencyc.org

causalidad.

- Capa 2 - Teorías específicas: definen los conceptos y sus relaciones en un dominio concreto.
- Base de la pirámide - Hechos: información concreta de los individuales definidos en una teoría específica.

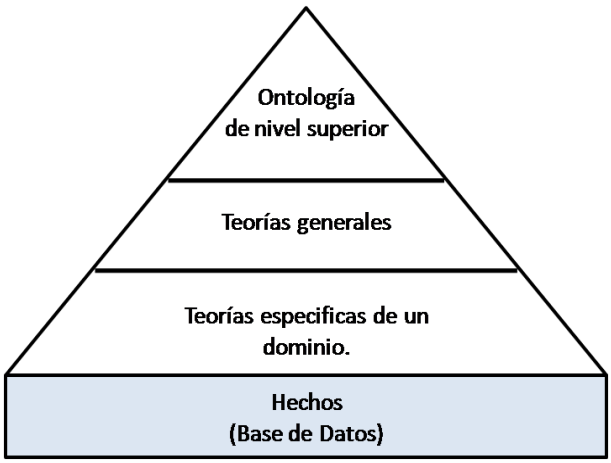

Figura 4.6: Modelo de Capas OpenCyc para una base de conocimiento

Los tipos de expresiones fundamentales existentes dentro de OpenCyc son los siguientes:

Constantes: hacen referencia a individuales, colecciones o colecciones de colecciones.

`#$GeorgeWBush y #$Sudan`

Son constantes que referencian a un individual específico.

`#$WorldLeader y  #$Country`

Son constantes que referencian colecciones.

`#$WineTypeByColor`

Es una constante que referencia a un grupo de colecciones de vinos (blanco, rosado, tinto).

Funciones: Retornan un resultado según los argumentos de entrada. Ej:

`#$PresidentFn`

Necesita un país como argumento y retorna como resultado el nombre de presidente del país. Ejemplo:

```
(#$PresidentFn #$Mexico) --> #$Vicente Fox
```

Relaciones: En CycL nos encontramos con los siguientes términos básicos para crear relaciones:

- `#$isa`

  Es el término más básico de CycL. Es utilizado para decir que algo es parte de una colección. Todos los conceptos almacenados pertenecen al menos a una colección.

  `(#$isa #$Cat #$BiologicalSpecies)`

  Expresa que los gatos son parte de las especies biológicas.

  `(#$isa #$Spain #$WesternEuropeanCountry )`

  Expresa que España es un país del oeste de Europa.

- `#$genls`

  Este termino es utilizado para decir que una colección es una subcolección de otra.

```
(#$genls #$Lion #$Carnivore)
```

Esta expresión dice que todos los leones son carnívoros.

- CycL posee multitud de términos relacionales definidos:

```
#$biologicalRelatives
       (#$biologicalRelatives #$JerryLeeLewis #$JimmySwaggart)

#$geographicalSubregions
      (#$geographicalSubregions #$UnitedStates #$Utah-State)

#$greaterThan
       (#$greaterThan 25 3)

#$orbits
      (#$orbits #$MoonOfEarth #$PlanetEarth)

#$authorOfLiteraryWork-CW
       (#$authorOfLiteraryWork-CW #$HermanMelville #$MobyDickNovel)
```

- En CycL se pueden definir predicados como en los lenguajes lógicos conocidos.

```
(#$arity #$biologicalMother 2)
    (#$arg1Isa #$biologicalMother #$Animal)
    (#$arg2Isa #$biologicalMother #$FemaleAnimal)
```

- También se pueden crear expresiones lógicas complejas que permiten establecer axiomas dentro de las semántica de las relaciones. El uso de lógicas conectivas y cuantificadores proporciona un buen nivel de expresividad

lógico. Ejemplo:

```
(#$forAll ?COUNTRY
    (#$forAll ?PERSON
       (#$implies
            (#$and
              (#$isa ?COUNTRY #$Superpower)
              (#$headsGovernment ?COUNTRY ?PERSON))
        (#$hasStatus ?PERSON #$WorldLeader))))
```

Expresa que para todos los paises (COUNTRY) y todas las personas (PERSON), si un país es una superpotencia (Superpower), entonces la persona cabeza de gobierno de ese país tiene el estado (status) de lider mundial (WorldLeader).

En la figura 4.7 se muestran algunas colecciones de las teorías generales. Puede denotarse una verdad universal, representada en diversos ensayos filosóficos. Todo lo que conocemos en nuestro universo son cosas, incluso las intangibles, por ello en OpenCyc se define Thing como la colección suprema de la base de conocimiento.

Dentro del marco de la descripción de recursos, OpenCyc juega un papel importante. La información almacenada en los registros de metadatos de un repositorio de objetos de aprendizaje (como hemos descrito en los problemas del capítulo anterior) no posee un modelo formal que dé soporte a comprender por parte del análisis de una máquina el significado del mismo. Gracias a la base de conocimiento OpenCyc será posible describir objetos de aprendizaje e interpretar esas descripciones. Así por ejemplo, si aportamos a nuestro repositorio una serie de cuadros de la época barroca podremos realizar búsquedas tan potentes como el obtener “aquellos cuadros

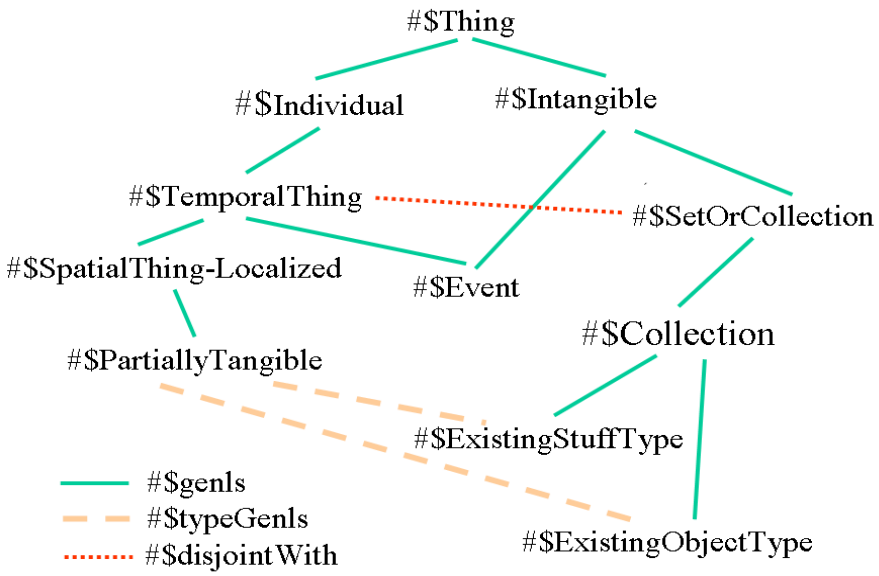

Figura 4.7: Algunas colecciones abstractas de OpenCyc

del periodo del renacimiento, cuyo autor sea español y sus creaciones contengan una manzana dentro de un jarrón". No es difícil interpretar y generar una búsqueda de esta forma, ya que OpenCyc posee relaciones y propiedades espaciales (en la figura 4.8 , se muestran algunas acepciones posibles del término "*in*"). Respecto a los periodos históricos, existe la definición del renacimiento "*TheRenaissance*" dentro de OpenCyc.

Dotar de significado semántico a los metadatos albergados en el repositorio será uno de los objetivos de esta investigación. La formalización de los datos en el lenguaje CycL es un proceso costoso, ni con herramientas avanzadas sería posible insertar unos datos con la facilidad que nosotros tenemos de expresar nuestro lenguaje.

Si se desea dotar de sentido común a las descripciones dadas en un modelo formal, es necesario operar con este tipo de ontologías de nivel superior. OpenCyc, al poseer

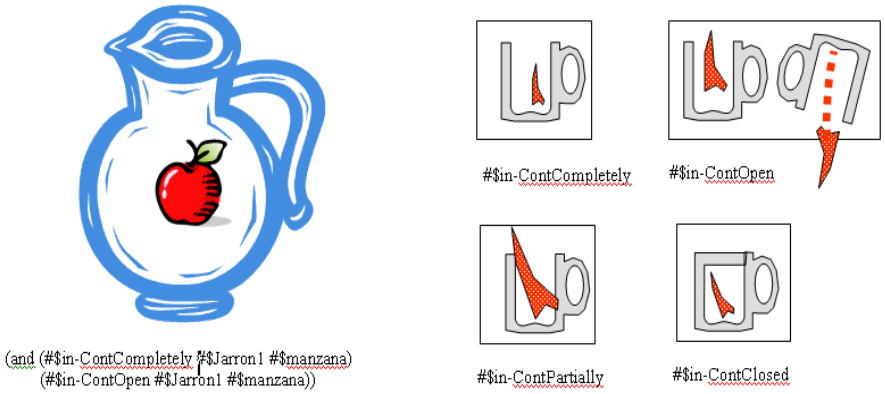

Figura 4.8: Sentidos posibles del concepto “in” en OpenCyc

parte de la estructura de unas de las mejores bases de conocimiento general del mundo, es fuerte competidor dentro de las propuestas del grupo de trabajo SUO del IEEE. Si se cumplen los objetivos propuestos, este grupo de trabajo dejará la mejor definición existente de una ontología superior como propuesta a la comunidad de desarrolladores de Internet e inteligencia artificial, para que posteriormente se creen nuevos modelos conceptuales de dominios concretos que hagan uso de un conocimiento común expresado en este tipo de ontología. Las nuevas aplicaciones que se desarrollen a partir de estos modelos podrán tener la capacidad de interpretar la información existente en Internet bajo un marco común conceptual que permita realizar razonamientos de ”sentido común”, similares a los que nosotros realizamos.

4.1.5. Herramientas

Actualmente existen multitud de herramientas para manejar las especificaciones de la Web semántica, pero lamentablemente muchas de ellas aún son prototipos lanzados por diversos grupos de trabajo. Las aplicaciones utilizadas en la elaboración de esta investigación son versiones estables de libre distribución que actualmente sirven

como pilar de las arquitecturas del software basado en la Web semántica. Entre estas herramientas están Jena, Sesame, Protégé y razonadores OWL. A continuación se detallan las características individuales de cada una de ellas.

JENA: Marco de trabajo en Java para crear aplicaciones en la Web semántica

Jena es un producto de los laboratorios de investigación de HP para el tratamiento de la Web semántica. Proporciona una API para RDF y OWL, un lenguaje de consulta (SPARQL) y un mecanismo de persistencia sobre bases de datos relacionales.

Para Jena un modelo ontológico es una extensión de un modelo RDF que proporciona capacidades adicionales para el manejo de las ontologías. En la figura 4.9 se muestra la estructura utilizada por Jena para tratar una ontología.

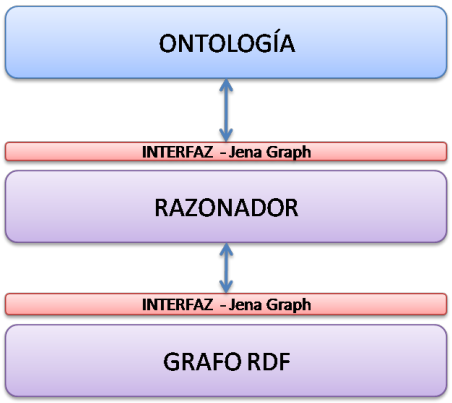

Figura 4.9: Jena, modelo de tratamiento de una ontología

El "*razonador*" o motor de inferencia es un componente débilmente acoplado de Jena que opera con un "*grafo RDF*" sobre el cual pueden realizarse determinadas operaciones de inferencia. El componente expone la interfaz "*Jena Graph*" que permite obtener las deducciones y nuevas aserciones realizadas sobre un modelo. Según el diseño de Jena es posible trabajar con otro motor de inferencia distinto, siempre y

cuando ese motor posea un adaptador a la interfaz de Jena.

Jena proporciona las siguientes operaciones de inferencia:

- Validación: para poder realizar una inferencia sobre un modelo es necesario verificar que este modelo esté correctamente escrito, comprobando todos los rangos definidos en RDF, ver si existen las clases que se utilizan y ver que las propiedades que utilizan tipos de datos (XSD) tienen valores consistentes. Es decir, que todo esté correctamente escrito para poder proceder a realizar inferencias sobre el modelo.

- Inferencia sobre las relaciones: con las relaciones entre clases o propiedades insertadas en el modelo, es posible realizar inferencias sobre las relaciones directas e indirectas.

- Derivaciones: nos permite ver las derivaciones que realiza el motor de inferencia sobre un modelo. Se pueden obtener todos los pasos que realiza el motor de inferencia para obtener el resultado de una cierta consulta.

- Acceso al flujo de datos y a las deducciones: para poder extraer las deducciones realizadas sobre el modelo, el motor de inferencia proporciona un mecanismo de obtención de los resultados. Es un simple método que retorna la colección de objetos deducidos.

- Control de proceso: el motor nos permite controlar el mecanismo de inferencia sobre las reglas individualizadas. El algoritmo RETE [FS87] utilizado puede trabajar de forma incremental, reduciendo así los costes asociados a la inserción de nuevos objetos dentro de la memoria de trabajo, ya que está desencadenaría una nueva inferencia sobre el conjunto conflicto construido.

- Trazas: es posible generar trazas sobre las operaciones realizadas por el motor de inferencia. Muy útil para utilizar depuración.

Jena proporciona un motor de inferencia capaz de procesar una base de reglas expresada en un formato propio, que utiliza el conocimiento expresado en la ontología. El motor de inferencia se configura con el algoritmo a utilizar, se cargan los hechos y reglas iniciales y se empieza a realizar la inferencia basándose en una entrada inicial, que indica consulta o inicio del motor de inferencia.
Existen dos tipos de reglas en Jena:

- Fordward rules (encadenamiento hacia delante – algoritmo RETE [FS87]):
  (?p rdfs:subPropertyOf ?q), notEqual(?p,?q) ← [(?a ?q ?b) → (?a ?p ?b)]

- Backward rules (encadenamiento hacia atrás - resolución SLG [Swi]):
  (s, p, o), (s1, p1, o1) ... ← (sb1, pb1, ob1)

Estos son los primeros pasos de la formalización del conocimiento en la nueva Web. En [BKPP07] se presenta un estudio notable sobre los distintos trabajos existentes en materia de lenguajes de reglas de la Web. El organismo W3C ha creado el grupo de trabajo RIF[29] con el objetivo de proponer recomendaciones para el intercambio de reglas en la Web semántica. Actualmente, de entre los proyectos más notables cabe destacar "Semantic Web Rule Language (SWRL)" [W3C04a] que utiliza la combinación de las ideas aportadas por los lenguajes OWL y RULE-ML[30].

Otros proyectos de apoyo:

- Mandarax [31]: libreria "Open Source" utilizada para la deducción de reglas. Es orientado a objetos, utiliza encadenamiento hacia atrás, con fácil integración con cualquier tipo de datos. Las reglas son almacenadas en RULE-ML.

[29] http://www.w3.org/2005/rules/wiki/RIF_Working_Group
[30] http://www.ruleml.org/
[31] Mandarax - http://mandarax.sourceforge.net/

- j-Drew [32]: es un potente motor de reglas escrito en Java, que utiliza lógica de primer orden. Actualmente proporciona interpretación de reglas escritas en Prolog y RULE-ML.

Respecto al lenguaje de consulta SPARQL, el grupo de trabajo de Jena proporciona un motor de proceso de consultas denominado ARQ y una aplicación servidor llamada JOSEKI[33], con la capacidad de recibir este tipo de consultas a través de peticiones GET o POST por el protocolo HTTP.

Actualmente, Jena es la herramienta más utilizada en los desarrollos basados en tecnología semántica. El soporte dado por el laboratorio de investigación en Web semántica de HP y la capacidad de manipular e inferir sobre un modelo ontológico, hacen de Jena uno de los marcos de trabajo que dará soporte al desarrollo del prototipo de esta investigación. En el apéndice D se detallan las operaciones básicas a realizar para habilitar la base de soporte de la gestión del modelo semántico del repositorio.

SESAME

SESAME es un conjunto de componentes software que proporcionan un marco de trabajo útil para crear aplicaciones basadas en tecnologías de la Web semántica. La figura 4.10 muestra una vista descriptiva de todos los componentes:

- Storage and inference layer (SAIL API): proporciona un conjunto de funcionalidades respecto al modo de almacenar un modelo RDF. Existen varias implementaciones como *MemoryStore* para el almacenamiento en memoria o *NativeStore* para el almacenamiento en estructuras de disco.

[32] j-Drew - http://www.jdrew.org
[33] Joseki - http://www.joseki.org/

- RDF Input / Output (RIO): son un conjunto de funciones de análisis y creación de modelos RDF para los flujos de entrada o salida de datos.
- Repository API: es una librería de alto nivel que proporciona un conjunto de funciones generales de manipulación de modelos RDF.
- HTTP Server: conjunto de Servlets creados con la finalidad de permitir el acceso a un repositorio SESAME a través del protocolo HTTP. El componente HTTP-Client proporciona el conjunto de funciones que permiten conectar múltiples repositorios SESAME.

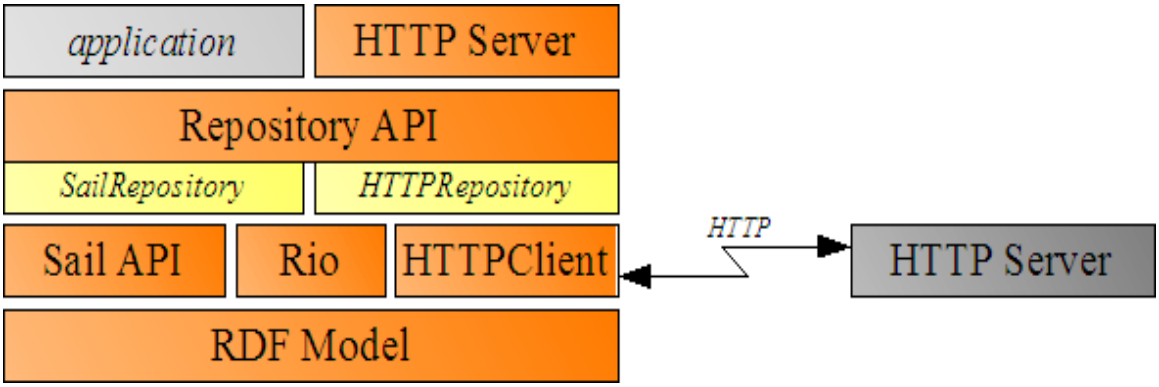

Figura 4.10: componentes de SESAME

SESAME incluye la implementación de un lenguaje de consultas sobre modelos RDF denominado Re-SQL [SeR04]. Este lenguaje (con un objetivo similar al perseguido por SPARQL) permite obtener conjuntos de elementos de los modelos RDF gestionados por SESAME.

Respecto a la inferencia, existen implementaciones como la de la universidad *Freie* de Berlin [34] que permiten la ejecución de reglas SWRL sobre un modelo RDF almacenado en un repositorio SESAME.

[34]Sesame - http://www.inf.fu-berlin.de/inst/ag-nbi/research/swrlengine/

PROTÉGÉ: Editor de Ontologías y marco de trabajo para la adquisición de conocimiento

Protégé[35] es una herramienta para el desarrollo de ontologías y sistemas basados en el conocimiento creada en la Universidad de Stanford. Protégé está desarrollado en Java, luego puede ejecutarse en cualquier plataforma que soporte la máquina virtual. Las aplicaciones desarrolladas con Protégé son empleadas en resolución de problemas y toma de decisiones en dominios particulares.

Además de proporcionar una herramienta visual para el desarrollo de ontologías, Protégé incluye una herramienta de desarrollo (PDK - Protege Development Kit[36]) para la gestión de sus repositorios, que proporciona, desde el código, la capacidad de gestión de múltiples características tales como la persistencia del modelo, la ejecución de consultas o la modificación del propio esquema del modelo. En el caso particular de modelos OWL, Protégé ofrece la biblioteca Protégé-OWL[37] para la gestión de modelos OWL. Esta biblioteca utiliza funciones proporcionadas por Jena. En el apéndice E se muestran las clases e interfaces utilizadas para la gestión de modelos de ontologías.

Razonadores OWL

Nos permite realizar operaciones básicas de inferencia sobre una ontología escrita en OWL. Las operaciones básicas de un razonador basado en lógica de descripciones son:

- Verificar consistencia: detección de inconsistencias dentro del modelo. Asegura que todos los elementos queden bien definidos según las aserciones lógicas incluidas en una ontología: Clases, propiedades e instancias bien definidas. Ejemplo:

[35]Protégé - http://protege.stanford.edu/
[36]PDK - http://protege.stanford.edu/doc/dev.html)
[37]Librería Protege OWL - http://protege.stanford.edu/plugins/owl/api/guide.html

si una persona es vegetariana no puede ser carnívora.

- Realizar una clasificación taxonómica del modelo: obtiene la jerarquía completa del modelo, basada en deducciones sobre la lógica descriptiva.

- Satisfacibilidad de conceptos: determina para una clase si es posible que tenga alguna instancia. Si una clase es insatisfactible y tiene definida una instancia entonces existe una inconsistencia en la ontología.

- Realización: encuentra las clases más específicas a las que pertenece un individual (los tipos directos). Este proceso se puede ejecutar después de realizar la clasificación taxonómica del modelo. Por ejemplo, si hemos definido en nuestro sistema que el concepto "*pizza pobre*" es aquella que tiene menos de tres ingredientes y definimos un nuevo individual como una "*pizza vegetariana*" con dos ingredientes, el proceso de realización determinará que además de ser el individual una "*pizza vegetariana*" es una "*pizza pobre*".

Los razonadores suelen ser componentes independientes que puede incluirse como elementos adicionales a cualquier librería que gestione modelos OWL. Tanto en el caso de las liberías de Jena como Protégé se permite el uso de razonadores externos a los componentes aportados. La conexión se realiza a través de una interfaz denominada DIG[38].

De entre los razonadores OWL-DL cabe destacar los proyectos Racer, Pellet y Facet.

- Racer Pro[39]: razonador OWL-DL comercial creado por la empresa Racer Systems GmbH & Co.KG. Proporciona un marco de trabajo para realizar las operaciones básicas anteriormente enunciadas, además de un amplio conjunto posible

[38] http://dl.kr.org/dig/
[39] http://www.racer-systems.com/

de inferencias y un lenguaje de consulta específico para OWL "*nRQL*".

- Pellet[40]: se distribuye bajo los términos de la licencia MIT. Empezó como un proyecto de los laboratorios Mindswap y ahora ofrecen soporte y personalización del producto a través de la empresa Clark & Parsia. Incluye un motor de inferencia que interpreta reglas SWRL sobre un modelo OWL.

- FaCT++[41]: distribuido bajo licencia pública GNU, escrito en C++. Razonador OWL muy eficiente, posee librerías de acceso que permiten la comunicación desde Java. Incluye como el resto de los razonadores OWL-DL mencionados la funcionalidad general descrita al principio de la sección, además incluye la interpretación e inferencia sobre modelos descritos en SROIQ [HKS06], una extensión de la lógica de descripciones.

[40]http://pellet.owldl.com/
[41]http://owl.man.ac.uk/factplusplus/

4.2. Descripción inicial de la solución

Para solucionar los problemas planteados en el capítulo 3, en esta investigación se ha diseñado un prototipo de un repositorio semántico de objetos de aprendizaje que cubre los siguientes objetivos:

- Flexibilidad: capacidad de albergar cualquier tipo de conceptualización de un "objeto de aprendizaje".
- Independencia: un "objeto de aprendizaje" puede ser descrito de diferentes formas según las distintas conceptualizaciones del mismo.
- Interoperabilidad: fomento de la interoperabilidad entre los distintos sistemas de e-learning.
- Semántica: posibilidad de enlazar información existente en los registros de metadatos con conceptos de otras ontologías.
- Razonamiento: capacidad de realizar inferencias sobre los registros almacenados.
- Comunicación: proceso de la información de forma autónoma por parte de cualquier tipo de agente software.

La figura 4.11 representa los diferentes actores que interactúan con el repositorio de objetos de aprendizaje SLOR.

- LO-CREATOR: es el usuario que construye los objetos de aprendizaje. Generalmente utiliza herramientas de autor especiales para crear el contenido del objeto de aprendizaje. El contenido puede tener asociados determinados registros de metadatos respecto al diseño de la actividad de aprendizaje (ej. IMS-LD).

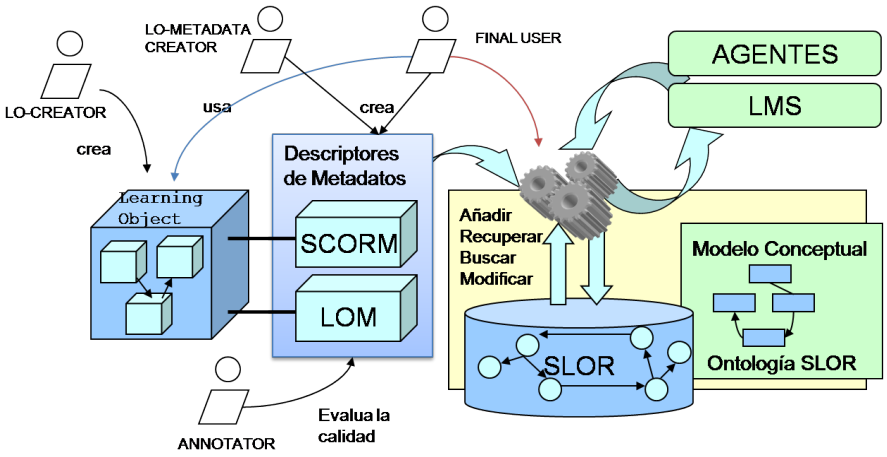

Figura 4.11: SLOR - Semantic Learning Objects Repository

- LO-METADATACREATOR: usuario encargado de describir y catalogar los objetos de aprendizaje puestos a disposición de publicación en el repositorio por parte de los autores. Normalmente en instituciones educativas donde existe un gran volumen de información digital no estructurada, se contratan a personas para que ejerzan la labor de catalogar la información insertando registros de metadatos definidos en algún estándar válido en un repositorio central.

- FINAL USER: es el usuario que realiza las búsquedas sobre el repositorio para localizar los objetos de aprendizaje que le interesen. También puede introducir registros de metadatos respecto a nuevos objetos de aprendizaje ubicados y accesibles por el repositorio en Internet o en algún otro tipo de red.

- ANNOTATOR: una vez publicado un objeto de aprendizaje los usuarios con conocimientos y criterio aportan valoraciones sobre el contenido de un objeto de aprendizaje y la calidad de los registros de metadatos albergados en el repositorio. Este tipo de valoraciones también se publican para enriquecer la información existente en el repositorio.

- LMS: en el esquema representado en la figura 4.11 son los sistemas LMS externos

que acceden a contenidos del repositorio mediante una interfaz especifica de acceso.

- AGENTS: representa a cualquier otra aplicación software que desee utilizar algunos de los servicios ofertados por el repositorio.

La secuencia del ciclo de vida de un objeto de aprendizaje gestionado por SLOR sigue los siguientes pasos:

1. El punto de comienzo del ciclo de vida de un objeto de aprendizaje se inicia por la creación del objeto de aprendizaje realizada por un autor ("*LO-CREATOR*", que puede ser una persona, grupo de personas o empresa).

2. Una vez finalizado el proceso de creación del objeto de aprendizaje se elige por parte del creador de registros de metadatos (que puede ser un usuario final del repositorio"*FINAL USER*" o un usuario específico encargado de catalogar los objetos de aprendizaje"*LO-METADATACREATOR*") el conjunto de registros de metadatos que ha de llevar asociados en función de su uso final, si va a ser interpretado por uno o varios LMS utilizara un conjunto de registros de metadatos diferentes a otros objetos como los libros o apuntes que simplemente requieren de un único registro para ser catalogados.

3. Finalmente un usuario general del repositorio "*FINAL USER*" puede localizar el objeto de aprendizaje insertado por el creador gracias a los servicios de búsqueda proporcionados por el repositorio.

4. Existe otro tipo de usuario representado en la figura 4.11 como "*ANNOTATOR*", encargado de realizar las revisiones del contenido del repositorio, así como las anotaciones sobre la calidad del objeto de aprendizaje.

Un objeto de aprendizaje en SLOR puede ser descrito con múltiples registros de metadatos. Por ejemplo, en SCORM indicando la organización de contenidos, el tipo de recursos utilizados, la secuenciación del mismo, etc... o en IEEE LOM. Agentes externos pueden buscar y operar con esos objetos gracias al modelo formal propuesto en esta investigación.

Para poder dar soporte a las distintas conceptualizaciones de un objeto de aprendizaje, en este capítulo se describe la representación semántica de metadatos de los objetos de aprendizaje, la caracterización ontológica necesaria, así como el soporte formal utilizado para la descripción de recursos. Finalmente se expone un modelo de la arquitectura que da soporte al modelo formal, además de la descripción de la implementación del prototipo utilizado para evaluar la investigación.

4.2.1. Representación semántica de metadatos de objetos de aprendizaje

Como ya se ha comentado en la sección 4.1.2, las ontologías son un instrumento utilizado para el modelado conceptual. En particular, la existencia de esquemas ontológicos es necesaria cuando necesitamos algún grado de automatización, proporcionando un escenario para la delegación de tareas a agentes software con capacidades autónomas de razonamiento. Asumiendo este hecho, si una ontología de la Web semántica es modelada con un propósito específico de una investigación, las definiciones de todos los elementos (clases, objetos, acciones y propiedades) respecto al concepto de objeto de aprendizaje y las acciones asociadas que lo rodean respecto al desarrollo y despliegue, serán parte de la ontología utilizada en está investigación.

Los repositorios de objetos de aprendizaje juegan un importante papel en las áreas

de reutilización y diseño de contenidos, siendo los proveedores de los artefactos orientados al aprendizaje. Sin embargo, las actuales prácticas de creación de metadatos, generan colecciones de artefactos muy difíciles de interpretar por un agente. Este hecho obstaculiza las oportunidades de reutilización [SAS05]. Las ontologías de la Web semántica pueden ser utilizadas para mejorar la calidad de los registros de metadatos de los objetos de aprendizaje, aunque no es bastante con esto. Para responder de forma adecuada a las peticiones realizadas por agentes externos, los repositorios han de conocer la cantidad, el tipo y la calidad de los registros de metadatos que almacenan. Las técnicas de la Web semántica aplicadas a los metadatos de los objetos de aprendizaje sirven como:

- Mecanismo para la integración de los distintos tipos de objetos de aprendizaje, esencialmente para el desarrollo de los sistemas que permiten elegir y distribuir objetos de aprendizaje.

- Modo de establecer una taxonomía común unificada para todos los sistemas de gestión de contenidos de aprendizaje.

- Modo de proporcionar conocimiento definido en los estándares de objetos de aprendizaje (como en LOM), dirigida a enriquecer comportamientos autónomos por parte de agentes software externos.

- Medio para proporcionar capacidades de razonamiento a los LMS, por el uso de la lógica de descripciones en las definiciones de las conceptualizaciones y sus relaciones [BCM$^+$02].

Los objetos de aprendizaje son artefactos capaces de ser utilizados en distintos escenarios, principalmente en los procesos de aprendizaje, pero también en los procesos entre sistemas, como por ejemplo, la adquisición previo pago de objetos de aprendizaje, la selección de los mismos, la composición automática o el intercambio. Cada

escenario requiere un conjunto específico de metadatos. Por lo tanto, proporcionar más y mejor los metadatos a los objetos de aprendizaje amplia la colección de escenarios en los que puedan ser utilizados, generando como efecto secundario un alto nivel en la reutilización. Si el objetivo es la construcción de un software que se aproveche de la información existente en los metadatos, es necesario un marco de representación del conocimiento que vaya más allá que los simples registros de metadatos. La utilización de las ontologías en el contexto de la tecnología de la Web semántica puede abrir nuevos escenarios donde puede alcanzarse un alto nivel de automatización procesando el significado existente en el contenido de los registros de los metadatos. En la siguiente sección, se analiza en detalle el papel que juegan las ontologías de la Web semántica en la creación de repositorios de objetos de aprendizaje más flexibles, el tipo de representaciones del conocimiento que pueden ser utilizadas, así como la base para extender las especificaciones de metadatos de los objetos de aprendizaje.

Semántica de las definiciones

Muchas veces es desconocido el hecho de que los metadatos sean creados para soportar funciones específicas asociadas al objeto descrito. Los creadores no ofrecen importancia a detalles concretos de los requisitos de las funciones que eventualmente hacen uso de los registros generados. Cada una de las caracterizaciones de los objetos de aprendizaje [McG04], descritas en la sección 3.1 del capítulo anterior, cubre diferentes tipos de requisitos tanto por la forma de los metadatos utilizados para describir el objeto como por el tipo de funciones que pueden ser realizadas con ellos. La tabla 4.2 resume las principales consideraciones a tener en cuenta.

En la primera definición no es necesario definir metadatos para describir un objeto de aprendizaje, consecuentemente el tipo de funciones permitidas están orientadas al consumo humano, no al procesamiento automático de los mismos. El sentido de esta

Conceptualizaciones	Metadatos Nec.	Funciones
`LearningObject- AsAnything`	Ninguno	Consumo humano
`LearningObject- AsAnythingDigital`	Ninguno	Consumo humano, semántica tácita
`LearningObject- AsAnythingWithEducational Purpose`	Algunos	Consumo humano, semántica tácita en campos de datos concretos
`LearningObject- OtherSpecificAccounts`	Basados en una especificación	Consumo humano, semántica tácita con datos concretos y acordados basados en una especificación
Descripciones conformes con las especificaciones basadas en la ontología del modelo	Basados en una ontología que representa una especificación	Todas las funciones posibles a realizar con técnicas de inferencia sobre modelos semánticos formales

Tabla 4.2: Tipos de funciones permitidas en las diferentes caracterizaciones del término "objeto de aprendizaje"

definición es el de poseer la conceptualización de un "objeto de aprendizaje".

La segunda definición introduce el concepto "digital", en el que existe la posibilidad de extraer una semántica tácita [SRT05]. Por ejemplo, puede extraerse con técnicas de minería de datos el contenido de los objetos, incluido el resumen de textos existentes, extracción de palabras claves e índices utilizados en la recuperación de información. Desafortunadamente, para conseguir este tipo de semántica no existen herramientas relevantes, siendo a día de hoy lo más similar los motores de búsqueda.

La tercera definición introduce la necesidad de algún tipo de descripción. Esta puede ser una simple anotación escrita sin ningún formalismo o complejos registros de metadatos. Al no existir restricción en la forma de la descripción, en sentido general, las capacidad de procesamiento automático son limitadas.

La cuarta definición progresa en la formalización de los metadatos por considerar una conformidad con alguna especificación existente. Por lo tanto engloba muchas de las definiciones aportadas por las especificaciones de la tecnología de e-learning como LOM y SCORM. En este marco, los metadatos poseen complejos esquemas que permiten la explotación de la reutilización de los objetos definidos, además del intercambio y comercialización de los mismos.

La última fila de la tabla representa el máximo grado de especificación formal, proporcionado por el uso de registros de metadatos que usan descripciones y relaciones con términos de una ontología. Las descripciones pueden estar conectadas a esquemas complejos de relaciones descritas en una ontología, proporcionando así un alto nivel de expresividad formal capaz de ser procesado por un motor de inferencia en un agente externo. Por ejemplo, el campo "*coverage*" de LOM puede utilizar términos de la ontología TGN[42], que proporciona una representación de las entidades geográficas. Búsquedas sencillas, como recuperar los objetos de países que tengan una monarquía, pueden ser realizados por medio de una simple inferencia sobre los conceptos referenciados por los metadatos.

Descripción semántica de los metadatos en el estándar IEEE LOM

Dentro de la conceptualización "*LearningObject-OtherSpecificAccounts*" definida en la sección 3.1, podemos especificar una relación entre los metadatos existentes descritos por la especificación formal de LOM creando así la conceptualización "*LOM-LearningObject*" derivada de este término. Esta conceptualización necesita al menos la descripción de algún registro de metadatos de LOM ($\exists LOM\_Record$). En

[42]Getty Thesaurus of Geographic Names - http://www.getty.edu/research/tools/vocabulary/tgn/

Propiedad LOM-LearningObject	Rango
Identifier	single String
Title	single LangString
Language	multiple object oc_HumanLanguage
Description	multiple object DescriptiveProperty
Keywords	multiple DescriptiveProperty
Coverage	multiple object DescriptiveProperty

Tabla 4.3: Esquema básico de LOM utilizado en la conceptualización *LOM-LearningObject*

el desarrollo de esta etapa de investigación, usaremos una caracterización formal simplificada del estándar LOM, dado que el objetivo principal se focaliza en la inserción flexible de diferentes conceptualizaciones de un objeto de aprendizaje.

Para utilizar la descripción de recursos semántica (explicada en este capítulo), es necesario utilizar un concepto abstracto que agrupe sentencias escritas en lenguaje natural en cualquier idioma (representadas por la clase "*LangString*") y sentencias que expresen relaciones o descripciones en un modelo formal ("*Formal_Statement*"). La clase "*DescriptiveProperty*" que deriva de la clase "*oc_Statement*" de OpenCyc, agrupa estos dos tipos de expresión. Este esquema de clases representa la posibilidad de almacenar la información de un campo del registro de metadatos, tanto en lenguaje natural como en un lenguaje formal capaz de relacionar o describir un objeto con varios conceptos de otras ontologías externas del modelo base subyacente. El resto de propiedades poseen definiciones simples de tipos de datos normalizados en la especificación de XSD.

4.2.2. Caracterización ontológica necesaria para el diseño flexible de repositorios

Los repositorios almacenan datos bajo un esquema poco flexible que recoge elementos fundamentales para su localización, búsqueda y organización. La descripción

del contenido del objeto es muy variable, ya que la propia naturaleza del lenguaje humano ofrece múltiples formas de expresión para un mismo significado. En el análisis de repositorios presentado anteriormente se han expuesto las carencias existentes en la definición de los metadatos: los campos se expresan en lenguaje natural, hecho que restringe la potencia de búsqueda y clasificación de objetos. Para entenderlo mejor se expone el siguiente caso de ejemplo: en cualquier repositorio de los analizados podemos almacenar un LO cuya finalidad didáctica es la de explicar la evolución histórica de la catedral de La Almudena. Dentro del sistema podemos incluirlo con esta misma descripción y catalogarlo en alguna de las categorías predefinidas, es más incluso en los repositorios más modernos podemos enlazarlo con alguna etiqueta ("tag"). Si quisiéramos recuperar los cursos sobre los edificios religiosos de la Comunidad de Madrid, no sería posible. Los repositorios actuales carecen de los mecanismos necesarios para recuperar los objetos que cumplen con el sentido común de la consulta anterior. ¿Cuál es el problema? Es sencilla la respuesta, los repositorios no poseen estructuras de almacenamiento de conocimiento, ni ontologías que permitan estructurar determinadas expresiones de conocimiento. La búsqueda de la solución a este problema ha sido fruto del estudio realizado sobre los distintos modelos de representación del conocimiento.

Con las distintas definiciones de objeto de aprendizaje proporcionadas por el análisis de McGreal, ha sido creada una ontología. La tabla 4.4 muestra la correlación existente entre las definiciones de McGreal y las clases de la ontología.

Cualquier concepto enlazado a la representación de actividad de aprendizaje puede ser considerado un objeto de aprendizaje. En sentido general, el aprendizaje se considera un evento dentro del modelo. En la figura 4.12, todas las clases prefijadas con "*oc*" representan una clase de la base de conocimiento OpenCyc. Por lo tanto, el término "aprendizaje" es representado por la clase "*oc_Learning*", que

Definiciones del análisis de McGreal	**Clases de la ontología**
Cualquier cosa	`LearningObject-AsAnything`
Cualquier cosa digital, con o sin propósito educativo	`LearningObject-AsAnythingDigital`
Cualquier cosa con propósito educativo	`LearningObject-AsAnythingWithEducationalPurpose`
Objetos digitales que tengan un propósito educativo Objetos digitales que tengan definidos de forma especial su propósito educativo.	`LearningObject-OtherSpecificAccounts`

Tabla 4.4: Correlación entre las definiciones de McGreal y las clases de la ontología

simboliza la definición de aprendizaje en OpenCyc. La representación abstracta en la cual derivan todos los términos analizados por McGreal es definida en la clase "*LearningObject-Generic*". Una instancia de esta clase será "algo" utilizado en el aprendizaje. La relación "utilizado" entre los dos términos se describe con la propiedad "*used-in*". La clase "*LearningObject-AsAnything*" engloba todos los posibles significados de objeto de aprendizaje. La clase "*LearningObject-AsAnythingDigital*" representa los objetos digitales utilizados en el aprendizaje. El término OpenCyc "*ComputerFileCopy*" puede ser utilizado como una caracterización de cualquier objeto digital, por lo tanto, sirve como término base de la definición de objeto de aprendizaje digital. La clase "*LearningObject-AsAnythingWithEducationalPurpose*" ha sido definida para representar los objetos que tienen descrito su propósito educativo. Estas dos últimas caracterizaciones están combinadas en las especificaciones actuales de e-learning. Cualquier individual de la clase "*LearningObject-AsAnythingDigital*" puede tener asociado un registro de metadatos según alguna especificación formal existente, como LOM, ISBD o IMS-LD. Por ello, se ha creado en el modelo formal la propiedad "*hasAssociatedMetadataRecord*" cuyo rango es una clase que representa

un registro abstracto de metadatos. Para dar soporte a las definiciones cuarta y quinta definición de McGreal, es necesario introducir dentro del modelo formal los esquemas de metadatos de las especificaciones estándar, tales como LOM ("*LOM_Record*") o SCORM ("*SCORM_SCO_Manifest*"). La figura 4.12 muestra las relaciones entre todos los términos descritos. En esta figura, las clases de la ontología son mostradas en rectángulos con un borde sólido negro, con varios compartimientos separados por líneas horizontales. El nombre de la clase está en el compartimiento superior, mientras el resto almacena las propiedades (aparecen precedidas por un círculo "o"). En cuanto a las restricciones aparecen precedidas por un círculo "R". Por ejemplo, la clase "*LearningObject-Generic*" aparece enlazada a la clase "*oc_Learning*" por una flecha que representa la propiedad objeto "*used-in*" en el correspondiente compartimiento de la clase. Esta figura también muestra la jerarquía de relaciones entre las clases de la ontología. Las clases que representan conceptos genéricos aparecen en la parte superior de la figura, en cambio las clases especificas están en la parte inferior. La relación de jerarquía es representada con una flecha acabada en un triángulo hueco. Por ejemplo, la clase "*LearningObject-LOM*" está enlazada a la clase "*LearningObject-AsAnythingDigital*" indicando la relación de jerarquía. Para un análisis más profundo de este modelo formal, revisar el apéndice A donde se lista al completo el código OWL de la ontología.

4.2.3. Descripción de recursos

La descripción de recursos tradicional, se realiza en los repositorios de objetos de aprendizaje cumplimentando los campos de metadatos en lenguaje natural o con conceptos sujetos a fuertes restricciones, pero sin utilizar un lenguaje formal para los mismos. Este hecho, ya discutido en esta investigación, determina la carencia de realizar tratamientos complejos con la metainformación almacenada. Para solucionar este problema, la información escrita en lenguaje natural puede ir complementada

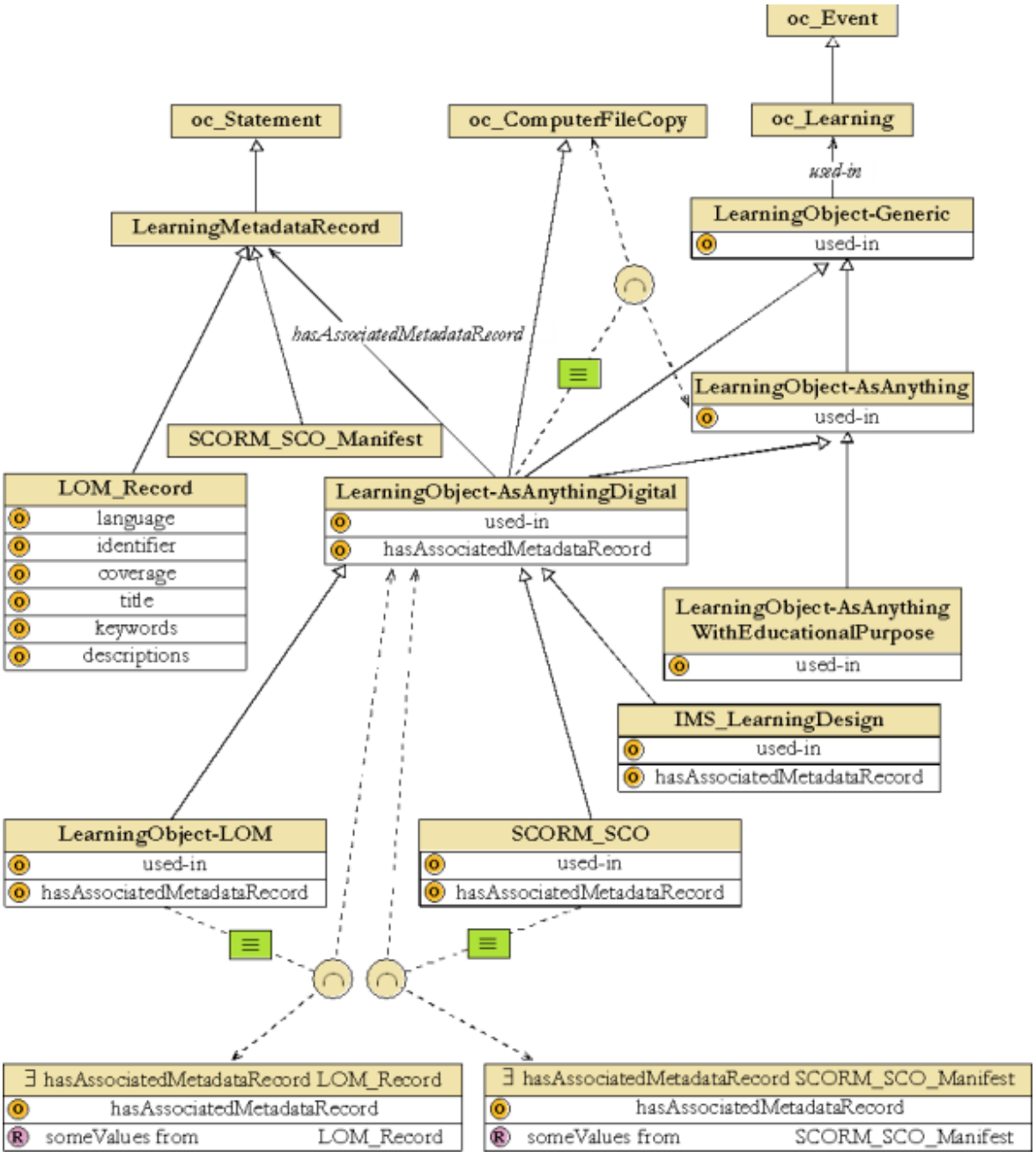

Figura 4.12: Modelo flexible del repositorio - representación gráfica de la ontología

con referencias a conceptos o predicados formales lógicos que inserten algún aspecto descriptivo formal.

Los conceptos pueden representarse de la siguiente forma:

(ontología) Término[43]

Los predicados pueden estar definidos en la siguiente forma:

(ontología) predicado [Clase]: instancia

Figura 4.13: Recurso imagen - Cuadro la meninas de Diego Velázquez

Gracias a este modelo de expresión formal, los registros de metadatos pueden enlazar información con elementos de una ontología. En el campo "*1.4 Description*"

[43]Clase o instancia definida en la ontología

de un registro de metadatos IEEE LOM[44] asociado a una imagen del cuadro de las "*Meninas*" existente en Internet y utilizada como recurso de aprendizaje (ver figura 4.13), podemos encontrarnos los siguientes predicados:

```
(AAT)reproduces[Oil-Painting]:Las_Meninas
(OpenCyc)painter[Artist]:Diego_Velazquez
(OpenCyc)has-painted[Picture]:Infant_Margarita
(OpenCyc)has-painted[Picture]:Menina_Isabel_Velasco
(OpenCyc)has-painted[Picture]:Menina_Agustina_Sarmiento
(OpenCyc)has-painted[Picture]:María_Bárbola
(OpenCyc)has-painted[Picture]:Nicolás_Pertusato
(OpenCyc)has-painted[Picture]:Diego_Velazquez
(OpenCyc)has-painted[Mirror-Figure]:King_FelipeIV
(OpenCyc)has-painted[Mirror-Figure]:Queen_Mariana_de_Austria
```

De la misma forma, en el campo "1.6 Coverage" podemos insertar conocimiento sobre la época y el lugar dónde han sido pintados, haciendo uso de la ontología propuesta por el TGN (Thesaurus of Geographic Names) y por conceptos de sentido general aportados en OpenCyc.

```
(TGN) situatedIn[TGN-nation]:Spain
(OpenCyc)situatedIn[OpenCyc-CalendarCentury]:SeventeenthCenturyCE
```

En el campo "keywords" podemos enlazar con algún concepto de una ontología para facilitar la labor de indexado de las búsquedas.

[44]Utilizado para realizar la explicación, a modo práctico sobre el esquema del repositorio podría definirse una nueva clase "*XMP_Record*" que represente un registro de descripción específico para una imagen. De esta forma, objetos de tipo "*LearningObject-AsAnyThingDigital*" como imágenes digitales utilizadas para alguna actividad de aprendizaje pueden ser descritas con registros "*XMP_Record*" incluyendo las expresiones semánticas descritas.

```
(AAT)Oil-Painting
(AAT)Diego_Velazquez
```

Incluso también puede establecerse de forma unificada un esquema de clasificación que permita facilitar la labor de clasificación dentro de los repositorios. En el siguiente ejemplo se muestra la expresión semántica asignada en el campo *9.2 Taxon* de la categoría "9. Classification" de un registro de metadatos IEEE LOM:

```
(OpenCyc)History \\ Art \\ Pictures \\ SeventeenthCenturyCE
```

Aún así, es necesario ampliar este modelo para expresar las relaciones del concepto "objeto de aprendizaje" con los otros conceptos evocados por las ontologías gestionadas por el repositorio. Por ello es necesario el uso de predicados relacionales con la siguiente forma:

(Ontología) predicadoRelacional(arg1 arg2 .. argn)

Ejemplo: en el cuadro, la infanta Doña Margarita está cerca de las meninas.

```
(OpenCyc)near(Infant_Margarita Menina_Isabel_Velasco)
(OpenCyc)near(Infant_Margarita Menina_Agustina_Sarmiento)
```

Gracias a estas expresiones, agentes externos podrán realizar inferencias complejas sobre el repositorio. Por ejemplo extraer los identificadores de los registros de metadatos almacenados en el repositorio que describen los cuadros que tengan al menos dos meninas que estén cerca de una infanta. En las siguiente sección se describe la arquitectura de un modelo semántico que permite albergar cualquier tipo de conceptualización de objeto de aprendizaje, incluyendo el marco de descripción de recursos mencionado, consiguiendo así la flexibilidad deseada en nuestro repositorio.

4.3. Versión inicial de la arquitectura para el almacenamiento semántico de los objetos de aprendizaje

A lo largo de esta sección se presenta la primera versión del prototipo de arquitectura basada en los requisitos formulados en la sección 4.2 a partir de los objetivos descritos en el capítulo 3. Después se presentan los inconvenientes encontrados tras la implementación del primer prototipo, obligando a cambiar la orientación inicial para ser derivada en un nuevo modelo que será presentado en la última sección y descrito en el siguiente capítulo.

Los requisitos de SLOR determinan el modelo de diseño representado en la figura 4.14. Como puede apreciarse, la arquitectura está estructurada en tres capas, según el modelo-vista-controlador:

- La capa de interfaz es la encargada de interactuar bien con los usuarios finales del sistema o bien con los creadores de "objetos de aprendizaje". Las Interfaces Web proporcionan acceso al repositorio, por lo tanto, todo usuario registrado, desde cualquier lugar podrá acceder para consultar e introducir datos si lo desea (característica implementada en repositorios actuales, tal como MERLOT, NLN o CAREO). También los agentes software y los sistemas de gestión de contenidos didácticos pueden acceder a las funcionalidades ofrecidas por el repositorio.

- La capa de servicio agrupa toda la funcionalidad del repositorio estructurada en módulos y basada en contratos por interfaz. Posee 3 subcapas, la primera es la capa de interfaz del servicio, que incluye la implementación de los contratos de acceso de los agentes externos, seguida de la capa de módulos encargados de abstraer los bloques de funcionalidad ofertados por el repositorio y finalmente el núcleo, dónde se implementan las funcionalidades básicas de acceso al esquema

ontológico definido en OWL.

- Para manipular el modelo, es necesario utilizar la última capa de la arquitectura, que es la encargada de mantener la persistencia entre el modelo manipulado con clases y un sistema gestor de base de datos.

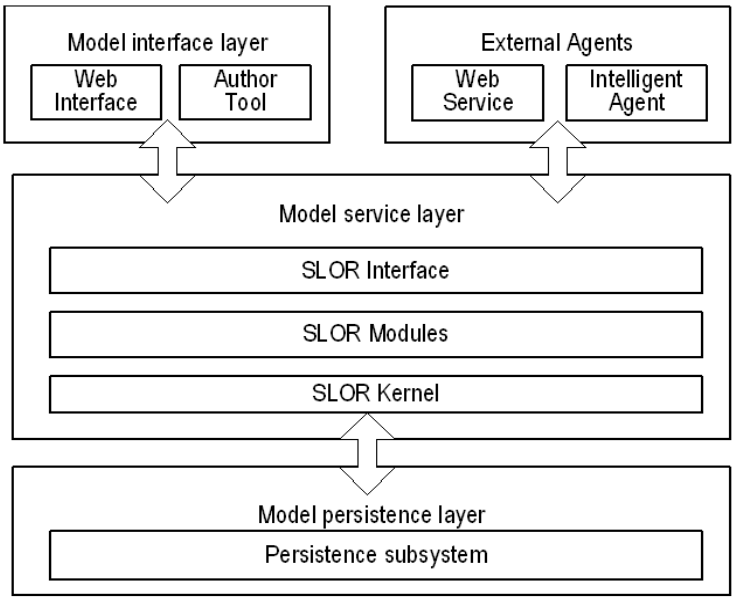

Figura 4.14: SLOR - Arquitectura

Esta arquitectura permite almacenar y gestionar objetos de aprendizaje con la representación semántica propuesta en las secciones anteriores. Sistemas multiagentes y Servicios Web externos podrían acceder a las funcionalidades del repositorio al compartir el modelo formal común de representación de las diferentes caracterizaciones de objeto de aprendizaje. De esta forma operaciones de búsqueda, composición automática de objetos de aprendizaje y demás funciones que requieran algún tipo de inferencia podrán ser implementadas.

De cara a la implementación de un prototipo según esta arquitectura, es necesario analizar las diferentes tecnologías. Para realizar un modelo sencillo con el único

propósito de servir de prueba inicial, implementaremos la capa de interfaz con tecnología web JSP, la capa de servicio con clases de control Struts y la capa de persistencia con el subsistema de persistencia de Jena. En la siguiente sección se detalla el modelo utilizado.

4.3.1. Diseño inicial del prototipo del repositorio semántico de objetos de aprendizaje

El prototipo de SLOR basado en la ontología descrita en las secciones anteriores, ha sido especificamente diseñado para la creación y administración de metadatos de objetos de aprendizaje con propósitos de integración e intercambio con otros sistemas. La figura 4.15 muestra las principales capas que definen el modelo de arquitectura propuesto en la sección anterior. Para mantener la consistencia entre las diferentes capas, el prototipo SLOR utiliza la ontología que describe el modelo formal utilizado. Las funcionalidades están agrupadas en módulos siguiendo el principio de escalabilidad.

Para separar la lógica web del resto de la aplicación, la capa interfaz utiliza tecnología *Apache Struts*[45]. Las acciones de Struts invocan los servicios de SLOR y recuperan los resultados, que serán adaptados al estado actual de la interfaz de usuario, porque diferentes conceptualizaciones de "objeto de aprendizaje" pueden ser utilizadas. Gracias a la ontología definida, el modelo conceptual del cliente no determina las funcionalidades ofrecidas por SLOR. Por ejemplo, un agente software que entienda que cualquier objeto digital utilizado en la enseñanza es un objeto de aprendizaje, utilizará la conceptualización de la ontología "*LearningObject-AsAnythingDigital*" y recuperará aquellos objetos del repositorio que sean instancias directas o indirectas de está clase. En cambio, si un agente software sólo puede procesar objetos con

[45]Jakarta Struts - http://www.sfu.ca/ wcooi/projects/470/struts.html

registros de metadatos de SCORM, necesitará utilizar una restricción más fuerte a la hora de procesar la información, luego el agente solicitará instancias de la clase "*SCORM_SCO_Manifest*". Es importante señalar que un objeto de aprendizaje puede estar descrito por varios registros de metadatos, luego diferentes aplicaciones clientes podrán usar el objeto de aprendizaje en distintos escenarios. Un ejemplo es la organización didáctica de un objeto sobre el manifiesto de un registro SCORM y el diseño de la actividad de aprendizaje en un registro IMS LD ("*IMS_LearningDesing*"). Este modelo es extensible y está preparado para soportar futuras conceptualizaciones que podrían incluirse en la ontología.

Para transmitir información entre la entrada de un formulario y una acción, el marco de trabajo de Struts define la clase "*BeanActionForm*". Cada "*BeanActionFrom*" manipula una clase en la ontología. Ej: "*LearningObject_LOMActionForm*" corresponde a la clase "*LearningObject-LOM*".

La capa de servicio proporciona un acceso transparente a las diferentes funcionalidades del repositorio semántico de objetos de aprendizaje:

- La interfaz de SLOR define el protocolo de comportamiento entre el servicio ofrecido y las acciones de la interfaz de usuario. El principio de diseño está basado en las acciones de los metadatos, implementadas en módulos que permiten la accesibilidad, interoperabilidad, durabilidad y reutilización. Esta interfaz recibe las peticiones de una interfaz web o de otros agentes, y las redirige al módulo encargado de procesarlas en la siguiente capa.

- Los módulos de SLOR proporcionan una arquitectura escalable que permite añadir nuevas funcionalidades fácilmente. Para incluir las funcionalidades relacionadas con la creación, borrado y actualización de los objetos de aprendizaje,

ha sido implementado un módulo de administración.

- El núcleo de SLOR proporciona una capa intermedia con funcionalidades básicas para manipular el modelo. Ejemplos de funciones son: "*getIndividuals*" (que recupera todos los individuales de una clase dada) o "*setMultipleProperty*" (que inserta todos los objetos de una lista como valores múltiples de una propiedad). Para manipular a bajo nivel el modelo de la ontología OWL, el núcleo de SLOR utiliza Jena (descrito en la sección 4.1.5).

La persistencia es la capacidad de almacenar estructuras de datos en ficheros o bases de datos relacionales. Jena proporciona persistencia transparente de los modelos conceptuales basados en una ontología por el uso de un motor de base de datos. Esta característica permite una fácil administración del contenido de grandes de modelos, en lugar de utilizar un esquema de almacenamiento complicado e ineficiente basado ficheros XML. En SLOR el modelo persistente subyacente se almacena en una base de datos MySQL; para el programador esta característica es transparente, permite que se manipulen miles de registros de distinto esquema sobre una misma tabla. En el apéndice D se describe el modo de crear un modelo persistente en Jena.

Inserción de la semántica de la información en los metadatos

El creador de metadatos "*LO-METADATACREATOR*" es el usuario que inicia la acción de creación de un registro de metadatos, invocando desde un explorador web la interfaz de creación de metadatos del tipo correspondiente que desee (LOM, ISBD, SCORM, etcétera). A lo largo de este apartado se detalla el conjunto de acciones que, a modo general, se desencadenan internamente en el repositorio para la inserción de un nuevo registro de metadatos LOM a partir de la interfaz representada en la figura 4.16.

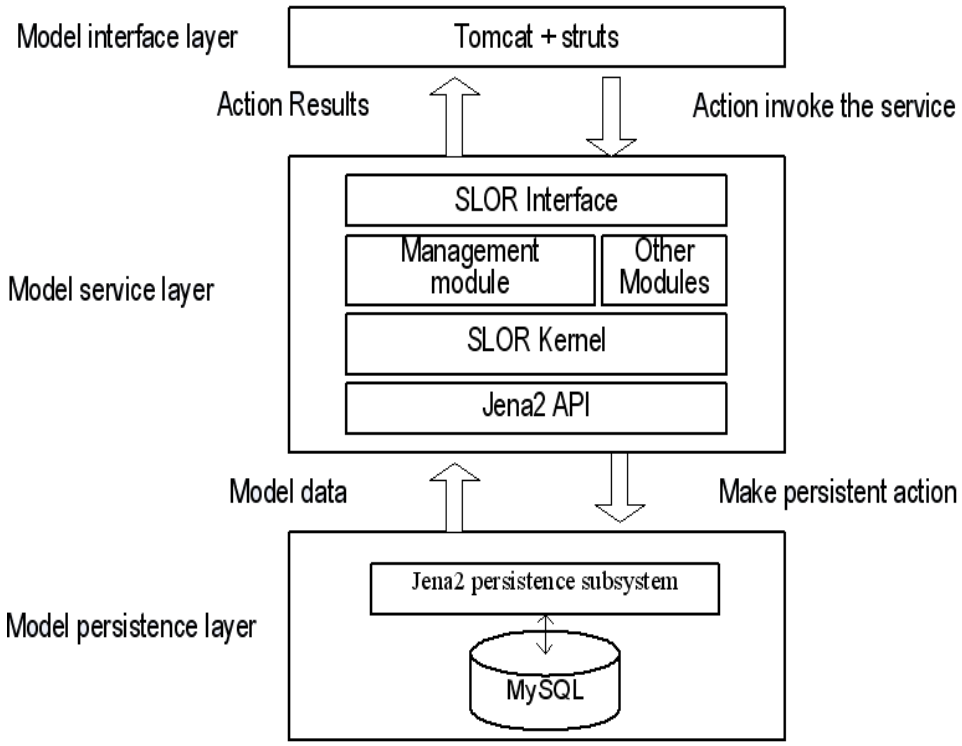

Figura 4.15: SLOR - Arquitectura Prototipo

La secuencia de acciones representada en la figura 4.17 se describe a continuación:

1. Cuando el creador de metadatos hace click sobre el botón "New RLO" de la interfaz del repositorio (figura 4.16), el cliente Web envía la petición NewRlo al servidor TomCat.

2. El Servidor Tomcat llama a la acción Struts "*LearningObject_LOMAction*", que utiliza la interfaz SLOR para invocar la función de creación de un registro individual metadatos LOM.

3. La entidad de intercambio de metadatos entre la interfaz Web y el repositorio SLOR es un ActionForm construido con la tecnología Struts. La acción "*LearningObject_LOMAction*" crea una nueva instancia de la clase Struts "*LearningObject_LOMActionForm*".

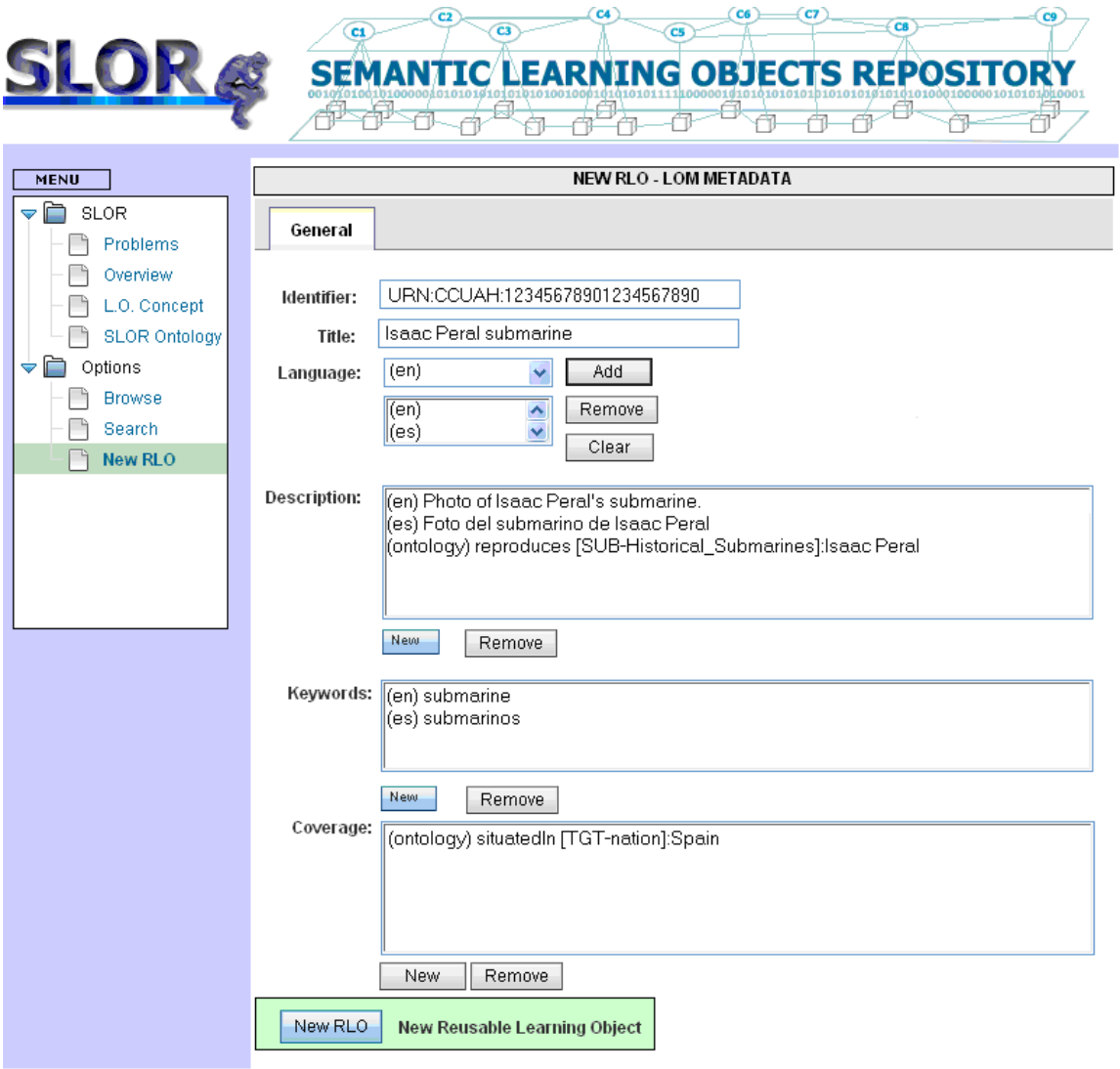

Figura 4.16: SLOR - Creación de metadatos

4. Las entidades tienen una relación con la ontología SLOR. En este caso, la clase "*LearningObject-LOM*" se traduce en código Java a una clase Struts "*LearningObject_LOMActionForm*".

5. La interfaz de SLOR encamina la acción de crear al módulo de administración.

6. Seguidamente se invoca la acción "*createIndividualLearningObject_LOM*".

7. El subsistema de persistencia de Jena almacena en la base de datos MySQL el nuevo individual.

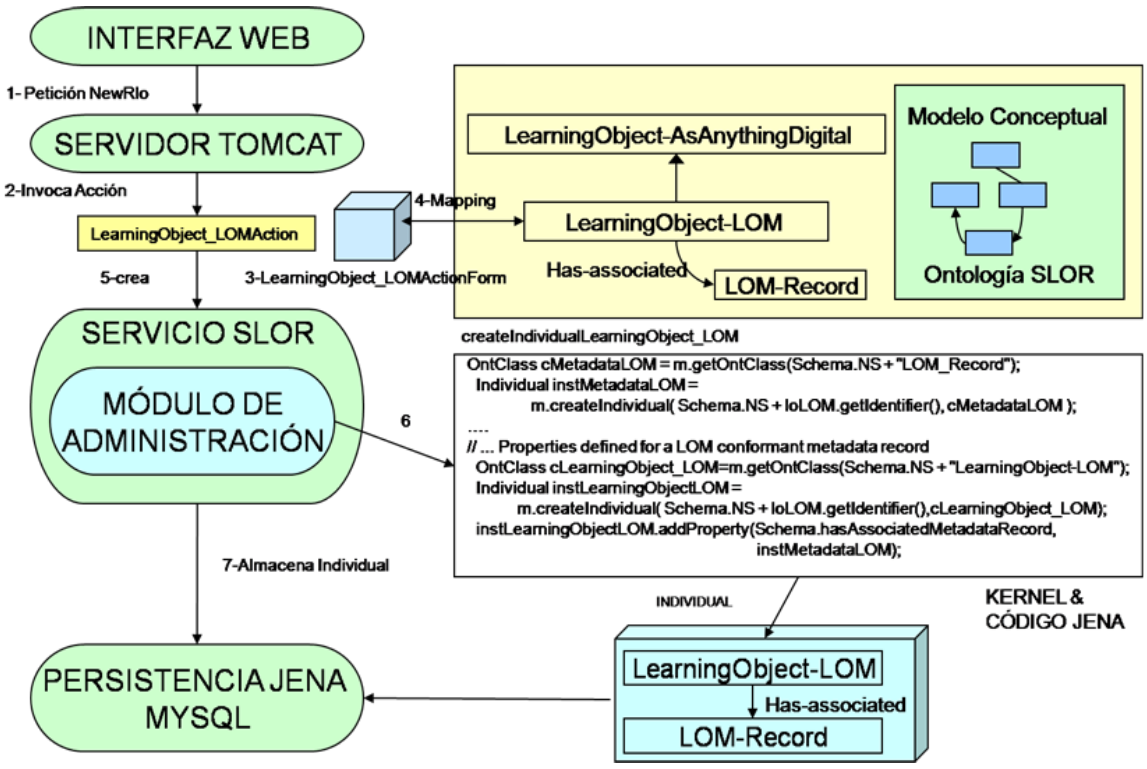

Figura 4.17: Secuencia de inserción de un registro de metadatos en el prototipo SLOR

La función "*createIndividualLearningObject_LOM*" implementa la creación de un nuevo registro de metadatos basado en la especificación LOM. Esta función permite incluir un nuevo registro de metadatos de un objeto de aprendizaje que corresponde

al modelo conceptual de "*LearningObject-LOM*" (en la figura 4.12). El método de creación es parte del módulo de administración de SLOR, se encarga de obtener una referencia a un registro "*LOM_Record*" por la llamada al método "*getOntClass()*". Seguidamente se crea la instancia ("*instMetadataLOM*") de la clase "*LOM_Record*". Invocando el método "*addProperty()*" sobre el individual, se establecen las propiedades del registro de metadatos LOM.

Las clases del núcleo de SLOR están destinadas a ofrecer funcionalidades complejas, como por ejemplo la clase "*KernelProperty*" que implementa funciones de inserción de múltiples propiedades a partir de una lista de objetos dada (método "*setMultipleProperty()*". Finalmente, tras la secuencia de instrucciones de inserción de los campos de los metadatos y el enlace de las propiedades de las conceptualizaciones con las instancias creadas, se crea un individual de la clase "*LearningObject-LOM*" asociado a una instancia de la clase ("*LOM_Record*").

Como ya se ha explicado, la información puede ser enlazada a conceptos de otras ontologías externas al modelo base de conceptualización. En la figura 4.18 se muestra un ejemplo de inserción de una expresión semántica, donde se especifica la ontología y a continuación en una caja de texto se introduce la expresión semántica. Posteriormente, al pulsar el botón "*Insert Semantic Expression*" se analiza la expresión y se procesan los términos introducidos, en caso de no existir alguna propiedad, predicado o término se notifica al usuario los errores de construcción. En caso de pulsarse el botón "*Add Description*" se insertaría como una cadena de tipo "*LangString*" sin ser procesada como una expresión formal.

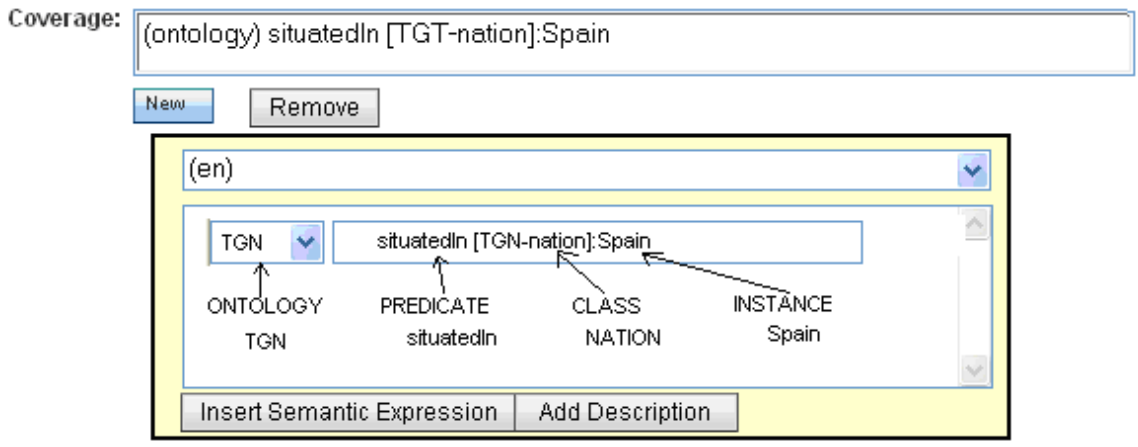

Figura 4.18: SLOR - Enlace de información de los metadatos con elementos de una ontología

Búsqueda semántica de objetos de aprendizaje

En la primera versión del prototipo inicial se implementaron las funciones básicas de búsqueda, útiles para verificar el correcto funcionamiento del modelo y la viabilidad de una posterior implementación de búsquedas más complicadas con expresiones de consulta formadas por términos de diferentes ontologías. En la función "*semanticSearchLearningObject*" se han implementado los mecanismos básicos de recuperación de registros de metadatos a partir de unos criterios simples. Esta función permite buscar instancias de las distintas conceptualizaciones del modelo ontológico, por ejemplo, la recuperación de todos los posibles objetos digitales genéricos "*LearningObject-AsAnyThingDigital*" o los objetos de propósito educacional "*LearningObject-AsAnythingWithEducationalPurpose*". En la figura 4.19 se muestra la interfaz de búsqueda de una consulta simple por tipo de conceptualización de objeto de aprendizaje y el contenido del campo "*title*" de los registros de metadatos. Cuando se pulsa el botón "*Search*" se envía la petición al servidor Tomcat,

y seguidamente se procesa una consulta de tipo RDQL[46] sobre el modelo persistente, obteniendo los resultados para ser mostrados en la tabla.

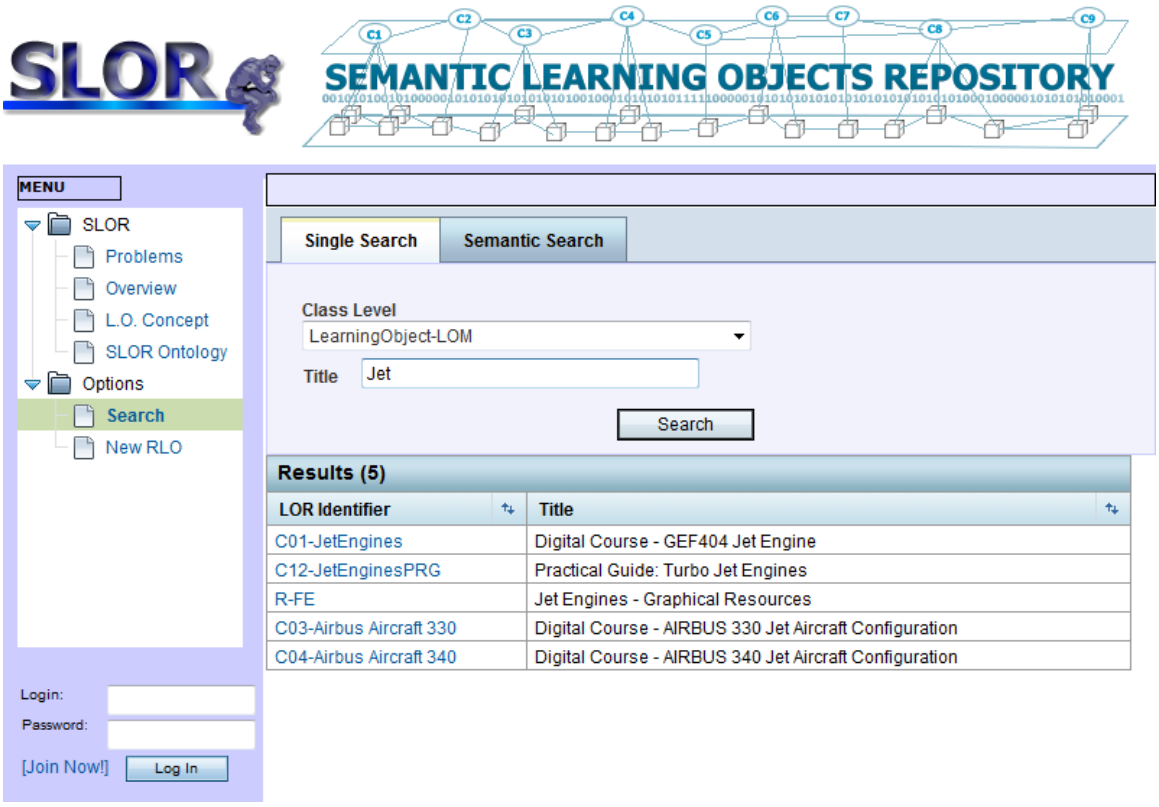

Figura 4.19: SLOR - Búsqueda simple de registros con las conceptualizaciones de la ontología

La ejecución de las consultas sobre el modelo persistente de SLOR mostraron malos tiempos de respuesta debido a algún defecto de la arquitectura. Por ello se decidió evaluar detenidamente los problemas encontrados y plantear otro posible diseño más óptimo.

Debido a esta situación no se decidió profundizar en otro tipo de búsquedas más

[46]La primera versión se implemento en el año 2005.

complejas, ya que era necesario solucionar los problemas encontrados. En este punto termina la primera fase del desarrollo de SLOR. A continuación en la siguiente sección se expone problematica y las soluciones adoptadas para habilitar un modelo general eficiente que permitiese la ejecución de consultas complejas con expresiones semánticas y con tiempos de respuesta viables.

4.4. Problemas encontrados

A continuación se exponen los problemas encontrados que impiden continuar con el desarrollo del prototipo inicial. Se presenta un análisis que dará origen a la propuesta final de diseño del repositorio.

4.4.1. Modos de integración de OpenCyc

Las investigaciones realizadas en la implementación del prototipo revelan la posibilidad de enlazar conceptos de diferentes ontologías dentro de los campos de LOM. De esta forma se ha obtenido un modelo de almacenamiento de objetos de aprendizaje que permite insertar "hechos" dentro de los campos LOM, tratando la base de datos del repositorio como una base de conocimiento sobre la cual se pueden realizar complejos razonamientos e inferencias.

Las ontologías nos proporcionan un lenguaje común capaz de hacer que distintos sistemas software puedan entenderse y procesar "conocimiento humano". Para poder describir de forma lógica conceptos "generales" (cómo "barroco", "renacimiento", "España", "rey", etc..) es necesario utilizar una ontología que normalice estos conceptos y describa sus relaciones. Este tipo de ontologías son las denominadas "Upper Ontologies".

Como ha sido expuesto detalladamente la sección 4.1.4, actualmente OpenCyc es

una de las ontologías más completas dentro del dominio general humano. Posee millones de aserciones de conocimiento y el respaldo de la compañía de Lenat (Cycorp). Las investigaciones realizadas se han centrado en la combinación de la ontología de SLOR con OpenCyc y otras ontologías de conocimiento específico.

Dados los problemas encontrados en la primera aproximación del prototipo, se han realizado dos experimentos para averiguar cual es la mejor forma de integración con el modelo OWL del repositorio, descrito en la sección 4.4.1:

1. Utilizando un servidor de OpenCyc en el que todo se almacena en CycL.

2. Almacenando OpenCyc en una base de datos relacional mediante los mecanismos de persistencia proporcionados por las bibliotecas de programación de la Web semántica (tales como Jena o Sesame), almacenando todo en OWL.

Descripción de los experimentos

1- OpenCyc como servidor de almacenamiento y gestor de ontologías único

En el primer experimento se analizaron las estructuras y elementos utilizados por la base de conocimiento OpenCyc, con los siguientes objetivos:

1. Verificar la posibilidad de integrar cualquier tipo ontología dentro de OpenCyc y realizar inferencias de forma conjunta con toda la vasta base de conocimiento.

2. Estudiar el rendimiento del sistema mediante una batería de consultas que analice los tiempos de respuesta en determinadas situaciones.

Para conseguir el primer objetivo se creo una microteoría "*slor*" y se definió una parte relevante del esquema descrito en la sección en el lenguaje CycL y también desde la librería de acceso Java. Se comprobó que OpenCyc posee un modelo basado en "*HOL*" que abarca un modelo más completo que la aportada por el modelo OWL

basado en lógica de descripciones del repositorio.

En cambio, se encontraron problemas a la hora de analizar la posibilidad de utilizar ontologías del exterior, CycL sólo permite describir relaciones con términos y expresiones descritas dentro de la base de conocimiento. No es posible indicar expresiones del tipo:

```
(prefix eng http://slor.sourceforge.net/ontology/engines.owl)
(#$isa eng:GX025 #$JetEngine)
```

Dónde "*GX025*" es un concepto albergado en la ontología "*eng*" descrita en OWL y "*JetEngine*" un término de la base de conocimiento OpenCyc.

Si se desea realizar este tipo de operaciones es necesario crear un procedimiento de importación de una ontología escrita en OWL al lenguaje CycL para ser incluida dentro de la base de conocimiento OpenCyc. Es viable, pero se han de contemplar los problemas que puedan surgir debido a las posibles actualizaciones que puedan darse de la fuente.

Respecto al segundo objetivo del experimento, para verificar el rendimiento de OpenCyc y contrastarlo con otras pruebas en igualdad de condiciones, se lanzo la ejecución de una serie de operaciones relevantes en la interacción con este tipo de bases de conocimiento, sobre un procesador Intel Quad 2400Mhz, con 4GB de memoria RAM DDR800Mhz con Windows Vista Ultimate, la JDK 1.5 instalada con NetBeans y el Visual Web Pack de desarrollo, y finalmente un Virtual PC ejecutando una máquina virtual Linux con 1 núcleo del procesador dedicado y 1GB de memoria RAM destinado a la ejecución de la base de conocimiento OpenCyc.

El análisis comprende desde el estudio del tiempo medio de localización de un elemento hasta el tiempo medio de ejecución de una inferencia. En el gráfico mostrado en la figura 4.22 se pueden apreciar, al ejecutar una batería de 5 consultas diferentes los tiempos medidos en microsegundos de localización de un elemento (TMLE), de recuperación de los individuales de una clase (TMRI), de recuperación de una clase directa (TMRCD), y de inferencias transitivas sobre relaciones "*isa*" (TMRINF). En los resultados se ha obviado la primera consulta ya que en la primera etapa del proceso se inicia la conexión contra la base de conocimiento OpenCyc, proceso que cuesta unos 5 segundos.

2- Versión OWL 0.78b de OpenCyc integrada dentro del esquema de SLOR gestionado por Jena

El segundo experimento fue planificado con las siguientes tareas:

1. Codificar un método que permita cargar el modelo OWL de OpenCyc directamente en un modelo persistente Jena.

2. Evaluar el rendimiento del modelo gestionado por Jena.

El código relativo a la inclusión de OpenCyc dentro de Jena se muestra a continuación en el siguiente listado de código. Cabe resaltar las instrucciones de la línea 17 donde se establece el generador de modelos persistente asociado a la base de datos MySQL al invocar al método "*createModelRDBMaker*". A partir del generador se puede crear un nuevo modelo (ver líneas 21, 23) o abrirlo si este ya existe (ver línea 26).

```
1 public static Model createOpenCycModel() throws PersistenceException {
2     // Load the Driver
3     try {
4           Class.forName("com.mysql.jdbc.Driver"); // load driver
5     } catch (ClassNotFoundException ex) {
```

```
6             SystemPersistenceException errsp = new
                 SystemPersistenceException();
7             throw (errsp);
8     }
9     String DB_URL2 = "jdbc:mysql://localhost/opencyc"; // URL of
          database server
10    String DB_USER = "root"; // database user id
11    String DB_PASSWD = "senux"; // database password
12    String DB = "MySQL"; // database type
13
14    // Create database connection
15    IDBConnection conn2 = new DBConnection(DB_URL2, DB_USER, DB_PASSWD
          , DB);
16    try {
17         ModelMaker c = ModelFactory.createModelRDBMaker(conn2);
18         Model opencyc;
19         if (!conn2.containsModel("opencyc")) {
20             //Create Model OpenCyc
21             opencyc = c.createModel("opencyc");
22             System.out.println("CONSTRUCCIÓN DEL MODELO");
23             opencyc.read("http://www.cyc.com/2004/06/04/cyc/#");
24          } else {
25             //Open Model OpenCyc
26             c.openModel("opencyc");
27          }
28          opencyc = ModelRDB.open(conn2, "opencyc");
29          return (opencyc);
30    }catch (RDFRDBException ex1) {
31            ConnectPersistenceException errcon =
32                      new ConnectPersistenceException();
33            throw (errcon);
34    }
35}
```

Esta operación tardó cinco minutos y 12 segundos sobre el sistema utilizado para realizar las pruebas OpenCyc, con el servidor de bases de datos MySQL instalado.

Después de la ejecución del método se analizó el contenido de tabla "*jena_stmt*" para indagar sobre el modo de traducción CycL - OWL realizada. Se descubrió que la traducción de los predicados es directa a propiedades OWL de tipo "*ObjectProperty*" con el inconveniente de no quedar especificado en la versión OWL ni la aridad ni el rango de cada uno de los argumentos descritos en CycL. En la figura 4.20 se representa el contenido de la tabla "*jena_stmt*" en relación al predicado "*boyfriend*", como puede apreciarse se especifican los valores de las relaciones "*subPropertyOf*" con otros predicados manteniendo la jerarquía establecida de OpenCyc pero no se indica de ninguna forma la aridad y el tipo de los argumentos.

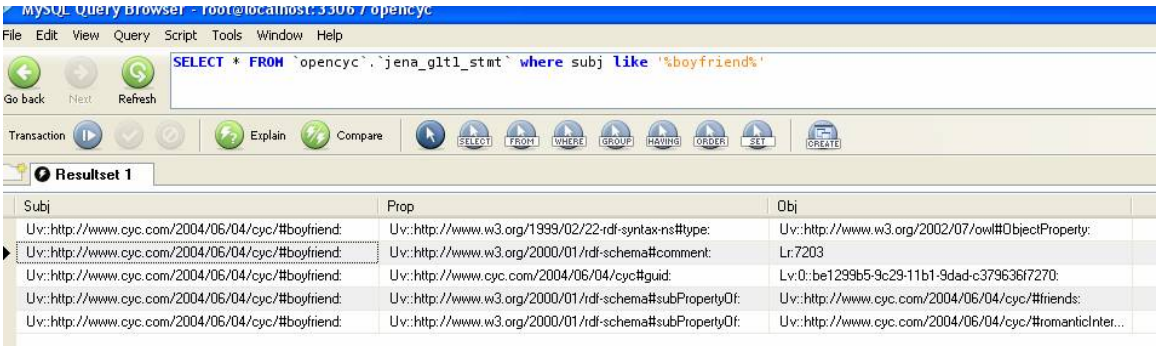

Figura 4.20: boyfriend(PERSON MALEHUMAN): predicado CycL convertido a una propiedad OWL ObjectProperty dentro de un modelo persistente de Jena.

Al igual ocurre con la propiedad "*borderOf*" elegida al azar, en la figura 4.21 se muestra el contenido de la tabla "*jena_stmt*" en relación a los triples definidos pertenecientes a la versión OWL 0.78b de OpenCyc.

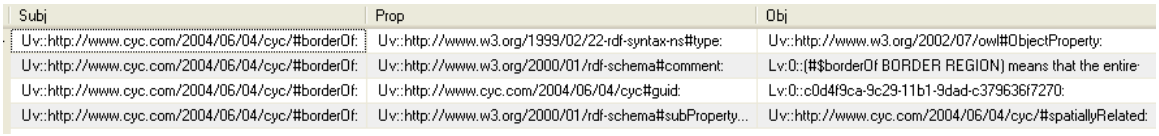

Figura 4.21: borderOf(BODER REGION): predicado CycL convertido a una propiedad OWL ObjectProperty dentro de un modelo persistente de Jena.

Una vez establecido el modelo relacional, para cumplir el segundo objetivo definido en este experimento se creo una batería de operaciones de búsqueda y consultas similar a la analizada en el primer experimento.

Resultados

Los resultados analizados en el primer experimento indican la posibilidad de integrar el modelo de SLOR, pensado apriori en OWL, en CycL con posibilidad de ofrecer un buen rendimiento. En cambio, como gran inconveniente a resaltar, no es posible integrar de forma directa (sin ningún proceso de importación) otras ontologías escritas en OWL, por lo tanto la capacidad de descripción queda supeditada a los conceptos almacenados en OpenCyc, así como a todas las ontologías escritas en CycL. Sin embargo, en el estudio [SL02] se permite la posibilidad de relacionar términos de CycL con otros términos escritos en otras ontologías, aún así el motor de inferencia de CycL no puede procesar el conocimiento expresado en otros lenguajes de otras ontologías.

En el segundo experimento, que sigue el modelo del repositorio actual hasta ahora comentado, se consigue un buen nivel de integración gracias al uso de los lenguajes basados en XML de la Web semántica. Cualquier tipo de ontología escrita en un lenguaje de esquemas u ontologías de la Web semántica es capaz de integrarse dentro de SLOR para incrementar la capacidad de expresividad en la descripción de los registros de metadatos. El principal problema encontrado es el bajo rendimiento la hora de tratar grandes cantidades de metadatos. La gran cantidad de términos descritos en OpenCyc, así como la multitud de datos manejada por un repositorio de este tipo, ha revelado un serio problema a la hora de plantear este modelo como diseño definitivo de SLOR. También hay que añadir los inconvenientes proporcionados por la versión 0.78b OpenCyc escrita en OWL, ya que carece de muchos términos de los publicados

en la versión CycL y además no se han utilizado aserciones escritas en lógica de descripciones para la declaración de un modelo más completo. Tampoco queda definido el dominio y rango en las propiedades.

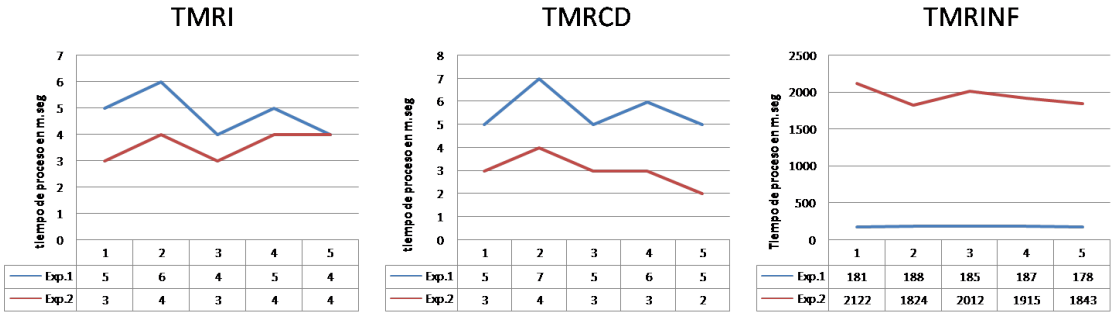

Figura 4.22: Resultados de los experimentos realizados

Conclusiones y soluciones

Los resultados demuestran una mejora sustanciable del rendimiento en los procesos de inferencia frente a los aportados por el modelo persistente OWL gestionado por Jena. Ante estos resultados, ¿se han de almacenar todos los registros del repositorio en la base de conocimiento OpenCyc utilizando el lenguaje CycL como herramienta para la definición de las nuevas aserciones sobre la base de conocimiento? Esta es una de las preguntas clave necesarias para dar soporte a la nueva arquitectura del diseño, como cabe esperar no ha sido fácil obtener la respuesta.

Si optamos por elegir una arquitectura de almacenamiento basada en OpenCyc, la solución posible a plantear es la traducción del modelo OWL del repositorio al lenguaje CycL para ser introducido dentro de la base de conocimiento OpenCyc como una microteoría. Se ha de tener en cuenta que OpenCyc posee un modelo cerrado, aunque se esta trabajando en aportar nuevos trabajos de investigación basados en

la integración de conocimiento, a día de hoy no es posible enlazar y definir nuevas aserciones o reglas utilizando otros términos de otras ontologías descritos en lenguajes universalmente aceptados y definidos en estandares (como OWL). Esta característica impide realizar inferencias más complejas ya que a día de hoy no existe una correlación entre conceptos de distintas ontologías. Es decir, si se utiliza el concepto "humano" en una ontología definida en OWL, no existe ninguna relación con el mismo concepto descrito en OpenCyc, por lo que todas las aserciones y reglas definidas sobre el concepto en OpenCyc no son utilizadas en la ontología de OWL y viceversa.

Por otro lado, la versión de OpenCyc escrita en OWL carece de muchos términos que si están presentes en la versión CycL. Además no se hah utilizado en esta versión aserciones escritas en lógica de descripciones para declarar un modelo más completo, tampoco queda definido el "dominio" y el "rango" en las propiedades de la ontología. En el gráfico 4.22 se muestran los resultados negativos con tiempos de respuestas muy altos en la ejecución de simples inferencias. Este problema es debido a la gran cantidad de aserciones proporcionada por OpenCyc almacenadas sobre una base de datos relacional, y a la carencia de un proceso de traducción claro a un sistema persistente que permita aprovechar al máximo las características de optimización de modelos, como el uso de las tablas de propiedades. A la hora de realizar una inferencia, el sistema de persistencia basado en tecnologías de la Web semántica necesita recuperar múltiples aserciones de la base de datos para poder llevar a cabo la inferencia.

Por tanto, se elige como solución una opción mixta que mejore los tiempos de repuesta de inferencia y no rompa con la filosofía de la flexibilidad y la gran potencia semántica definida en el modelo formal de SLOR. La propuesta consiste en separar la base de conocimiento OpenCyc del modelo OWL de SLOR, ya que esta la utilizamos normalmente para ejecutar consultas que nos permitan obtener los elementos de la base de conocimiento, para posteriormente generar una expresión formal que se

incluirá en algún registro de metadatos del repositorio. De esta manera se mantiene la base de conocimiento OpenCyc como sistema de consulta y una base de datos relacional gestionada por un sistema persistente que mantendrá el modelo OWL de SLOR.

4.4.2. Problemas en la interfaz

A la hora de insertar un nuevo registro de metadatos sobre SLOR es necesario rellenar múltiples campos, así como escribir expresiones complejas para generar aserciones que permitan enlazar conceptos con otras ontologías. El tiempo medio para rellenar un perfil general de LOM es alto. Además, cada especificación proporciona un gran conjunto de campos para describir un objeto de aprendizaje. Estudios recientes sobre la creación automática de metadatos [VM05] plantean nuevos caminos que permitirán agilizar el trabajo del creador de metadatos, además de hacer mucho más usable este tipo de repositorios.

Solución

La solución posible a plantear consiste en crear una interfaz más amigable con un asistente que ayude al usuario en el proceso de crear las expresiones semánticas. Utilizaremos las ideas planteadas por la tecnología Intellisense de Microsoft[47].

4.5. Iteración y adaptación del diseño original

Debido a los problemas descritos anteriormente, se ha propuesto un nuevo desarrollo para obtener un modelo final de SLOR más desacoplado, rápido y eficaz, así como una nueva interfaz más usable a la hora de insertar los metadatos de los objetos de aprendizaje.

[47] http://msdn2.microsoft.com/es-es/library/ms174184.aspx

- **Integración de AJAX y Java Server Faces:** estas tecnologías nos permiten construir interfaces más "usables". La programación web requiere de una interactividad entre un cliente y un servidor. AJAX [Gar05] es una tecnología que aumenta la comunicación entre el servidor y el cliente mediante mensajes XML enviados entre ambos, en lugar de los GET o POST realizados hasta el momento para actualizar los datos en una interfaz. Por otro lado, Java Server Faces[48] proporciona un conjunto de utilidades que permiten construir páginas web de forma rápida y cómoda para el programador.

- **Independencia del framework semántico:** para no depender directamente de un framework que permita tratar las tecnologías de la Web semántica, se ha trabajado en la integración del patrón de software "abstract factory" [GHJV95] para la construcción del núcleo de SLOR. Las búsquedas e inferencias realizadas sobre OpenCyc pueden construirse directamente sin tener que realizar razonamientos sobre el sistema persistente en la base de datos relacional.

- **Federación de repositorios:** la federación de los distintos repositorios de objetos de aprendizaje existentes en la web, permite la posibilidad de realizar consultas distribuidas, así como la inserción y la no duplicación de datos en distintos repositorios distribuidos. Este tema proporciona una nueva línea de investigación sobre los repositorios de objetos de aprendizaje.

Gracias a los estudios realizados, en esta investigación se ha conseguido implementar el nuevo diseño del repositorio SLOR creando potentes mecanismos de búsqueda e intercambio de conocimiento sobre los metadatos de los objetos de aprendizaje.

[48] Tecnología Java Server Faces: http://java.sun.com/javaee/javaserverfaces/

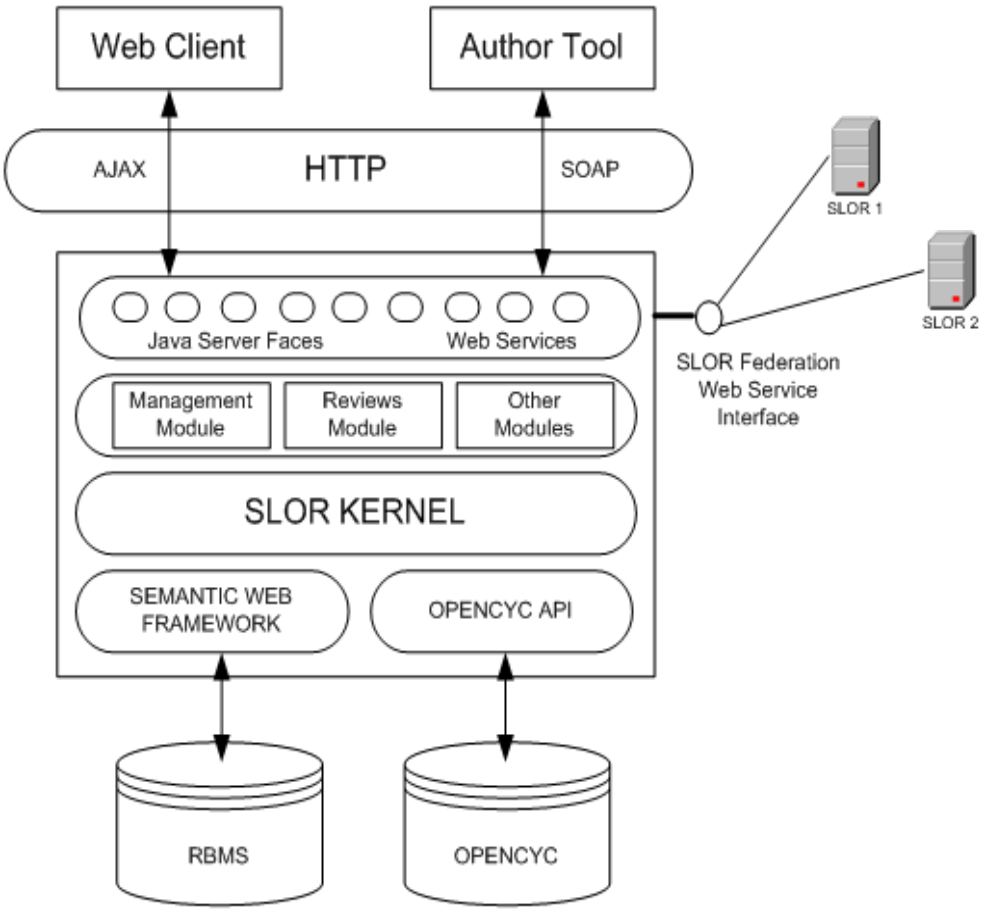

Figura 4.23: Diseño adaptado de SLOR

4.6. Resumen

En este capítulo se ha descrito el largo proceso que ha conseguido perfilar los principios de diseño del repositorio semántico de objetos de aprendizaje. Tras la elaboración de una versión inicial del prototipo se ha conseguido detectar una serie de carencias, las cuales han permitido elaborar un diseño más flexible y adaptado a la realidad tecnológica de la situación actual dada en las herramientas de la Web semántica. En el siguiente capítulo se describe de forma detallada el diseño y la implementación de los componentes más relevantes de la arquitectura del repositorio.

Capítulo 5

Descripción Técnica: Repositorio Semántico de Objetos de Aprendizaje

One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man.

Elbert Hubbard

Como parte integral de la investigación se ha realizado la implementación de un prototipo de repositorio de objetos de aprendizaje que sigue los principios de diseño expuestos en el capítulo anterior. Este capítulo ofrece una descripción técnica de los componentes más relevantes del prototipo de repositorio semántico, desde la visión general de la arquitectura hasta los detalles de los componentes más importantes.

5.1. Arquitectura y principios de diseño

Tras las pruebas iniciales realizadas con los diseños anteriores se llegó al resultado obtenido en la última iteración de la arquitectura presentada en la sección 4.5 del capítulo anterior. La arquitectura tiene en cuenta todos los problemas encontrados, considerando los diferentes tipos de arquitecturas software (MVC, Java Server Faces,

SOA) y nuevos paradigmas de programación web “AJAX” con las bibliotecas de programación basadas en las tecnologías de la Web semántica.

A partir de este modelo de arquitectura se han implementado un conjunto de componentes especializados en procesar una serie de funciones descritas en las interfaces expuestas. Cada componente expone un grupo de interfaces y requiere comunicación con las interfaces de otros componentes. La operación de asociación entre la interfaz expuesta de un componente y la interfaz requerida se denomina ensamblado.

El comportamiento e implementación de los componentes queda definido en una o varias clases, independientes del resto de clases del repositorio. Aquellos componentes que poseen un carácter de comunicación distribuida requieren ser desplegados en un entorno de ejecución adecuado, un ejemplo de ello sería un servicio web que exponga una interfaz IMS-DRI para la interoperabilidad entre repositorios.

En la figura 5.1 se representa el diagrama de los componentes principales del repositorio. Organizados en capas por funcionalidad según el diseño expuesto en la sección 4.5 del capítulo anterior, se distinguen las siguientes categorías de componentes:

- Componentes de persistencia: su función principal es abstraer el sistema de persistencia final (marco de trabajo semántico y sistema de almacenamiento RDBMS o OODBMS) y mantener todos los registros de metadatos bajo el modelo flexible descrito en el capítulo anterior (ver sección 4.2.2).

- Componentes del núcleo: proporcionan la funcionalidad básica para almacenar, recuperar y realizar búsquedas e inferencias sobre grandes conjuntos de registros de metadatos. Requiere de los componentes de persistencia para realizar la comunicación con el sistema final.

- Componentes de tipo módulo: proporcionan una funcionalidad específica, destinada al almacenamiento u otras operaciones como seguridad o gestión de revisiones aportadas por los profesionales. Cada módulo que requiere interacción por parte de los usuarios aporta un conjunto de interfaces específicas. Por ejemplo, un módulo de gestión de registros IEEE LOM aporta un conjunto de interfaces para la inserción y gestión (búsqueda, borrado y modificación) de los registros de metadatos.

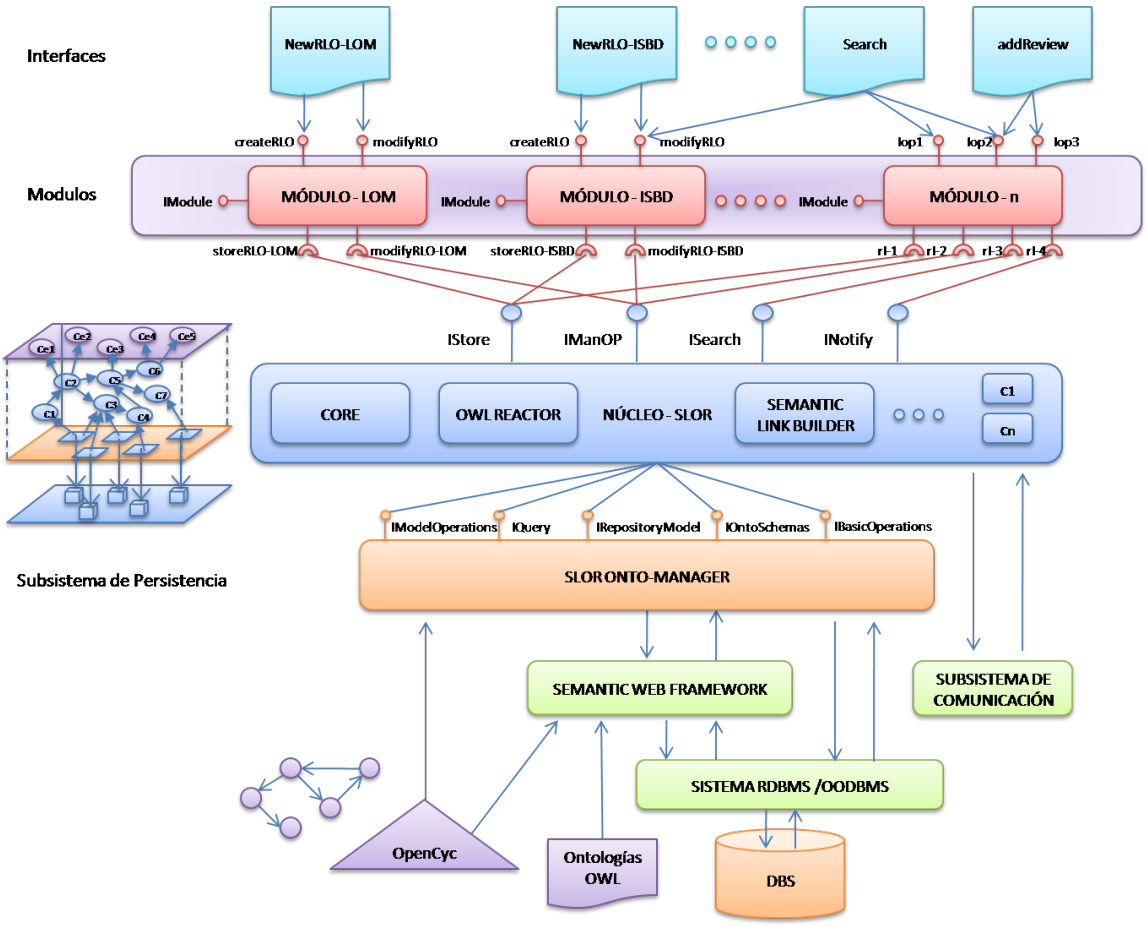

Figura 5.1: Diagrama general de componentes de SLOR

A continuación se describen los principios de diseño basado en componentes independientes representado en la figura 5.1. Al finalizar la descripción general, en el resto

de secciones del capítulo se detallaran los componentes del sistema comenzando por la capa más interna, el subsistema de persistencia y el núcleo hasta los componentes de la capa más externa.

5.1.1. Independencia del marco de trabajo semántico

Debido al número diferente de bibliotecas de programación que implementan las tecnologías y estándares aportados por la Web semántica, se ha decidido crear un modelo independiente que establece las diferentes funciones de acceso a un repositorio de modelos basados en OWL o RDF. El modelo esta inspirado en los principios de diseño de generación de clases de la herramienta Protégé.

El núcleo de SLOR posee varios componentes que hacen uso del subsistema de persistencia que hemos denominado "*SLORONTO-Manager*" a través de las interfaces "*IRepositoryModel*", "*IModelOperations*", "*IQuery*", "*IOntoSchemas*" e "*IBasicOperations*" representadas en la figura 5.1. Las operaciones relativas a la conexión e inicialización de los modelos albergados en el repositorio se realizan a través de la interfaz "*IRepositoryModel*". La interfaz "*IModelOperations*" proporciona las acciones de creación, modificación y borrado de los elementos del modelo (clase, propiedad o individual). Las consultas semánticas que ejecutan determinados procesos de inferencia se realizan a través de las operaciones proporcionadas por la interfaz "*IQuery*". Para la gestión de los diferentes esquemas proporcionados por los módulos de gestión de SLOR (LOM, IMSLD, ISBD, etcétera) el núcleo de SLOR utiliza la interfaz "IOntoSchemas". Finalmente, la interfaz "*IBasicOperations*" es utilizada para las operaciones más comunes con la base de conocimiento OpenCyc. En la sección 5.2 se detalla con más profundidad los componentes más importantes del sistema de persistencia.

5.1.2. Construcción modular

Como ya se ha comentado en el capítulo anterior, para separar las distintas partes funcionales con vistas a facilitar en el futuro el añadir otras nuevas en un entorno de desarrollo abierto, se ha decidido incluir una especificación común genérica de módulo. Un módulo posee una descripción de las operaciones que puede realizar, la relación de dependencias entre otros módulos y un descriptor de despliegue sobre el sistema.

Los módulos exponen un conjunto de operaciones accesibles desde las interfaces del usuario (páginas web o formularios de escritorio), aplicaciones externas (mediante Servicios Web) o desde otros módulos si existe alguna dependencia. En la figura 5.1 se puede ver como el módulo LOM proporciona a la interfaz "*NewRLO-LOM*" las operaciones "*createRLO*" y "*modifyRLO*", que a su vez pueden ser utilizadas por un módulo de búsqueda que requiera de estas operaciones, u otro dedicado a la interoperabilidad con otras aplicaciones exteriores. También se han representado las interfaces más importantes del núcleo utilizadas por los módulos; "*IStore*" contiene las operaciones relativas al almacenamiento de los diferentes registros de metadatos soportados, "*IManOP*" proporciona las operaciones de administración de los registros albergados (tales como eliminar o modificar alguna propiedad), "*ISearch*" aporta las operaciones de búsqueda semántica dentro del repositorio y finalmente "*INotify*" es utilizada para invocar los servicios de notificación interna entre usuarios.

En la sección 5.4 de este capítulo se describe en mayor detalle la estructura y programación de los módulos de SLOR.

5.1.3. Comunicación con otras aplicaciones y repositorios

Para promover la reutilización de los materiales didácticos referenciados en los metadatos de un repositorio, es necesario proporcionar un conjunto de herramientas

de acceso a través de determinados protocolos de comunicación existentes. Desde el punto de vista de las tecnologías de comunicación distribuida (comentadas en la sección 2.5.1 del capítulo 2), se ofrece la posibilidad de crear componentes distribuidos de comunicación entre distintos repositorios y/o aplicaciones.

En el diseño de SLOR se ha decidido crear un módulo encargado de proporcionar un conector de federación entre repositorios y de acceso externo por parte de aplicaciones tales como sistemas gestores de aprendizaje, búscadores o herramientas de autor. En la sección 5.5.2 se describe en mayor detalle los mecanismos de comunicación distribuida contemplados en el diseño de SLOR.

5.2. Subsistema de persistencia gestor de ontologías

Dado el carácter experimental del prototipo, se ha decidido implementar un grupo de componentes que permita definir un conjunto de operaciones sobre un modelo lógico y hacer independiente el repositorio del marco de trabajo semántico. En la figura 5.2 se muestran los componentes implementados y las relaciones de dependencia entre sus interfaces.

Como puede apreciarse, el paquete superior "*SLOR OntoManager*" engloba todas las funcionalidades de bajo nivel que pueden realizarse sobre el modelo semántico del repositorio. A continuación se describe cada uno de sus componentes:

- SLOR Model: componente que encapsula toda la funcionalidad transparente de gestión del modelo de datos. Contiene todas las clases abstractas e interfaces necesarias para manipular en código las clases e individuales del modelo. El diseño de cada una sigue los principios de generación del código de Protégé [RTJ05].
- Persistent Subsystem: paquete que engloba todo el sistema de persistencia del

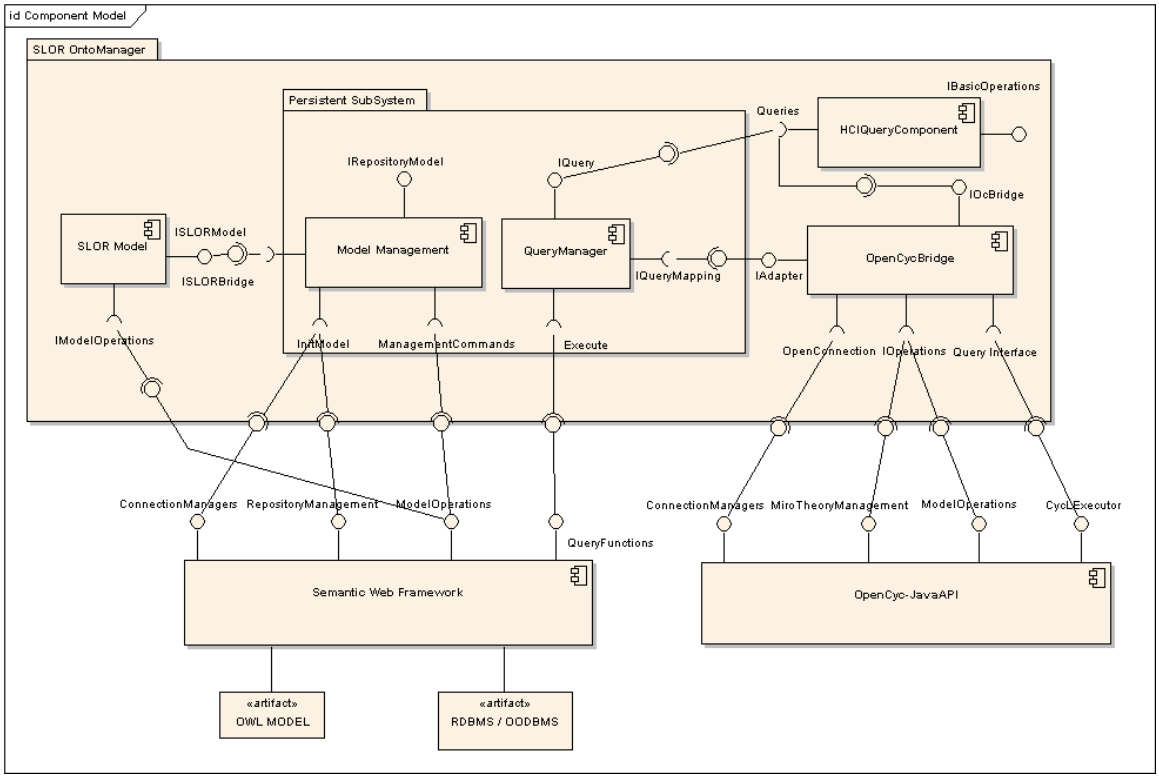

Figura 5.2: Diagrama de componentes del subsistema de persistencia de SLOR

modelo del repositorio. Sus componentes son "*Model management*", encargado de gestionar todas las operaciones que pueden realizarse (ej. crear un individual, borrar una clase, etcétera) y "*QueryManager*", responsable de la creación, compilación y ejecución de las consultas sobre el modelo.

- OpenCyc Bridge: componente que gestiona el acceso a la base de conocimiento OpenCyc. Define las principales operaciones necesarias para interactuar con el subsistema de persistencia del modelo semántico del repositorio.

- HCIQueryComponent: define las operaciones de consulta básicas necesarias para la elaboración de interfaces que requieran del uso de un asistente inteligente de escritura de expresiones semánticas (llamado *semantic intellisense*).

- Semantic Web Framework: representa cualquier marco de trabajo semántico al que dé soporte el repositorio, por ejemplo Jena o Sesame.

- OpenCyc-JavaAPI: biblioteca utilizada por el módulo "*SLOROntoManager*" para el acceso a la base de conocimiento OpenCyc.

5.2.1. Evaluación de sistemas de almacenamiento persistente

A continuación se exponen las alternativas analizadas de almacenamiento persistente de modelos semánticos basados en OWL/RDF que fueron estudiadas de cara a ser utilizadas en el repositorio SLOR. No se sabía cuál usar, así que hemos evaluado las más populares y tomado una decisión en función de la estructura y soporte para el manejo versátil de distintos modelos de ontologías.

JENA

El subsistema de persistencia de Jena para grafos RDF utiliza una base de datos relacional a través de una conexión JDBC. Las experiencias con Jena1 (primera versión) muestran que el uso de un esquema sin normalizar reduce los tiempos de respuesta [WSKR03]. La segunda versión de Jena (la actualmente denominada Jena2) compensa el espacio por el tiempo. Tanto los recursos URI como los valores simples de literales son almacenados directamente en la tabla de sentencias ("*stmt*", en la figura 5.3 "*jena_g1t1_stmt*"). Para distinguir los literales y las URIs, los valores de las columnas son codificados con un prefijo (Uv:: para un recurso o Lv:: de un literal) que indica el tipo de valor. Si la longitud de un literal excede de un cierto valor, este es almacenado en una tabla separada llamada "*jena_long_lit*". Este modelo proporciona la ventaja de realizar varias operaciones de búsqueda sin necesitar operaciones de reunión (join), al almacenar todos los valores directamente en la tabla de sentencias. Al albergar todas las tripletas del modelo RDF dentro de una tabla

pueden ocurrir determinados problemas de gestión y operación, por lo que Jena2 proporciona varias opciones para reducir el espacio. Entre estos, merece la pena citar la compresión de espacios de nombres mediante la definición de un prefijo en la tabla "*jena_long_uri*" para utilizarlo en la tabla "*stmt*" como referencia a espacio de nombres, o evitar las repeticiones de múltiples referencias dadas en un grafo RDF, mediante el almacenamiento del literal referenciado en la tabla "*jena_long_lit*", o utilizando las tablas de propiedad [Wil06].

Table Name	Engine	Rows	Data length	Index length
jena_g1t0_reif	InnoDB	0	16 kB	32 kB
jena_g1t1_stmt	InnoDB	229	64 kB	64 kB
jena_graph	InnoDB	1	16 kB	0 B
jena_long_lit	InnoDB	3	16 kB	16 kB
jena_long_uri	InnoDB	0	16 kB	16 kB
jena_prefix	InnoDB	0	16 kB	16 kB
jena_sys_stmt	InnoDB	60	16 kB	32 kB

Figura 5.3: Tablas utilizadas por Jena2 en un sistema gestor de bases de datos relacional

SESAME

Sesame [BKvH01] proporciona dos formas diferentes de almacenar RDF en bases de datos, bajo esquemas fijos relacionales o bases de datos orientadas a objetos.

- Esquema fijo relacional: las propiedades básicas descritas en la especificación RDF son transformadas en tablas (ver figura 5.4). Este método utiliza un esquema fijo, que constituye una ventaja en el rendimiento respecto a los cambios realizados a menudo en un modelo RDF. Para reducir tanto la sobrecarga como el espacio en la base de datos, todos los recursos y literales son codificados con un identificador asociado. El valor de la columna "*is_derived*" es utilizado para indicar si un elemento del modelo deriva de otro.

- Esquema relacional de objetos: el rendimiento de Sesame utilizando sistemas de bases de datos orientados a objetos ha sido mostrado en diversos artículos [BKvH02]. Estos estudios coinciden en la misma conclusión: se ha detectado un bajo rendimiento debido a que el sistema de base de datos crea una tabla cada vez que una nueva clase o propiedad es añadida, por lo tanto en escenarios donde el esquema cambie a menudo, la correspondencia de modelos RDF no es óptima sobre un sistema gestor de base de datos orientado a objetos.

Un estudio realizado con PostgreSQL [BKvH02] utiliza un enfoque distinto (similar a Jena) en el que todas las sentencias RDF son insertadas en una única tabla con tres columnas: "Subject, Predicate, Object". En escenarios donde el esquema cambia a menudo, este enfoque es preferible al esquema relacional de objetos.

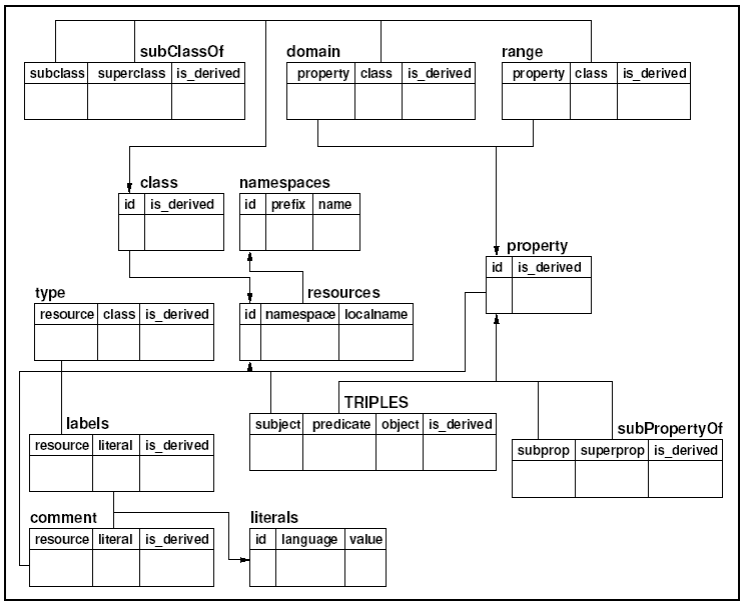

Figura 5.4: Tablas utilizadas por Sesame en un SGBD

ORACLE

Los estudios realizados con el framework de persistencia Sesame ponen de manifiesto el bajo rendimiento del almacenamiento RDF sobre bases de datos orientadas a objetos. El método de almacenamiento RDF proporcionado por Oracle aporta una mejora considerable a las técnicas de persistencia RDF mencionadas. El sistema de base de datos Oracle utiliza un tipo de objeto llamado "*SDO_RDF_TRIPLE_S*", incluido en el nivel superior del modelo NDM (Oracle Spatial Network Data Model). El modelo de almacenamiento NDM esta especializado en el almacenamiento, administración y análisis de grafos. En Oracle, las tripletas RDF son analizadas y almacenadas como entradas en las tablas del NDM "*node$*" y "*link$*". Como características a destacar: las tripletas son almacenadas en la base de datos por cada reunificación, y sólo se almacena una única vez un nodo RDF. La aplicación de estas características reduce el espacio ocupado en la base de datos y el tiempo de proceso necesario para ejecutar una consulta. Sin embargo, es un modelo bastante complejo para gestionar la información RDF debido al requisito, para los usuarios, de llevar a cabo un número considerable de operaciones de bajo nivel.

PROTÉGÉ

Las formas de almacenar un esquema RDF/OWL compensan en una balanza las características de espacio, tiempo de ejecución de consultas, tiempo de modificación, borrado e inserción y finalmente el tiempo de modificación del esquema (clases, propiedades, relaciones y restricciones). En un lado de la balanza se encuentra la modificación del esquema del modelo, que ha de ser flexible y lo menos costosa posible en términos de rendimiento. Como se ha visto, en los sistemas de almacenamiento persistente basados en esquemas flexibles relacionales de objetos la creación de una nueva clase o propiedad genera una nueva tabla. También cuando se realiza una consulta es necesario combinar múltiples tablas para obtener un resultado. Tanto la

modificación del esquema como la ejecución de varias consultas diferentes generan un coste elevado en este tipo de solución. En cambio, por otro lado existe la alternativa de albergar todo en un esquema fijo, formado por un conjunto de tablas (como las especificadas en Jena, Sesame y Oracle) que permite almacenar toda la estructura de un grafo RDF/OWL. En este tipo de solución se diferencia el modo de gestionar los elementos del modelo que deja en el otro lado de la balanza la optimización de consultas y modificación del modelo basado en el almacenamiento de tripletas absoluto. Este es el enfoque utilizado por Jena y Oracle en el que una tabla es la que almacena toda la estructura del modelo y esta puede ser simplificada con el apoyo de otras tablas. El otro modo de almacenamiento utiliza tripletas por referencias en el que existe una tabla que almacena todas las estructuras y además se apoya en otras tablas que permiten obtener mejores tiempos de respuesta en consultas, como en el modelo de Sesame que utiliza las tablas "*subPropertyOf*" y "*subClassOf*". Según este mecanismo, la modificación del modelo supone un gasto de cómputo adicional para reestructurar las tablas que permiten optimizar las consultas.

Existe otro enfoque aportado por Protégé que utiliza una arquitectura genérica de almacenamiento de modelos de ontologías basada en CLOS MOP(Common Lisp Object System - MetaObject Protocol) [Pae93] [KGdRB91] y el patrón de diseño llamado *Dynamic Object Model* [RTJ05].

La característica más importante que nos ha llevado a la elección de "*Protégé*" para el desarrollo del subsistema persistente "*SLOROntoManager*", es la flexibilidad aportada por el diseño de la biblioteca de programación a la hora de almacenar cualquier tipo de modelo RDF/OWL sobre un sistema de gestión de base de datos. La arquitectura esta preparada, en comparación con el resto de bibliotecas analizadas, para incluir otros tipos de almacenamientos persistentes utilizando un conjunto de

operaciones comunes (declaradas en las interfaces que serán presentadas en la siguiente sección). También se mejora la edición y simplificación de los errores cometidos en programación a la hora de acceder al modelo gracias al uso del editor Protégé.

Los modelos de ontologías que se almacenan bajo esta arquitectura utilizan una tabla central que permite albergar todas las entradas de clases "*:FRAME*", propiedades "*:SLOT*" y restricciones "*:FACET*". En la figura 5.5 se muestran las tablas que almacenan el modelo de SLOR. La tabla "*model*" es la más importante, permite albergar toda la definición de objetos existentes en el modelo SLOR basada en CLOS MOP. El resto de tablas con el sufijo "*_includes*" sirven para referenciar otros modelos incluidos ("*model_includes*") u otras bases de conocimiento ("*bdslor_includes*") o en un nivel general ("*_includes*"), finalmente la tabla "*n_users*" es la única separada del modelo de persistencia RDF/OWL ya que alberga los datos relevantes de los usuarios gestionados por el módulo correspondiente explicado en la sección 5.3.3.

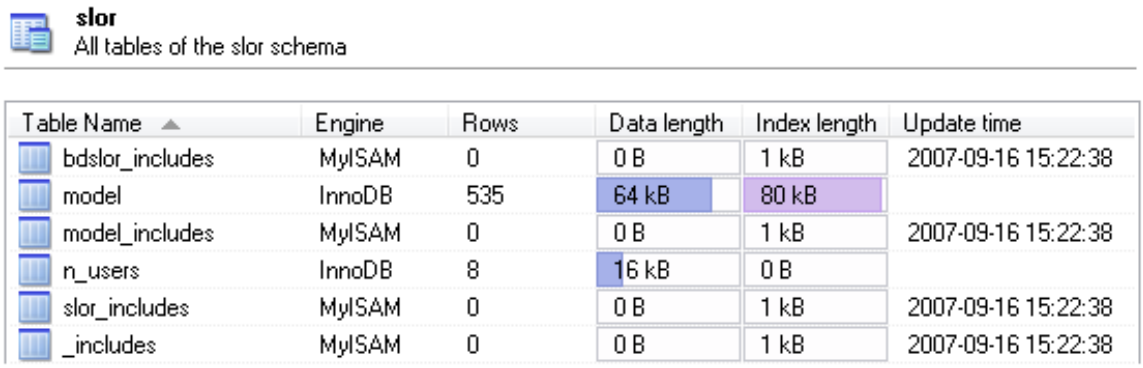

Figura 5.5: Tablas utilizadas por Protégé en un sistema gestor de bases de datos relacional

La tabla "*model*"[1] alberga los campos necesarios para almacenar un modelo OWL/RDF. La figura 5.6 muestra como queda almacenado una instancia de la clase

[1]El código de generación de la tabla que alberga un modelo se encuentra en el método "createTable()" de la clase "*edu.stanford.smi.protege.storage.database*".

"*LearningObject- AsAnythingDigital*" del modelo OWL de SLOR. El elemento se representa con los siguientes campos de la tabla "*model*":

- "*frame*": identificador de la declaración. Los identificadores utilizados por el sistema poseen un valor menor a 10000 (valor "*INITIAL_USER_FRAME_ID*"). En la clase "*edu.stanford.smi.protege.model.Model.java*" están especificados los identificadores reservados. En la figura 5.6 se ve representada la declaración de la clase "*LearningObject-AsAnythingDigital*" con el identificador 11557.

- "*frame type*": especifica el tipo de declaración.

- "*slot*": identificador de la ranura asociada a la declaración. En el ejemplo utilizado, el valor 2002 hace referencia al slot "*:NAME*", el valor 2004 al slot "*:DIRECT_SUPERCLASSES*" que especifica las clases de las que hereda la declaración, también se especifican las clases derivadas directas con el valor 2005 que hace referencia al slot "*:DIRECT_SUBCLASSES*", el valor 2006 indica el slot tipo de declaración "*:DIRECT_TYPES*" 9009 para las propiedades y 9008 para las clases, las propiedades se declaran a través del valor 2008 referido al slot "*:DIRECT_TEMPLATE_SLOTS*", el resto de valores son utilizados por el sistema como los comentarios 9093 o TODO 9092.

- "*facet*": especifica restricciones.

- "*is_template*": campo útil para plantillas OWL.

- "*value_index*": especifica el orden de carga de los elementos contenidos en la declaración de un frame. Es un valor útil para la reconstrucción en memoria del elemento.

- "*value_type*": indica el tipo de valor. En el caso de la primera fila (slot 2002, "*:NAME*") se especifica el valor 3 indicando el tipo "*:STRING*". Otros valores

como enteros 1, reales 2 , booleanos 4 o referencias a otros elementos 9 están especificados en la clase "*edu.stanford.smi.protege.storage.database*"

- "*short_value*": contiene un valor almacenado o referenciado de la declaración del slot, en el caso de la primera fila puede apreciarse el valor del nombre de la clase creada.

Como puede apreciarse en el ejemplo de la figura 5.6 las clases de las que hereda la declaración de la clase LearningObject- AsAnythingDigital se referencian por los identificadores 11563 (clase 11563 "*LearningObject-AsAnything*") y el 11570 (clase "*oc_ComputerFileCopy*") a través de los slots 2004. Aunque el mantenimiento de albergar las referencias a las clases derivadas sea más costoso las búsquedas e inferencias sobre el modelo ganan en rendimiento, por ello a la hora de albergar una nueva clase que herede directamente de "*LearningObject-AsAnythingDigital*" el sistema de persistencia de Protégé guarda los valores de los slots "*:DIRECT_SUBCLASSES*" que en el ejemplo mostrado en la figura 5.6 son los valores 11585 (clase IMSLearningDesign) , 11551 (clase "*LearningObject-LOM*") y 11579 (clase "*SCORM_SCO*") entre otros posibles más. La declaración de las propiedades se realiza utilizando el slot 2008 referenciado los valores 11565 (usedIn) y el 11556 ("*hasAssociatedMetadataRecord*").

Este tipo de arquitectura de almacenamiento persistente está orientada a realizar cualquier tipo de cambios sobre un modelo con el mínimo esfuerzo sin necesidad de prestar mucha atención al rendimiento de las consultas [Sta07], sólo una leve optimización de los slots referentes a las clases directas heredaras y derivadas. Esta característica ha sido tenida en cuenta en el diseño del prototipo.

Resultset 1

frame	frame_type	slot	facet	is_template	value_index	value_type	short_value	long_value
11557	9	2002	0	b'0'	0	3	LearningObject-AsAnythingDigital	NULL
11557	9	2004	0	b'0'	0	9	11563	NULL
11557	9	2004	0	b'0'	1	9	11570	NULL
11557	9	2005	0	b'0'	0	9	11585	NULL
11557	9	2005	0	b'0'	1	9	11551	NULL
11557	9	2005	0	b'0'	2	9	11579	NULL
11557	9	2006	0	b'0'	0	9	9004	NULL
11557	9	2008	0	b'0'	0	13	11565	NULL
11557	9	2008	0	b'0'	1	13	11556	NULL
11557	9	9087	0	b'0'	0	9	11563	NULL
11557	9	9087	0	b'0'	1	9	11570	NULL
11557	9	9123	0	b'0'	0	9	9004	NULL

Figura 5.6: Persistencia del elemento - LearningObject-AsAnythingDigital

5.2.2. Inicialización del repositorio

El componente "*Model Management*" representado en la figura 5.2 requiere de las operaciones de inicialización ("*InitModel*") del componente "*SemanticWebFramework*". En el prototipo desarrollado se ha utilizado la arquitectura de interfaces y clases aportadas por el marco de trabajo para recursos RDF/OWL de Protégé, definido por el componente "*SemanticWebFramework*". Para realizar la operación de inicialización se han de seguir los siguientes pasos:

1. Crear una instancia de la clase "*Project*".

2. Establecer los parámetros de acceso al sistema de persistencia, indicando las fuentes de datos de origen, el driver de acceso a la base de datos, la cadena de conexión, el nombre del modelo y el usuario y la contraseña.

3. Invocar al método "*createDomainKnowledgeBase*".

4. Devolver la referencia.

En el listado de código 5.1 se describe el método "*getProtegeModel()*", encargado de crear e inicializar un modelo OWL sobre el servidor de base de datos utilizado

por SLOR. En la línea 6 se invoca al método "*createNewProject*" con el parámetro de entrada "*factory*" instancia de la clase "*OWLDataBaseKnowledgeBaseFactory*" y el parámetro de salida "*errors*" que referencia a una instancia de lista enlazada utilizada como almacén de los posibles errores que puedan darse en el proceso de inicialización.

En la línea 7 se establecen los parámetros de acceso al sistema de persistencia, se utiliza el parámetro "*DRIVER*" de la clase "*InitSchemaFactory*" para indicar el driver del sistema gestor de base de datos utilizado como almacen persistente de los objetos OWL manejados por SLOR. En el prototipo implementado se ha utilizado MySQL por lo tanto esta variable en ejecución adquiere el valor "*com.mysql.jdbc.Driver*". También se ha de especificar la cadena de conexión, si la comunicación se realiza por el protocolo TCP/IP es necesario indicar la dirección IP y el puerto TCP de acceso. Como ejemplo en el prototipo implementado la variable "*ConnectionString*" adquiere el valor *jdbc:mysql://192.168.0.15:3306/slor* ya que la base de datos "*slor*" utilizada por las clases de persistencia de Protégé está albergada en un servidor MySql con dirección IP 192.168.0.15 y puerto 3306 abierto. Finalmente se han de indicar el usuario y la contraseña del servidor de base de datos.

Una vez configurado los parámetros necesarios, en la línea 8 se ejecuta el método "*createDomainKnowledgeBase*", encargado de desencadenar todas las acciones para la construcción del esquema inicial del repositorio sobre la base de datos (sólo si este no existe). Finalmente en la línea 9 se obtiene la referencia al modelo existente o inicial creado en la base de datos.

Listado 5.1: Código de inicialización del repositorio

```
1   public static OWLModel getProtegeModel() throws
       PersistenceException
2   {
3       try{
4           OWLDatabaseKnowledgeBaseFactory factory = new
               OWLDatabaseKnowledgeBaseFactory();
5           java.util.LinkedList errors=new java.util.LinkedList();
6           Project prj = Project.createNewProject(factory, errors);
7           OWLDatabaseKnowledgeBaseFactory.setSources(prj.getSources()
               , DRIVER, ConnectionString, "model", DBUSER, PWD_DBUSER)
               ;
8           prj.createDomainKnowledgeBase(factory, errors, true);
9           OWLModel owlModel = (OWLModel) prj.getKnowledgeBase();
10          return owlModel;
11      }catch(Exception err){
12          err.printStackTrace();
13          throw(new PersistenceException());
14      }
15  }
```

El marco de trabajo de Protégé aporta un modelo abstracto de clases e interfaces definidas que permite extender nuevos modelos de gestión OWL/RDF con la misma interfaz de operación. En la figura 5.7 se describen dos modelos implementados por el marco de trabajo de Protégé, la clase "*OWLDatabaseModel*" utilizada en el listado anterior de código para la inicialización del repositorio con un modelo persistente sobre un sistema gestor de bases de datos relacionales y la clase "*JenaOWLModel*" implementada para trabajar con modelos RDF/OWL de Jena. Si deseáramos incluir un nuevo modelo de gestión OWL/RDF deberíamos implementar una clase que herede y redefina los métodos descritos en la clase "*AbstractOWLModel*".

5.2.3. Operaciones de administración del repositorio

Aparte de las operaciones de inicialización de un repositorio incluidas en la interfaz "*IRepositoryModel*" del componente "*ModelManagement*", se han incluido dos

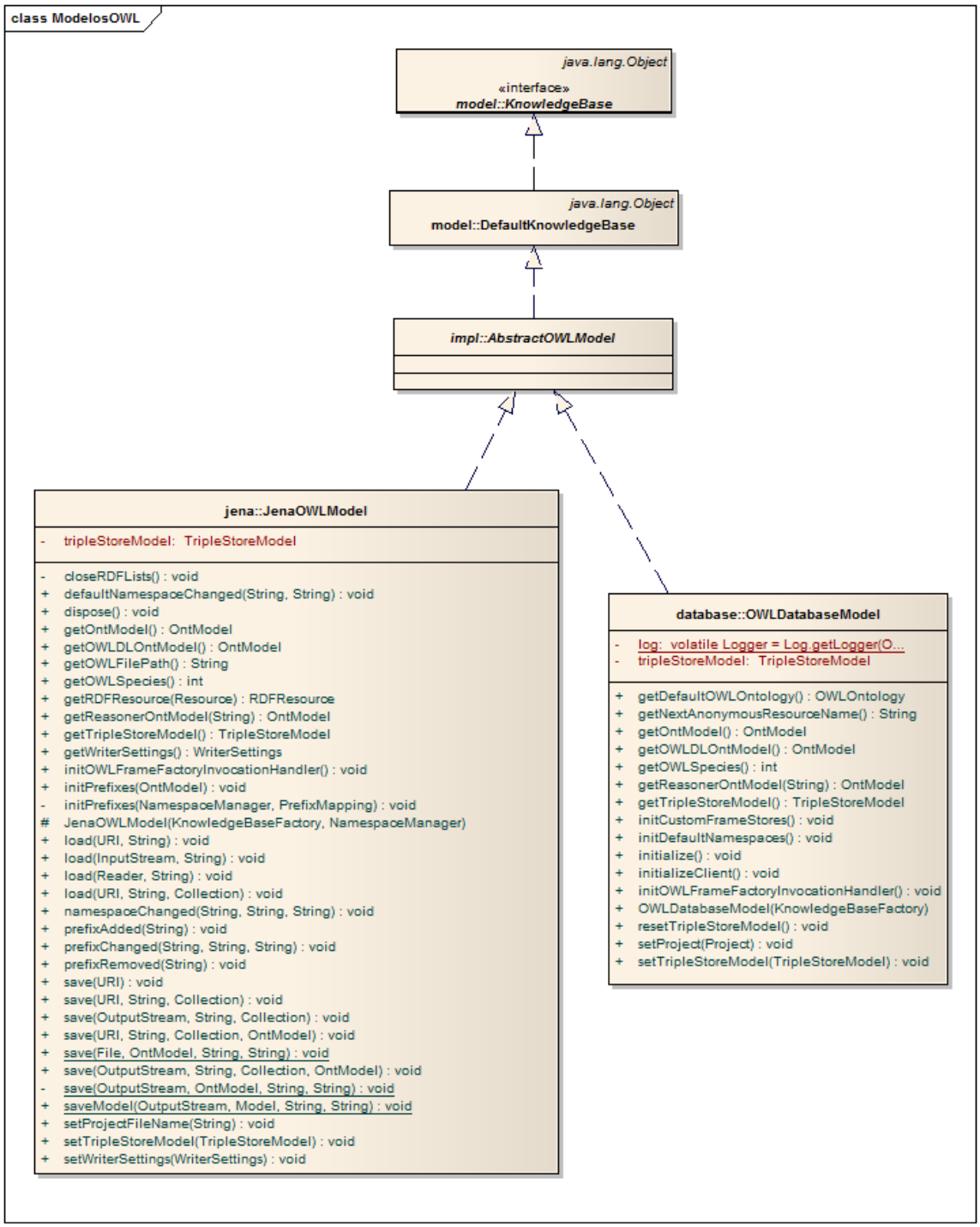

Figura 5.7: Diagrama de clases e interfaces de la arquitectura de gestión de modelos OWL de Protégé

operaciones adicionales para la importación y exportación de los datos albergados en el repositorio en el formato OWL. Estas operaciones son utilizadas a nivel de gestión administrativa (copia de un modelo en otro soporte) o para los procesos de comunicación distribuida (ver sección 5.5.2) que requieran la obtención del código OWL/RDF de un registro de metadatos de un objeto de aprendizaje albergado en el repositorio.

En general todas las operaciones a bajo nivel con las que podemos operar con el repositorio quedan definidas en la interfaz "*OWLModel*" de la biblioteca de clases "*Protégé-OWL*". Por lo tanto se incluye en la interfaz "*IRepositoryModel*" un método para obtener la referencia la interfaz de operaciones a bajo nivel del repositorio "*getOWLModel(String IDModel)*". Entre las operaciones que pueden realizarse cabe destacar las siguientes:

- "*OWLNamedClass createOWLNamedClass(IDCLASS)*": crea una clase OWL con el identificador IDCLASS. A partir de la definición de una clase de una ontología representada por la interfaz "*OWLNamedClass*" se pueden especificar las clases de las que hereda utilizando el método "*addSuperClass(refOntologyClass)*". Los individuales de las clases se generan invocando al método createRDFIndividual(ID) de la interfaz "*OWLNamedClass*".

- "*OWLObjectProperty createOWLObjectProperty(IDOP)*": crea una propiedad "*owl:ObjectProperty*" con el nombre IDOP. El rango de la propiedad se especifica a través del método "*setRange(refOntologyClass)*" y el dominio con el método "*setDomain(refOntologyClass)*" de la interfaz Java "*OWLObjectProperty*".

- "*OWLDatatypeProperty createOWLDatatypeProperty(IDDP)*": crea una propiedad "*owl:DatatypeProperty*" con el nombre IDDP. Al igual que con la clase "*OWLObjectProperty*", la clase "*OWLDatatypeProperty*" posee los métodos "*setRange(refOntologyClass)*" y "*setDomain(refOntologyClass)*" para especificar el rango y el dominio de la propiedad. En el caso de las propiedades Datatype el dominio puede ser declarado utilizado los métodos "*getXSD[type]()*" de la interfaz "*OWLModel*". A modo de ejemplo en un tipo de dato entero sería "*getXSDint()*".

- "*OWLObjectProperty getOWLNamedClass(IDCLASS)*": permite recuperar una clase "*owl:Class*" previamente creada del modelo. El método getSubClasses() permite obtener las clases derivadas y getSuperClasses() permite obtener las clases de las que hereda. Los dos métodos reciben como parámetro una variable lógica que indica si es verdadera (true) una exploración recursiva, en otro caso la exploración se limita a un nivel directo de jerarquía.

- "*OWLObjectProperty getOWLObjectProperty(IDOP)*": permite recuperar una propiedad "*owl:ObjectProperty*" previamente creada del modelo.

- "*OWLDatatypeProperty getOWLDatatypeProperty(IDDP)*": permite recuperar una propiedad "*owl:DatatypeProperty*" previamente creada del modelo.

- "*OWLIndividual getOWLIndividual(ID)*": permite recuperar un individual a con identificador ID.

Si se desea eliminar un objeto del modelo, se debe ejecutar el método "*delete()*" proporcionado por las clases "*OWLNamedClass*", OWLObjectProperty, OWLDatatypeProperty y OWLIndividual.

5.2.4. Componente de gestión del modelo

El componente SLOR Model incluye la funcionalidad de construcción y recuperación de las clases del modelo OWL almacenado en el repositorio. Aporta las clases necesarias para simplificar las operaciones de construcción utilizando el patrón "Factory Method" [GHJV95]. En la figura 5.9 se representa la clase "*OntoSlorFactory*" en la que se implementan los métodos para construir ("*create[Clase](name)*") y recuperar ("*get[Clase](name)*", "*getAll[Clase]Instances(name)*") individuales almacenados dentro del repositorio OWL. Todas las clases pertenecientes a este componente están incluidas dentro del paquete "*slor.ontology*".

En este componente también se incluyen las interfaces correspondientes a las operaciones de gestión de las propiedades definidas en una clase OWL. En la figura 5.8 se representan las interfaces correspondientes a las clases base del modelo descrito en la sección 4.2.2 del capítulo anterior. Estas interfaces han sido obtenidas del proceso de generación de código utilizando la herramienta Protégé. Como puede apreciarse, la correspondencia es directa, cada clase definida en OWL se traduce en una interfaz y dos métodos de generación y recuperación implementados en la clase "*OntoSlorFactory*". En el listado de código 5.2 la clase "*LearningObject-AsAnythingDigital*" descrita en OWL deriva de la clase "*LearningObjectAsAnything*" y "*ocComputerFileCopy*", en el diagrama de clases 5.8 se puede apreciar la herencia múltiple entre interfaces.

Listado 5.2: Código OWL - Definición de la clase "*LearningObject-AsAnythingDigital*"

```
<owl:Class rdf:about="#LearningObject-AsAnythingDigital">
    <rdfs:subClassOf rdf:resource="#LearningObject-AsAnything"/>
    <rdfs:subClassOf rdf:resource="#oc_ComputerFileCopy"/>
</owl:Class>
```

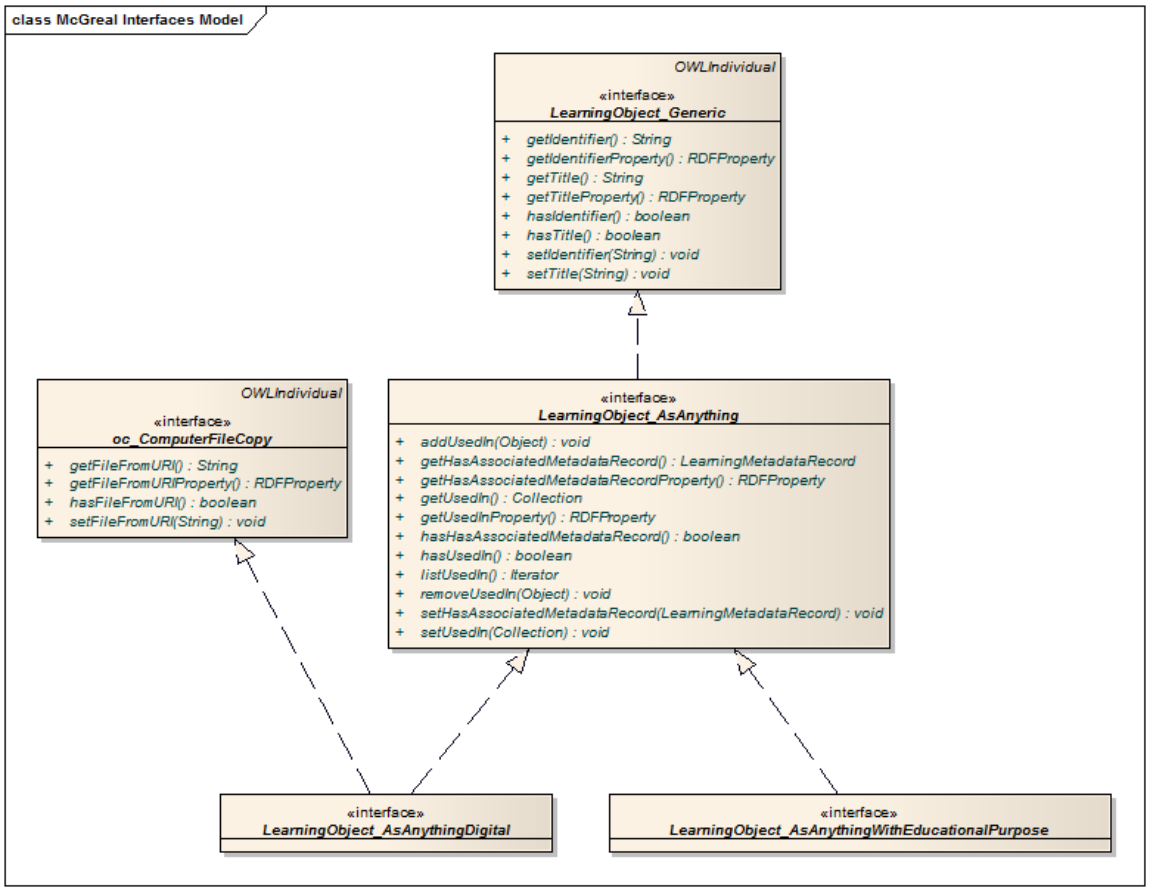

Figura 5.8: Interfaces de las clases base del modelo de SLOR

Utilizando la herramienta "*Protege-OWL Code*", ejecutable desde la interfaz del programa de edición de ontologías, se genera la funcionalidad básica para administrar un repositorio con el modelo descrito en el código OWL. El modelo de interfaces y clases proporcionado por Protégé permite abstraer las funciones de gestión y persistencia de los modelos OWL. Incluye soporte para modelos Jena, pero se puede extender incluyendo soporte para Sesame y otros marcos de trabajo para RDF/OWL.

En el listado de código 5.3 se representa el método "*createLearningObject_LOM*" de la factoria "*SlorOntoFactory*". Este método es el encargado de crear una entrada

dentro del sistema de almacenamiento (memoria, fichero o base de datos) y devolver la referencia para su posterior gestión, en general la inclusión de sus propiedades.

Listado 5.3: Uso de las clases abstractas de Protégé

```
1  public LearningObject_LOM createLearningObject_LOM(String name) {
2        final RDFSNamedClass cls = getLearningObject_LOMClass();
3        return (LearningObject_LOM) cls.createInstance(name).as(
            LearningObject_LOM.class);
4  }
```

El constructor de la factoría "*OntoSlorFactory*" requiere como parámetro para la creación de una instancia el modelo a utilizar. Una vez instanciado el objeto factoría pueden empezarse a crear los objetos RDF/OWL. Si el modelo es persistente (uso de la clase "*OWLDatabaseKnowledgeBaseFactory*") se invocarán las secuencias de comandos SQL necesarias para mantener íntegro el modelo de la base de datos. En el siguiente ejemplo extraído del módulo de gestión LOM se puede apreciar el uso de la factoría:

```
1  OntoSlorFactory sf=new OntoSlorFactory(owlModel);
2  //...
3  slor.ontology.LearningObject_LOM lor= sf.createLearningObject_LOM("
      lom_"+ sb1.getLOID());
4  //...
```

5.2.5. Componente de ejecución de consultas - QueryManager

Todos los marcos de trabajo de la Web semántica aportan un conjunto de funciones relacionadas con el proceso y ejecución de consultas realizadas sobre modelos RDF/OWL. A modo de ejemplo, Jena aporta clases destinadas a realizar consultas

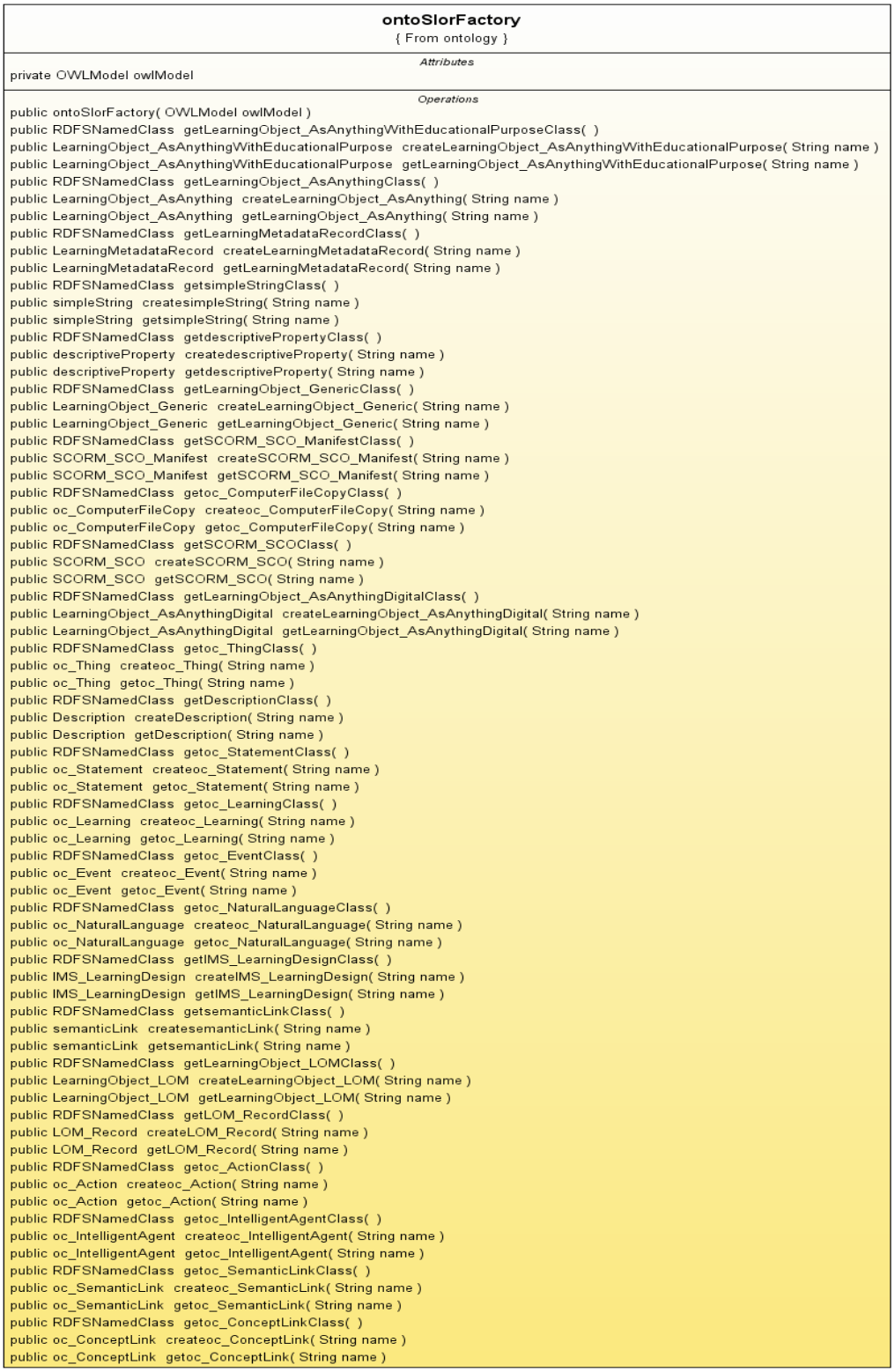

Figura 5.9: OntoSlorFactory - Factoría de clases persistentes

directas por medio de varios métodos de programación, o mediante el uso de complejas sentencias escritas en lenguaje SPARQL [W3C07]. En cambio, Sesame aporta un lenguaje diferente de consulta denominado SeSQL [SeR04] junto con sus propios métodos de búsqueda y obtención de elementos del repositorio. También cabe mencionar las iniciativas OWL-QL [FHH04] y nRQL [HMW] que proporcionan estudios para la definición de lenguajes de consulta específicos sobre bases de conocimiento OWL, que incluyan características de inferencia y deducción.

Actualmente existen diferentes iniciativas que intentan consensuar un lenguaje de consultas específico para un esquema de modelo determinado, los modelos basados en RDF poseen la propuesta de especificación SPARQL implementada en diferentes marcos de trabajo y utilizada actualmente en la mayoría de los sistemas que procesan modelos OWL/RDF. Sobre modelos basados en OWL pueden realizarse consultas con el lenguaje SPARQL pero no se aprovecha toda la capacidad ofrecida por un esquema OWL, por ello los últimos trabajos de investigación [FHH04] van orientados a un nuevo tipo de propuesta de especificación de lenguaje para consultas sobre modelos OWL. Ante este escenario, el diseño de SLOR ha sido preparado para procesar distintos tipos de lenguajes de consulta, para que en función del marco de trabajo utilizado pueda utilizarse el lenguaje de consulta más adecuado. Todas las operaciones de proceso de consultas son proporcionadas por el componente "*QueryManager*".

El objetivo principal del componente "*QueryManager*" es proporcionar una interfaz común a todas las funciones de búsqueda utilizadas por el repositorio, de forma independiente del marco de trabajo persistente RDF/OWL. Por ello, se ha creado la clase "*QueryManager*" que representa el componente de ejecución de consultas y se han incluido las siguientes operaciones:

```
1 public QueryResults executeQuery
2                (String query,SlorParameters.LanguageType lang);
3 public QueryResults executeQueryAndInference
4                (String query,SlorParameters.LanguageType lang,
                    LinkedList<InferenceTerm> inf);
```

Los métodos se apoyan en la interfaz "*QueryResults*" de la biblioteca de clases "*Protégé-OWL*" que encapsula las operaciones para manejar la lista de resultados obtenidos por la ejecución de consultas. Como puede apreciarse, posee una estructura de métodos similar al patrón "iterator" [GHJV95]. En la figura 5.10 se muestra la clase "*SPARQLQueryResults*" que implementa la interfaz "*QueryResults*", los objetos albergados en la lista son del tipo "*Map*" del paquete "*java.util*" por lo que contienen un identificador de objeto y la referencia al objeto RDF/OWL consultado. En el listado de código 5.4 puede apreciarse el manejo de los resultados con la interfaz QueryResults.

El modelo implementado permite fácilmente añadir el soporte para obtener los resultados de otro lenguaje de búsqueda, con sólo crear una clase que implemente los métodos de la interfaz "*QueryResults*". En el caso del lenguaje SeRQL de Sesame podría, por ejemplo, implementarse la clase "*SeRQLQueryResults*".

La implementación del método "*executeQuery*" se apoya del modelo implementado por Protégé, recibe como parámetros la cadena de la consulta a realizar (construida previamente por el módulo de búsqueda) y el tipo de lenguaje de consulta (SPARQL o SeRQL). Las últimas investigaciones realizadas aportan lenguajes más complicados que requieren el uso de un motor de inferencia asociado al modelo OWL para conseguir los resultados solicitados. Propuestas de lenguajes como OWL-QL [FHH04] o nRQL [HMW] permiten conseguir resultados bastante satisfactorios sobre un modelo descrito en el lenguaje OWL.

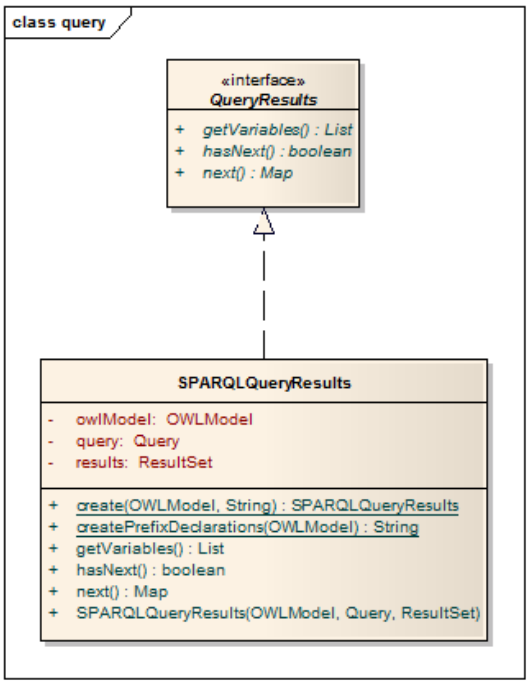

Figura 5.10: Jerarquía de la interfaz QueryResults

En la implementación del prototipo SLOR se ha decidido utilizar un modelo simple de ejecución de consultas basadas en el lenguaje SPARQL y un pequeño apoyo de inferencia sobre un módulo encargado de realizar deducciones sobre la jerarquía de clases de los modelos utilizados (SLOR, OpenCyc y otras ontologías y tesauros utilizados en la descripción de recursos) "*OpenCyc-OWL-Reactor*". También se han tenido en cuenta los estudios actuales en los nuevos lenguajes de consulta sobre OWL a la hora de diseñar el sistema de ejecución, dejando un modelo extendible para su posterior ampliación.

A continuación se muestra en el listado de código 5.4 la función "*executeQuery*" de la interfaz "*QueryManager*" implementado en la clase "*SLORQueryManager*". En la línea 5 se comprueba si el marco de trabajo semántico soporta el lenguaje de la

consulta pasado en la variable "*lang*", para ello se ejecuta el método de verificación "*isSupportedQueryLanguage*" de la clase que contiene los parametros de configuración generales del repositorio "*SlorParameters*". Si el lenguaje no es soportado se lanza la excepción "*UnSupportedQueryLanguage*". Si el sistema de almacenamiento del modelo del repositorio soporta el lenguaje, a continuación (línea 6) se comprueba el tipo de lenguaje para ejecutar la secuencia de operaciones que permitan realizar la consulta. En el ejemplo expuesto en el código sólo queda implementada la parte de SPARQL, si se desea añadir otro lenguaje de consulta habría que escribir el código correspondiente, como en el caso de la comprobación realizada en la línea 13. En la línea 8 se ejecuta la consulta invocando al método del modelo "*executeSPARQLQuery(query)*". Si se produjese algún error por sintaxis o algún fallo externo al sistema como la desconexión del almacenamiento persistente, en la línea 11 el método lanza la excepción "*QueryException*".

Listado 5.4: Método executeQuery

```
1  public QueryResults executeQuery
2                  (String query,SlorParameters.LanguageType lang)
3  throws QueryException,UnSupportedQueryLanguage
4  {
5      if (SlorParameters.isSupportedQueryLanguage(lang)){
6          if (lang==SlorParameters.LanguageType.SPARQL){
7              try{
8                  QueryResults r=model.executeSPARQLQuery(query);
9                  return r;
10             }catch(Exception err){
11                 throw(new QueryException())
12             }
13         }else if (lang==SlorParameters.LanguageType.SeRQL){
14             // Código de ejecución para un modelo Sesame.
15             // el modelo ha de implementar el método
16             // executeQuery(String query)
17         }//else if.. otro lenguaje..
18     }else{
```

```
        throw(new UnSupportedQueryLanguage())
    }
}
```

A efectos prácticos, el módulo de búsqueda (o el propio núcleo) utilizarán los métodos proporcionados por esta interfaz para construir las consultas solicitadas por los usuarios del sistema. A modo de ejemplo, si se desea obtener los objetos de aprendizaje de cursos relacionados con el análisis de ADN de Ploidy cuyo título contenga la palabra "ADN", se pueden especificar como consulta las siguientes restricciones:

- Realizar la búsqueda únicamente sobre registros de tipo "*LOM_Record*".
- Campo "*Title*" debe empezar con la cadena "ADN".
- Palabra clave (propiedad "*keywords*") debe ir asociada al concepto específico "*DNA_Ploidy_Analysis*" de la ontología OWL del NCI (National Cancer Institute)[2].

Sobre el modelo del repositorio SLOR, la consulta que posee las características descritas anteriormente tiene la siguiente forma en el lenguaje SPARQL:

```
PREFIX nci: <http://www.mindswap.org/2003/CancerOntology/nciOncology.
    owl>
SELECT ?object ?title
    WHERE {    ?object :Title ?title .
               FILTER(regex(str(?title),'ADN')) .
               ?object :hasAssociatedMetadataRecord ?mt .
               ?mt  rdf:type  :LOM_Record .
               ?mt :keywords ?sl .
               ?sl  rdf:type  :owl_ConceptLink .
               ?sl  :ref_Term nci:DNA_Ploidy_Analysis . }
```

[2]http://www.cancer.gov/

Este tipo de consulta se puede construir a través de la interfaz de usuario que utiliza el módulo de búsqueda con objeto de concatenar de forma correcta todas las sentencias que van en la sección WHERE de la consulta. En el código SPARQL se puede apreciar (en la línea 5) cómo se obtienen en la variable "*?mt*" las referencias a los registros de metadatos asociados al objeto de aprendizaje. También cabe resaltar cómo se especifica la condición del tipo de registro de metadatos asociado a analizar "*LOM_Record*" y la búsqueda de una palabra clave asociada al término "*ADN*" de la ontología "*NCI*".

Los motores de ejecución de consultas basados en modelos RDF y no OWL carecen de los mecanismos necesarios para aprovechar la expresividad ofrecida por el lenguaje OWL. Sin embargo, algunas técnicas de inferencia y deducción podrían aplicarse para obtener resultados relacionados. En el ejemplo anterior, si se deseara obtener también los resultados de todos los objetos de aprendizaje relacionados con técnicas de análisis de ADN utilizadas en procesos de detección de cáncer, se podrían inferir resultados utilizando el término base directo "*DNA_Analysis* " del concepto "*DNA_Ploidy_Analysis*".

También utilizando este análisis se podrían incluir búsquedas sobre los objetos de aprendizaje cuyas palabras clave sean términos derivados de los conceptos que estamos buscando. Así si en vez de buscar por un análisis específico de ADN utilizado en la investigación contra el cáncer como el de Ploidy, deseamos obtener todos los objetos de aprendizaje relacionados con cualquier tipo de análisis de ADN, se puede ejecutar el proceso de inferencia que permite obtener todos los términos derivados de uno dado para conseguir los resultados deseados. En el caso del término "*DNA_Analysis*" de la ontología NCI se realizaría la consulta para todos los términos derivados "*Allelotype Analysis*", "*Blotting*", "*DNA Fingerprinting*", "*DNA Ploidy Analysis*", "*Gene*

Deletion Detection",..., "*Tumor Replication Error Analysis*".

Nuevas especificaciones de lenguajes de consultas sobre OWL como son OWL-QL [FHH04] o nRQL [HMW] explotan las propiedades de los modelos OWL para crear especificaciones que puedan se utilizadas sobre nuevos tipos de motores de ejecución e inferencia (como el razonador RACER).

En el desarrollo del prototipo SLOR se ha implementado el componente de inferencia "*OpenCyc-OWL-Reactor*" con el objetivo de procesar operaciones de deducción (únicamente respecto a la jerarquía de clases) sobre el modelo OWL utilizando conceptos de múltiples ontologías escritas en diferentes lenguajes. El componente es necesario debido a la carencia descrita en el capítulo anterior en la sección 4.4.1, respecto a procesar la versión OpenCyc en OWL. En la implementación del prototipo se ha escrito una parte dedicada al proceso de las inferencias sobre la base de conocimiento OpenCyc y otra específica para OWL.

El segundo método de búsqueda de objetos de aprendizaje enunciado al principio de esta sección, retorna una lista de resultados que es la unión de la ejecución de todas las consultas generadas por el componente "*OpenCyc-OWL-Reactor*". Como se ha comentado, se permiten exploraciones a través de la jerarquía de clases de las ontologías utilizadas, la clase "InferenceTerm" referencia un término o concepto de una ontología que ha de ser procesado por el componente "*OpenCyc-OWL-Reactor*" según el comportamiento indicado en la propiedad "*inferenceType*". A continuación se explica la secuencia de acciones seguida por el proceso "*execute*" de la clase "*QueryBuilder*" del componente "*OpenCyc-OWL-Reactor*" del núcleo de SLOR para el proceso de los términos enviados:

1. Por cada término enviado en la lista "*inf*" se carga la ontología asociada especificada en la propiedad "*ontologyURI*" (este proceso puede optimizarse utilizando una caché de grafos RDF/OWL). En el caso de ser un concepto de la ontología OpenCyc se establece una conexión a través de la interfaz "*IocBridge*" del sistema de persistencia.

2. Una vez cargada la ontología, se ejecuta el tipo de inferencia sobre el término utilizado. Si es de tipo "*TOP-DOWN*", se obtienen todos los términos derivados del analizado y si es del tipo "*BOTTOM-UP*" se infieren los términos correspondientes a las jerarquías de nivel superior.

3. Tras el proceso de inferencia, con los resultados obtenidos se construye la sentencia o conjunto de sentencias en el lenguaje especificado que permitan obtener los resultados indicados en la consulta "*query*" pasada como parámetro en el constructor.

A continuación se muestra una sentencia SPARQL obtenida tras la ejecución del método "*execute*" de la clase "*QueryBuilder*", después de definir una instancia con los siguientes parámetros en el constructor de la clase: la sentencia SPARQL similar a la mostrada en el primer ejemplo de esta sección, el tipo de lenguaje (SPARQL) y la lista de términos sobre los que se desea realizar la inferencia. En este ejemplo sólo se utiliza el término "*nci:DNA_Analysis*" en el que se especifica la URI de la ontología (propiedad ontologyURI)[3], el tipo de inferencia (propiedad inferenceType) "*TOP-DOWN*" y el identificador del término (propiedad "*term*") "*DNA_Analysis*".

```
1 PREFIX nci: <http://www.mindswap.org/2003/CancerOntology/nciOncology.
    owl>
2 SELECT ?object ?title
3     WHERE {    ?object :Title ?title .
```

[3]http://www.mindswap.org/2003/CancerOntology/nciOncology.owl

```
4               FILTER(regex(str(?title),'ADN')) .
5               ?object :hasAssociatedMetadataRecord ?mt .
6               ?mt  rdf:type  :LOM_Record .
7               ?mt :keywords ?sl .
8               ?sl rdf:type :owl_ConceptLink .
9               {?sl  :ref_Term nci:DNA_Analysis}
10          UNION{?sl :ref_Term nci:DNA_Allelotype_Analysis}
11          UNION{?sl :ref_Term nci:DNA_Ploidy_Analysis}
12          UNION{?sl :ref_Term nci:DNA_Gene_Amplification}
13          UNION{?sl :ref_Term nci:DNA_Tumor_Replication}.
14    }
```

A partir de la línea 8 en el listado de código anterior puede apreciarse la generación de la expresión que permite consultar no sólo los objetos de aprendizaje que tienen definidos como palabras clave la clase DNA_Analysis, sino que además se incluye en la consulta la unión de todos los términos derivados. Si por cualquier restricción del lenguaje SPARQL no se pudiesen obtener todos los resultados en una única consulta, el método devuelve varias consultas SPARQL para su posterior ejecución en el sistema.

Para entender mejor el contexto en el cual se utiliza la clase "*QueryBuilder*" del componente "*OpenCyc-OWL-Reactor*", en el listado de código 5.5 se muestra la secuencia de instrucciones definidas para el método "*executeQueryAndInference*" de la interfaz "*QueryManager*" del sistema de persistencia de SLOR. A diferencia del primer método (más sencillo) explicado en esta sección, se ha definido una lista enlazada de referencias a resultados de varias consultas. Esto es debido a la característica comentada anteriormente sobre la construcción de sentencias de consulta basadas en operaciones de inferencia, ya que en algunos casos puede no ser válida una única consulta para recuperar todos los datos solicitados. La primera comprobación realizada en el método (línea 6) pregunta por el lenguaje de consulta utilizado.

A continuación (línea 9) se utiliza la clase "*slor.owlreactor.QueryBuilder*" para construir las sentencias SPARQL que permiten obtener los resultados deseados. Para el caso del ejemplo anterior, se construye una única sentencia con los términos proporcionados del análisis realizado por la clase "*ImplOWLReactor*" del paquete "*slor.owlreactor*" utilizada dentro del método "*execute*" invocado en la línea 10.

Listado 5.5: Ejecución de consultas e inferencia

```
1  public LinkedList<QueryResults> executeQueryAndInference
2                 (String query,SlorParameters.LanguageType lang,
                     LinkedList<InferenceTerm> inf){
3  throws QueryBuilderException,UnSupportedQueryLanguage
4  {
5      LinkedList<QueryResults> ret=new LinkedList<QueryResults>();
6      if (SlorParameters.isSupportedQueryLanguage(lang)){
7          if (lang==SlorParameters.LanguageType.SPARQL){
8              try{
9                  QueryBuilder qb=new slor.owlreactor.QueryBuilder(query,
                       lang,inf);
10                 LinkedList<String> queryList=qb.execute();
11                 for (String query:queryList){
12                     QueryResults r=model.executeSPARQLQuery(query);
13                     ret.add(r);
14                 }
15                 return ret;
16             }catch(QueryBuilderException err){
17                 // Process QueryBuilderException
18                 throw(err);
19             }catch(Exception err){
20                 throw(new QueryException());
21             }
22         }else if (lang==SlorParameters.LanguageType.SeRQL){
23             // Código de ejecución para un modelo Sesame.
24             // el modelo ha de implementar el método
25             // executeQuery(String query)
26         }//else if.. otro lenguaje..
27     }else{
28         throw(new UnSupportedQueryLanguage())
```

```
29      }
30  }
```

La funcionalidad proporcionada por la interfaz "*QueryManager*" es utilizada en múltiples componentes de SLOR. Para poderla comprender mejor es necesario evaluar el contexto global del modelo, ya que la propuesta incluye el trabajo con múltiples ontologías con el objeto de proporcionar un alto nivel de riqueza de expresividad semántica capaz de ser procesada por una máquina a través de este tipo de consultas. En la siguiente sección se describe el componente clave que permite enlazar el modelo de SLOR con OpenCyc. Los beneficios son notables y han sido expresados en la sección 4.4.1 del capítulo anterior.

5.2.6. Comunicación con OpenCyc

En el prototipo de SLOR se ha implementado el componente "*OpenCycBrige*" que permite realizar consultas a la base de conocimiento OpenCyc para aportar mayor riqueza semántica al modelo de descripción de recursos. El componente se localiza dentro del paquete "*slor.persistence.oc*" del prototipo. Para habilitar la comunicación con la base de conocimiento OpenCyc se han de seguir los siguientes pasos:

1. Descargar la base de conocimiento desde el sitio web www.opencyc.org. Existe versión tanto para el sistema operativo Windows como para el sistema operativo Linux.

2. Iniciar el servidor desde la línea de comandos. Si se desea puede configurarse el inicio del servidor dentro de uno de los pasos del arranque del sistema operativo.

Al iniciar el servidor OpenCyc se dejan abiertos el puerto 3600, para la comunicación por TCP/IP de los comandos principales de la biblioteca de programación y el puerto 3602, como servidor web. Desde el puerto 3602 se puede acceder a una interfaz web

que permite gestionar las múltiples reglas y conceptos albergados en la base de conocimiento.

A través de la API de Java (proporcionada en el mismo archivo comprimido de la distribución) podemos comunicarnos con el servidor OpenCyc desde una aplicación Java. Para abrir una conexión es necesario instanciar la clase "*CycAccess*", el constructor de la clase recibe como parámetros la dirección IP del servidor (parámetro IPocSERVER) y el puerto de conexión (por defecto 3600). Se puede utilizar el método "*traceOn*" para mostrar los mensajes de operación en la consola del servidor. En el listado de código 5.6 se muestra la clase "*OcConnection*" utilizada por el sistema de persistencia para establecer las conexiones con el servidor de OpenCyc. Los parámetros más relevantes, como la dirección IP del servidor de base de datos o la dirección IP del servidor OpenCyc, son almacenados en las variables globales definidas en la clase "*slor.kernel.SlorParameters*".

Listado 5.6: Conexión OpenCyc

```
1  package persistence.oc;
2  import org.opencyc.api.CycAccess;
3  import slor.kernel.SlorParameters;
4
5  public class OcConnection {
6       CycAccess _cycaccess;
7
8       public OcConnection() throws java.io.IOException
9       {
10         try
11         {
12           _cycaccess = new CycAccess(SlorParameters.IPocSERVER,3600);
13           _cycaccess.traceOn();
14         }
15         catch (java.io.IOException err)
16         {
```

```
17            throw (err);
18          }
19       }
20
21       public CycAccess getOcConnection(){
22          return _cycaccess;
23       }
24       //...
25 }
```

Una vez establecida la conexión se pueden enviar comandos, tales como consultas "*cyc-query*" o inferencias para obtener cualquier tipo de información de la base de conocimiento. Es un proceso simple que requiere normalmente de 4 pasos:

1. Declarar todas las variables utilizando la clase "*CycVariable*".
2. Construir la consulta utilizando un objeto instanciado de la clase "*CycAccess*".
3. Enlazar las variables declaradas a la consulta a través de la factoría "*CycObject*".
4. Finalmente, enviar la consulta por medio del objeto "*CycAccess*" y obtener los resultados en una lista de tipo "*CycList*".

A continuación, en el listado de código 5.7 se muestra un ejemplo de las instrucciones necesarias para la ejecución de una consulta (como "*#$isa ?X #$LivingLanguage*") implementada en el método "*queryVariable*" de la clase "*OpencycController*" perteneciente al componente "*OpencycBridge*". La finalidad del método es la de obtener una lista de todos los conceptos que cumplen la expresión y devolverlos en una lista (en Java utilizamos la clase "*ArrayList*"). Como se ha comentado anteriormente, es necesario declarar todas las variables utilizadas en la consulta. En la línea 7 se declara la referencia "*languageVariable*", después se declara la consulta como una estructura de lista CycL invocando al método "*makeCycList*" de la instancia de la clase "*CycAccess*" (línea 11), para seguidamente (línea 12) declarar las variables de

la consulta especificada. Tras ello se utiliza el método "*queryVariable*" de la clase "*CycAccess*", necesario para ejecutar la consulta y obtener los resultados en una lista de conceptos CycL "*CycList*". En este método es necesario especificar la microteoría sobre la cual se realiza la consulta. Si se desea tener en cuenta posibles aserciones en toda la base de conocimiento OpenCyc es necesario especificar "*inferencePSC*". Para utilizar los resultados en la interfaz y abstraer de la operaciones de la API de CycL la lista de resultados, "*response*" se traduce a un "*ArrayList*" de Java.

Listado 5.7: Clase OpenCyc Controller

```
1  public class OpencycController
2  {
3        CycAccess _cycaccess;
4        //...
5
6     public ArrayList<String> queryVariable(String q) throws java.io.
          IOException{
7        CycVariable languageVariable = null;
8        CycList response = null;
9        ArrayList<String> results=null;
10
11       CycList query = _cycaccess.makeCycList(q);
12       languageVariable=CycObjectFactory.makeCycVariable("?X");
13       try{
14           CycConstant mt=this._cycaccess.getConstantByName("
                InferencePSC");
15           response = _cycaccess.queryVariable(
16                            languageVariable,
17                            query,
18                            _cycaccess.inferencePSC,
19                            (HashMap) null);
20           results=new java.util.ArrayList<String>();
21           Iterator iterator=response.iterator();
22           while (iterator.hasNext())
23           {
24               CycConstant item=(CycConstant)iterator.next();
25               results.add(item.getName());
```

```
26              }
27              return results;
28          }catch(java.io.IOException err){
29              throw(err);
30          }
31      }
32  // ...
33  }
```

A continuación se analiza con detalle el componente "*HCIQueryComponent*" que hace uso de las clases e interfaces proporcionadas por el componente "*OpenCycBridge*".

HCIQueryComponent

El componente HCIQueryComponent incluye las consultas más comunes utilizadas para obtener los valores usados en las Interfaces Web o de escritorio de los módulos. Por ejemplo, el relleno de los campos de la lista desplegable "*KeyWords*" usando la tecnología AJAX. Cuando el usuario escribe varios caracteres, la interfaz automáticamente invoca al módulo correspondiente de gestión de registros de metadatos la solicitud de ese tipo de información, este a su vez invoca al núcleo los datos solicitados utilizando el patrón de búsqueda. Finalmente el núcleo utiliza la clase "*HCIQueryComponent*" para obtener el conjunto de resultados esperados y devolverlos al módulo que ha iniciado la consulta. En la sección 5.5 se describe en profundidad este tipo de interacción con las Interfaces Web de construcción de registros de metadatos.

La interfaz proporciona operaciones de consulta sobre la base de conocimiento OpenCyc que permiten facilitar los procesos de construcción de las expresiones semánticas incluidas en los registros de metadatos del repositorio, de forma similar a

la tecnología Intellisense[4]. En la interfaz se aportan los siguientes operaciones:

- "*public String[] getOpencycConcepts(String prefix,int maxResults, boolean comments)*": obtiene una lista de conceptos de la base de conocimiento ejecutando la expresión regular "*prefix**". Si se especifica la variable booleana "*comments*" el método devuelve en cada cadena asociado el comentario que explica el significado en el forma "`concepto||comentario`".

- "*public boolean isConcept(String concept)*": método utilizado para consultar si una cadena es un concepto OpenCyc o no.

- "*public boolean isPredicate(String predicate)*": método utilizado para saber si una cadena es un predicado OpenCyc o no.

- "*public int getArity(String predicate)*": obtiene el número de parámetros necesarios para construir un predicado.

- "*public List queryPredicate(String prefix, ArrayList cache)*": obtiene la lista de predicados que cumplen una expresión regular "*prefix**".

- "*public ArrayList <String> getInstancesOf(String predicate , int argIndex)*": obtiene las instancias correspondientes al concepto definido en el argumento "*argIndex*" del predicado.

Para obtener una lista de conceptos de la base de conocimiento OpenCyc es necesario ejecutar la consulta CycL "*(constant-complete "var_prefix":true)*" donde la variable "*var_prefix*" contiene el prefijo o el nombre completo del concepto que deseamos buscar; también se especifica como segundo parámetro un valor constante "*:true*" de tipo "*BOOLEANP*" que indica una búsqueda "*case-sensitive*". En la línea 5 del listado de código 5.8 se ejecuta la consulta invocando al método

[4]http://msdn2.microsoft.com/en-us/library/hcw1s69b(vs.71).aspx

"*converseList*" de la clase "*CycAccess*" de la biblioteca de programación Java OpenCyc.

Los resultados se obtienen en una lista de diferentes tipos de conceptos. Una vez obtenidos los resultados en la variable "*results*" se realiza la comprobación del tamaño total de la lista[5]. Si este es menor que el máximo número de resultados admitidos en la lista representado por el parámetro "*maxResults*", se instancia un array de cadenas del mismo tamaño, de lo contrario se instancia el array con el máximo tamaño admisible. Posteriormente se rellena el array con el nombre de los conceptos incluidos en la lista "*results*", y si se activa el flag "*comments*" se incluyen también los comentarios separados por la secuencia de caracteres ||.

Listado 5.8: Método getOpencycConcepts

```
1  public String[] getOpencycConcepts(String prefix,int maxResults,
       boolean comments){
2          CycList results=null;
3          String[] x=null;
4          try {
5              results = _cycaccess.converseList("(constant-complete \"" +
                   prefix + "\" :true) ");
6              Iterator it=results.iterator();
7              if(maxResults==-1 || results.size()<maxResults)
8                  x=new String[results.size()];
9              else x=new String[maxResults];
10             int i=0;
11             while (it.hasNext()){
12                 if (i>=maxResults) break;
13                 Object obj=(Object) it.next();
14                 if (!comments)
15                     x[i++]=obj.toString();
16                 else{
```

[5]Al obtener un conjunto completo de resultados se entorpece el rendimiento de la función constant-complete, esta función no puede configurarse con un valor TOP similar al utilizado en el lenguaje SQL sobre los resultados de una orden SELECT.

```
17                      CycConstant co=(CycConstant)obj;
18                      x[i++]=co.toString() + "||" + _cycaccess.getComment(
                            co);
19                  }
20              }
21          } catch (CycApiException ex) {
22              ex.printStackTrace();
23          } catch (IOException ex) {
24              ex.printStackTrace();
25          }
26          return x;
27  }
```

A continuación en los listados de código 5.9, 5.10 y 5.11 se describen métodos de apoyo utilizados por las funciones internas de este componente y por el resto de componentes que utilicen la interfaz expuesta.

La primera función es "*isConcept*", utiliza la función "*getConstantByName(concept)*" de la clase "*CycAccess*" que implementa el código necesario utilizado para comprobar si un concepto existe en la base de conocimiento OpenCyc. La ejecución de esta función se realiza en la línea 3 en el listado de código 5.9, si lo encuentra devuelve una referencia al concepto, sino devuelve el valor "*null*" que indica la no existencia del concepto.

Listado 5.9: Método isConcept

```
1  public boolean isConcept(String concept){
2          try {
3              CycConstant cc=_cycaccess.getConstantByName(concept);
4              if(cc!=null) return true; else return false;
5          } catch (CycApiException ex) {
6              ex.printStackTrace();
7          } catch (IOException ex) {
8              ex.printStackTrace();
9          }
```

```
10          return false;
11  }
```

Otra función utilizada frecuentemente por el núcleo y los módulos del repositorio SLOR es "*isPredicate*", su objetivo es el de comprobar si una cadena está declarada como predicado dentro de la base de conocimiento OpenCyc. El proceso sigue la secuencia de instrucciones mostradas en el listado de código 5.10, principalmente las instrucciones relevantes son:

- Línea 4, obtener una referencia a la constante que deseamos comprobar.
- Líneas 7-8, verificar si el valor de la constante deriva del término "*Predicate*".

Listado 5.10: Método getOpencycConcepts

```
1  public boolean isPredicate(String predicate){
2          CycConstant x;
3          try {
4              x = _cycaccess.getConstantByName(predicate);
5              if (x!=null)
6              {
7                  CycConstant pred=_cycaccess.getConstantByName("
                       Predicate");
8                  return _cycaccess.getAllIsa(x).contains(pred);
9              }else return false;
10         } catch (CycApiException ex) {
11             ex.printStackTrace();
12         } catch (IOException ex) {
13             ex.printStackTrace();
14         }
15         return false;
16 }
```

La función "*getArity*" devuelve el número de argumentos del predicado OpenCyc pasado como parámetro, para ello se ejecuta el método que posee el mismo nombre

de la clase "*CycAccess*" pero requiere un parámetro del tipo CycConstant.

Listado 5.11: Método getArity

```
1  public int getArity(String predicate){
2      int ret=-1;
3      try {
4          ret=_cycaccess.getArity(_cycaccess.getConstantByName(predicate)
               );
5      } catch (CycApiException ex) {
6          ex.printStackTrace();
7      } catch (IOException ex) {
8          ex.printStackTrace();
9      }
10     return ret;
11 }
```

Uno de los métodos más utilizados para la construcción de expresiones semánticas insertadas en el modelo de SLOR es "*queryPredicate*". Este método realiza una consulta sobre todos los predicados existentes en OpenCyc, recibe como parámetros la variable "*prefix*" y la lista "*cache*". El parámetro caché es utilizado para la optimización del método: en la línea 4 se realiza la consulta sobre OpenCyc obteniendo todos los predicados almacenados en OpenCyc, para no tener que volver a ejecutarla se almacenan los resultados en la lista "*cache*". Sobre el conjunto de predicados almacenados como cadenas, en la línea 8 se define una consulta joSQL[6], similar al lenguaje LINQ [BH07], para obtener el conjunto de predicados que concuerdan con la expresión regular "*prefix**". En la línea 15 se ejecuta la consulta declarada invocando al método "*execute*" de la clase "*josql.Query*", los resultados son recogidos en una lista y posteriormente se ordenan (línea 16) invocando al método estático "*sort*" de la clase "*Collections*". Finalmente, en la línea 21 se devuelven los resultados.

[6]SQL for Java Objects - http://josql.sourceforge.net/

Listado 5.12: Método queryPredicate

```
1  public List queryPredicate(String prefix,ArrayList cache) throws java.
       io.IOException{
2          try{
3               if (cache==null)
4                   cache=ocController.queryVariable("(#$isa ?X #$Predicate
                        )");
5
6               Query q=new Query();
7               try {
8                   q.parse("SELECT * FROM java.lang.String WHERE toString
                        $LIKE '"+ prefix + "%'");
9               } catch (QueryParseException ex) {
10                  ex.printStackTrace();
11              }
12
13              List<String> results=null;
14              try {
15                  results = (List<String>) q.execute(cache).getResults();
16                  Collections.sort(results);
17              } catch (QueryExecutionException ex) {
18                  ex.printStackTrace();
19              }
20
21              return results;
22
23         }catch(java.io.IOException err){
24             throw(err);
25         }
26     }
```

Otro de los métodos más utilizados en el proceso Intellisense con OpenCyc es (“*getInstancesOf*”), proporcionado por el componente HCIQueryComponent. Este método devuelve todas las instancias de la clase definida para un argumento (especificado en el parámetro “*argIndex*”) del predicado pasado como el parámetro “*predicate*”. Su funcionamiento es el siguiente: primero, en la línea 5 del listado de código 5.13 se obtiene la referencia al predicado de la base de conocimiento

OpenCyc, para seguidamente (línea 6) obtener la lista de referencias a los conceptos base definidos para el parámetro ("*argIndex*"). Por cada uno de estos términos se itera (línea 9) y se obtienen todas las instancias relacionadas invocando el método "*getAllInstances*" de la clase "*CycAccess*". Cada una de estas instancias es almacenada dentro de la lista devuelta ("*results*", línea 16).

Listado 5.13: Método getInstancesOf

```
1  public ArrayList<String> getInstancesOf(String predicate,int argIndex)
       {
2          CycFort _predicate;
3          ArrayList<String> results=null;
4          try {
5              _predicate = _cycaccess.getConstantByName(predicate);
6              CycList cl=_cycaccess.getArgNIsas(_predicate,argIndex);
7              Iterator i=cl.iterator();
8              results=new ArrayList<String>();
9              while(i.hasNext()){
10                 CycConstant typeArg=(CycConstant)i.next();
11                 CycList cli=_cycaccess.getAllInstances(typeArg);
12                 Iterator j=cli.iterator();
13                 while(j.hasNext()){
14                     try{
15                         CycConstant indv=(CycConstant)j.next();
16                         results.add(indv.getName());
17                     }catch(java.lang.ClassCastException ce){
18                         continue;
19                     }
20                 }
21             }
22         } catch (CycApiException ex) {
23             ex.printStackTrace();
24             return null;
25         } catch (IOException ex) {
26             ex.printStackTrace();
27             return null;
28         }
29         return results;
30 }
```

5.3. Núcleo de SLOR

El núcleo de SLOR engloba toda la funcionalidad común utilizada por los módulos de SLOR. La figura 5.11 muestra el diagrama de componentes y las dependencias de ensamblado existentes entre las interfaces. A continuación se describe a modo de resumen la funcionalidad aportada por cada uno de ellos:

- Core: es el componente más importante, aporta la funcionalidad básica para la administración de los objetos de aprendizaje almacenados en el subsistema de persistencia. También proporciona las operaciones de enlace de conceptos entre distintas ontologías utilizando los componentes "*OWLSemanticLink*" y "*OCSemanticLinkBuilder*".

- Componentes de "enlace" a conceptos de ontologías externas: en el núcleo de SLOR existen dos componentes encargados de aportar esta funcionalidad, el componente "*OWLSemanticLink*" encargado de crear enlaces con conceptos de otras ontologías escritas en lenguaje OWL, y el componente constructor "*OCSemanticLinkBuilder*", creado con el objetivo de generar enlaces a conceptos existentes en la base de conocimiento OpenCyc.

- OpenCyc-OWLReactor: proporciona la integridad necesaria del modelo para realizar operaciones de inferencia. Es necesario debido al uso de diferentes lenguajes de ontologías (OWL / CycL).

- Componentes de gestión de usuarios: los componentes "*Users*", "*Privileges*", "*Login*" y "*FOAFInterpreter*" forman parte del grupo de componentes de gestión de usuarios. El componente "*Users*" se encarga de la administración de usuarios y la asignación de privilegios en función del tipo de usuario. Los usuarios pueden especificar un archivo de descripción más completo utilizando

una URL de referencia al archivo OWL creado bajo la ontología FOAF[7].

- Componentes funcionales: como el sistema interno de mensajería utilizado para la comunicación y aviso de los usuarios "*Internal Message System*" y el componente que encapsula la funcionalidad del sistema de registro de las acciones realizadas "*Log Subsystem*".

El núcleo de SLOR se apoya en otras bibliotecas externas que aportan componentes de comunicación "*Communication Framework*" o acceso directo al sistema de base de datos "*JDBC*".

En el diagrama de componentes representado en la figura 5.11 se puede apreciar la fuerte cohesión existente entre los componentes del núcleo y el subsistema "*SLOROntoManager*". En la figura sólo se han representado las interfaces más importantes.

5.3.1. Enlaces a conceptos de ontologías externas (OpenCyc/OWL

Como ya se ha descrito a lo largo del capítulo, la descripción flexible de recursos es uno de los objetivos principales a conseguir en esta investigación. Para habilitar la búsqueda semántica con inferencia en los registros de metadatos almacenados, es necesario incluir dentro del repositorio los mecanismos de enlace de conceptos entre ontologías.

Para ello, se ha diseñado un meta-esquema que contiene las distintas clases que permiten enlazar conceptos entre ontologías. La figura 5.3.1 representa la clase OpenCyc "*oc_Statement*" que evoca el concepto de descriptor genérico, así un registro de metadatos LOM o IMS-CP serán individualizaciones específicas, o incluso una descripción

[7]Ontología FOAF - http://xmlns.com/foaf/0.1/

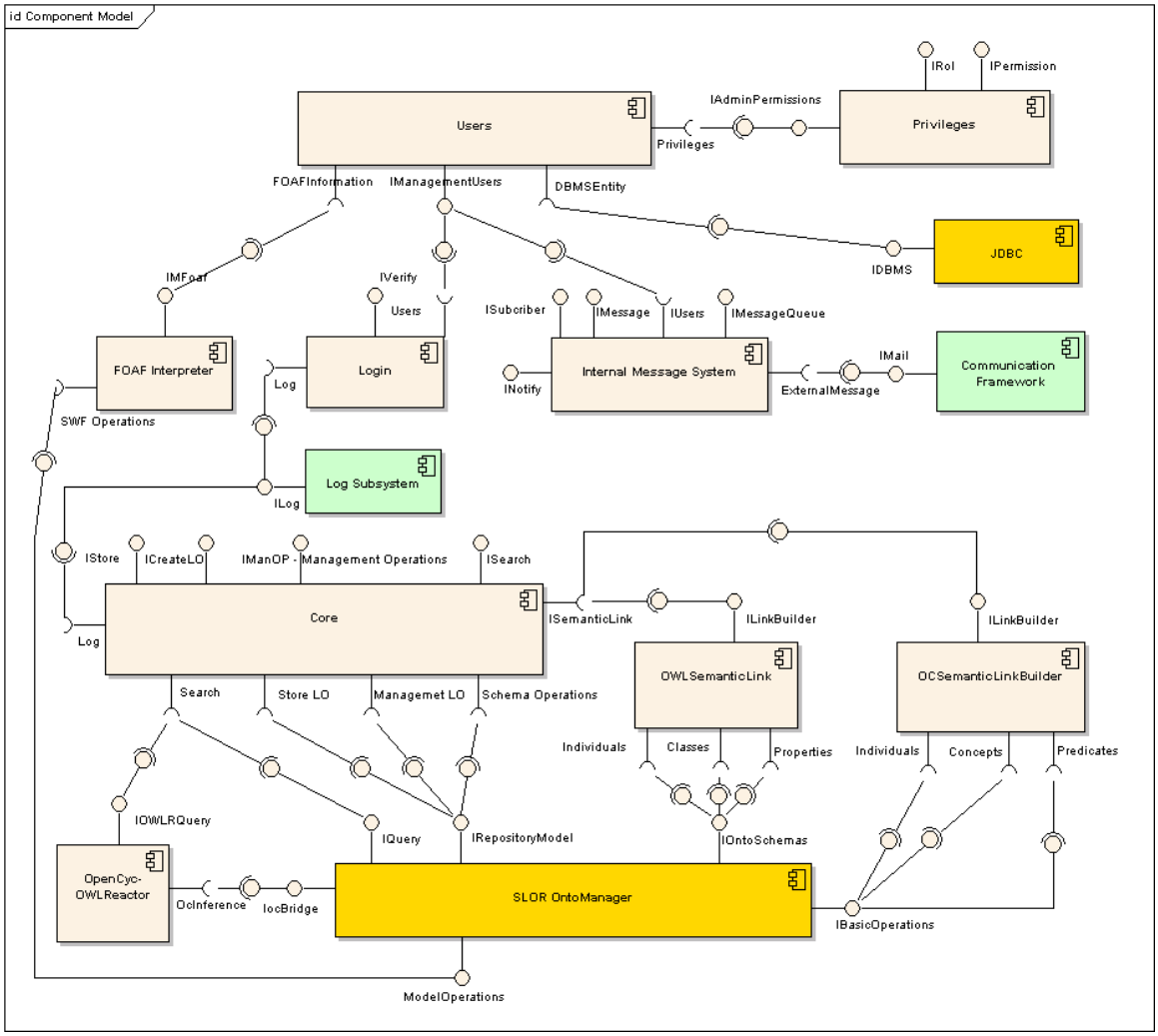

Figura 5.11: Diagrama de componentes del núcleo de SLOR.

de propiedad. La clase "*Descriptive_Property*" es la base del rango de las propiedades que permiten describir los distintos campos de los esquemas administrados por el repositorio (LOM, SCORM, IMS-LD,etc.).

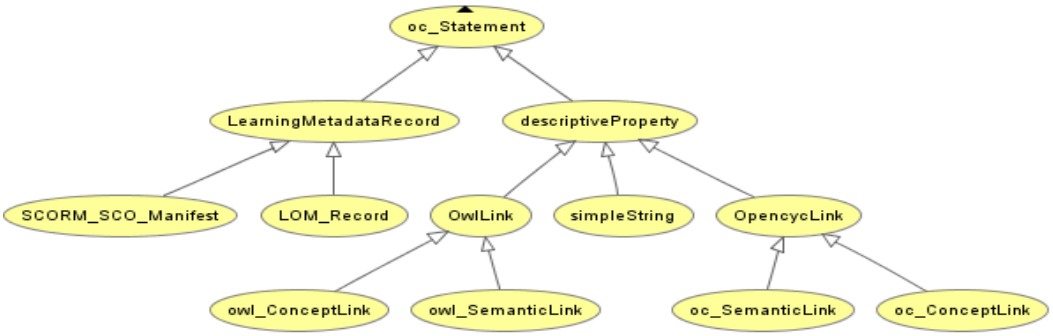

Figura 5.12: Meta-Esquema de descripción

Una descripción de propiedad puede ser una simple cadena escrita en algún lenguaje (instancia de la clase "*simpleString*") o una expresión que permita generar enlaces semánticos con otras ontologías. Como en el diseño final no se ha incluido la versión OWL de OpenCyc, por cuestiones de rendimiento y escasa definición, se ha optado por utilizar un mecanismo simple de enlace a través de las clases "*oc_ConceptLink*" y "*oc_SemanticLink*".

Cada instancia de la clase "*oc_ConceptLink*" contiene la referencia a un concepto de la ontología OpenCyc (en la implementación del prototipo el "nombre" del término, colección o individual). Normalmente se definen instancias de esta clase para campos del tipo "*Keyword*" o "*Classification*", en los que se referencian directamente conceptos que permiten optimizar los procesos de búsqueda y navegación. Cabe mencionar la utilidad de la clase "*oc_ConceptLink*" para clasificar los registros de metadatos insertados en SLOR mediante términos obtenidos de la base de conocimiento OpenCyc.

Para enlazar con un término de otra ontología definida en OWL se utiliza la clase "*owl_ConceptLink*". En el valor de la propiedad "*ref_Term*" se especifica el identificador del término asociado. Si existe un número elevado de conceptos referenciados de una ontología el sistema de persistencia de SLOR puede incluir el esquema de la ontología, dentro del modelo de la base de conocimiento.

En el caso de querer expresar conocimiento con el uso de las relaciones y predicados, el modo de inserción ha sido diseñado para la inclusión de forma flexible de cualquier expresión. La clase "*owl_SemanticLink*" permite utilizar propiedades de otras ontologías en cualquier registro del tipo "*Learning_MetadataRecord*". Para ello se han definido dos propiedades, "*property*" de rango "*owl:ObjectProperty*" para expresar la propiedad a enlazar y "*property_value*" de rango "*owl:Thing*" para definir el concepto con el que enlaza. Así, si fuera necesario almacenar un LO referente a un curso de arte sobre el barroco, habrá que insertar en el campo LOM "*Coverage*" de un registro de metadatos una instancia de la clase "*owl_SemanticLink*" con los valores "*aat:historicalPeriod*" para la propiedad "*property*" y "*aat:Baroque*" para la propiedad "*property_value*" definidos en el tesauro del AAT[8].

La utilización de la vasta base de conocimiento OpenCyc, ha planteado problemas tales como la incompatibilidad existente entre el modelo de descripción OWL-DL y el modelo original CycL. Es necesario utilizar un mecanismo similar de inserción de conocimiento al de CycL para mantener la compatibilidad con el modelo junto con la ontología de SLOR (ver sección 4.2.2), por ello se ha creado la clase "*oc_SemanticLink*" que permite describir predicados en modo CycL. Siguiendo un orden, se ha creado la propiedad "*oc_Predicate*" de tipo "*xsd:string*" para indicar la referencia del

[8]http://www.getty.edu/research/conducting_research/vocabularies/aat/

predicado OpenCyc a utilizar, la propiedad "*arity*" de tipo entero para expresar la aridad del predicado, y finalmente la propiedad múltiple "*oc_Args*" que enlaza los términos utilizados en la expresión del predicado. En cada argumento de tipo "*oc_ArgPredicate*" se define el nombre del término "*termArg*" y el número de orden del argumento "*norderArg*". Como ejemplo, en el listado de código 5.14 se muestra la representación en lenguaje OWL de la aserción:

```
(oc:courseOfStudyLevel #$LORID oc:DoctoralLevel)
```

Donde LORID es el identificador del objeto de aprendizaje, "*oc:courseOfStudyLevel*" el predicado que sirve para indicar el nivel de estudio requerido para un curso y "*oc:DoctoralLevel*" el nivel específico requerido para entender el objeto de aprendizaje.

Listado 5.14: Predicado OpenCyc descrito en OWL

```
<oc_SemanticLink rdf:ID="ocsl_LORID">
  <oc_Predicate rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      courseOfStudyLevel
  </oc_Predicate>
  <arity rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      2
  </arity>
  <oc_Args rdf:resource="#ocsl_LORID_arg1"/>
  <oc_Args rdf:resource="#ocsl_LORID_arg2"/>
</oc_SemanticLink>

<oc_ArgPredicate rdf:ID="ocsl_LORID_arg1">
  <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      LORID
  </termArg>
  <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      1
  </norderArg>
</oc_ArgPredicate>

<oc_ArgPredicate rdf:ID="ocsl_LORID_arg2">
```

```
22      <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
23          DoctoralLevel
24      </termArg>
25      <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
26          2
27      </norderArg>
28    </oc_ArgPredicate>
```

Métodos de apoyo a la construcción de expresiones semánticas

El proceso de construcción de este tipo de expresiones desde una interfaz de usuario es muy costoso. Por ello se ha incluido un conjunto de componentes visuales que facilitan la tarea de generar este tipo de enlaces entre conceptos de distintas ontologías. El sistema implementado es similar al sistema Intellisense de Microsoft, que permite evaluar y consultar de forma rápida el contenido de los esquemas de las ontologías tratadas en los enlaces a medida que el usuario va construyendo la expresión semántica.

En función del tipo de campo que se esté rellenando en una interfaz de usuario, se pueden generar expresiones semánticas que pueden enlazar conceptos existentes en la base de conocimiento OpenCyc ("*oc_ConceptLink*") o en otras ontologías definidas en el lenguaje OWL ("*owl_ConceptLink*" para clases o individuales owl), e incluir aserciones en forma de propiedades ("*owl_SemanticLink*") o predicados OpenCyc ("*oc_SemanticLink*").

En la interfaz ISearch existen cuatro métodos que permiten obtener un conjunto de términos o predicados de una ontología a través del componente "*SLOROntoManager*". Estos métodos son invocados por las interfaces que han sido diseñadas para introducir expresiones semánticas (lo cual se explicará más adelante, ver la sección 5.5.1), dos de ellos orientados a la base de conocimiento OpenCyc, y otros dos al manejo de

ontologías escritas en OWL.

- Para OpenCyc se han definido los métodos:

```
1 public String[] getOpencycCollections(String prefix,int
     maxResults,boolean comments);
2 public String[] getOpencycPredicates(String prefix,int maxResults
     ,boolean comments);
```

- Para ontologías escritas en el lenguaje OWL, se han definido los métodos:

```
1 public String[] getOWLClasses(String onto,String prefix,int
     maxResults,boolean comments);
2 public String[] getOWLProperties(String onto,String prefix,int
     maxResults,boolean comments);
```

Debido a la normativa de inserción utilizada dentro de la base de conocimiento OpenCyc, los métodos OpenCyc verifican la entrada recibida de una interfaz de usuario convirtiendo la letra inicial en mayúsculas si se desea obtener una lista de términos, o minúsculas si se desea obtener una lista de predicados. Gracias a esta normativa se ha creado un sencillo proceso que permite obtener los resultados invocando a la función "*getOpencycConcepts*" (descrita en la sección 5.2.6, listado de código 5.8), sin procesar una consulta más específica sobre la base de conocimiento OpenCyc. En el caso de realizar una búsqueda para un conjunto de colecciones o constantes se ha de invocar al método "*getOpencycCollections*" que devuelve los resultados que concuerdan con la expresión regular "*prefix**". La consulta en el lenguaje CycL posee la forma:

```
1 (constant-complete "prefix" :true)
```

Esta consulta se envía a la base de conocimiento OpenCyc a través del método "*getOpencycConcepts*" proporcionado por la interfaz "*IBasicOperations*" del componente "*HCIQueryComponent*". En la línea 7 del listado de código 5.15 se invoca

al método y se retornan los resultados de la consulta.

Listado 5.15: Método getOpencycCollections

```
1 public String[] getOpencycCollections(String prefix,int maxResults,
     boolean comments){
2         String filterPrefix="";
3         if (prefix.length()==1)
4             filterPrefix=prefix.toUpperCase();
5         else if (prefix.length()>1)
6             filterPrefix=prefix.substring(0,1).toUpperCase()+prefix.
                 substring(1);
7         return hciQ.getOpencycConcepts(filterPrefix,maxResults,comments
              );
8 }
```

Por otro lado, en los componentes de las interfaces que muestran según va escribiendo el usuario una lista de predicados OpenCyc, se invoca la función de ejecución en servidor "*getOpenCycPredicates*". Al igual que en el método anterior, la consulta se construye y envía a OpenCyc tras la invocación del método "*getOpencycConcepts*" realizado en la línea 7 del listado de código 5.16.

Listado 5.16: Método getOpencycPredicates

```
1 public String[] getOpencycPredicates(String prefix,int maxResults,
     boolean comments){
2        String filterPrefix="";
3        if (prefix.length()==1)
4             filterPrefix=prefix.toLowerCase();
5        else if (prefix.length()>1)
6             filterPrefix=prefix.substring(0,1).toLowerCase()+prefix.
                 substring(1);
7        return hciQ.getOpencycConcepts(filterPrefix,maxResults,comments
             );
8 }
```

En cambio, cuando se selecciona en una interfaz visual una ontología definida en el lenguaje OWL, el sistema ha de procesar su esquema de manera distinta a como se procesa para OpenCyc, ya que las bibliotecas de acceso utilizadas en la implementación son distintas. El subsistema de persistencia aporta las operaciones necesarias para consultar y manipular esquemas definidos en lenguaje OWL (ver sección 5.2.3), la interfaz "*IRepositoryModel*" incluye los métodos necesarios para obtener el conjunto de clases y propiedades de un modelo OWL. Estos métodos son procesados y filtrados mediante las instrucciones de librería joQL (de igual forma que en el método "*queryPredicate*" del listado de código 5.12 comentado en la sección 5.2.6) para obtener el conjunto de clases o propiedades en las que concuerda el identificador con la expresión regular "*prefix**".

Listado 5.17: Método getOWLClasses

```
1 public String[] getOWLClasses(String onto,String prefix,int maxResults
    ,boolean comments){
2         reactor.setCurrentOWLOntology(onto);
3         return reactor.getListOWLClasses(prefix,maxResults,comments);
4 }
```

Tanto en el listado de código 5.17 como en el 5.18, los métodos reciben como parámetros el indentificador de la ontología "*onto*", el prefijo que se desea buscar "*prefix**", el tope máximo de resultados devueltos "*maxresults*" y el flag que indica si se han de recuperar también los comentarios realizados de las clases o propiedades. Los dos métodos poseen una estructura similar, en la línea 2 se establece la ontología que ha de procesar el componente recibido como parámetro invocando al método "*setCurrentOWLOntology*" de la referencia "*reactor*" a la interfaz "*IOWLRQuery*".

Listado 5.18: Método getOWLProperties

```
1 public String[] getOWLProperties(String onto,String prefix,int
     maxResults,boolean comments){
2         reactor.setCurrentOWLOntology(onto);
3         return reactor.getListOWLProperties(prefix,maxResults,comments)
              ;
4 }
```

Inferencia

En el siguiente capítulo se analizarán los resultados obtenidos tras ejecutar operaciones semánticas descritas con mecanismos de expresión semánticos definidos en la sección 5.3.1. A modo resumen, se exhibe una amplia mejora respecto al tratamiento completo de OpenCyc en OWL desde múltiples perspectivas:

- Mejor nivel de expresividad, aprovechando la potencia descriptiva de Cyc.
- Mejora de rendimiento, respecto a la versión OWL de OpenCyc.

En contra, existe un problema de inferencia y recuperación de información. Las reglas de inferencia definidas en OWL no coexisten con el modelo descrito en OpenCyc. Para poder solucionar el problema y aprovechar la base de conocimiento OpenCyc se ha creado un sistema que habilita la coexistencia de ambos modelos dentro de un marco común de inferencia. El sistema lo hemos bautizado como "OpenCyc-OWL-Reactor".

Como ejemplo, si se desea realizar la búsqueda de todos los cursos de formación continua, el sistema construye una lista basada en los recursos que posean la propiedad "*learning-ResourceType*" ligada al concepto de la base de conocimiento OpenCyc "*oc:ContinuingEducationCOS*". Además, en paralelo, se consulta la jerarquía de herencia existente, obteniendo la listas de clases posibles que también se

verifican para el valor de la propiedad "*learning-ResourceType*", como son los términos "*oc:CourseOfStudy*", "*oc:Training*" u "*oc:FormalSchooling*".

Para conseguir estos resultados se han definido en la interfaz "*IOWLRQuery*" dos funciones implementadas dentro del componente "*OpenCyc-OWLReactor*" que permiten ejecutar procesos similares a los de un razonador transitivo:

- "*List getOpenCycUpperTerms(String term,int deep)*": obtiene los términos correspondientes a la jerarquía superior del término pasado como parámetro "*term*". Es posible especificar el límite de profundidad de exploración por la jerarquía superior se recibe en el parámetro "*deep*".

- "*List getOpenCycDerivedTerms(String term,int deep)*": obtiene todos los términos derivados del término pasado como parámetro "*term*". El límite de profundidad de exploración por la jerarquía inferior se recibe en el parámetro "*deep*".

En la figura 5.13 se puede apreciar el resultado obtenido tras la ejecución de estas funciones con el término "*Airplane*". La función "*getOpenCycUpperTerms*" recorre hasta el límite especificado nivel a nivel, utilizando una técnica de búsqueda en profundidad sobre la estructura jerárquica definida en OpenCyc, iniciada desde el punto de partida sobre las aserciones:

```
1 (isa #$AirPlane ?X)
```

El proceso continúa de forma recursiva hasta los términos que se encuentran en el nivel de jerarquía superior. En la figura 5.13 se representa a modo de ejemplo el proceso seguido con el término "*TransportationDevice-Vehicle*".

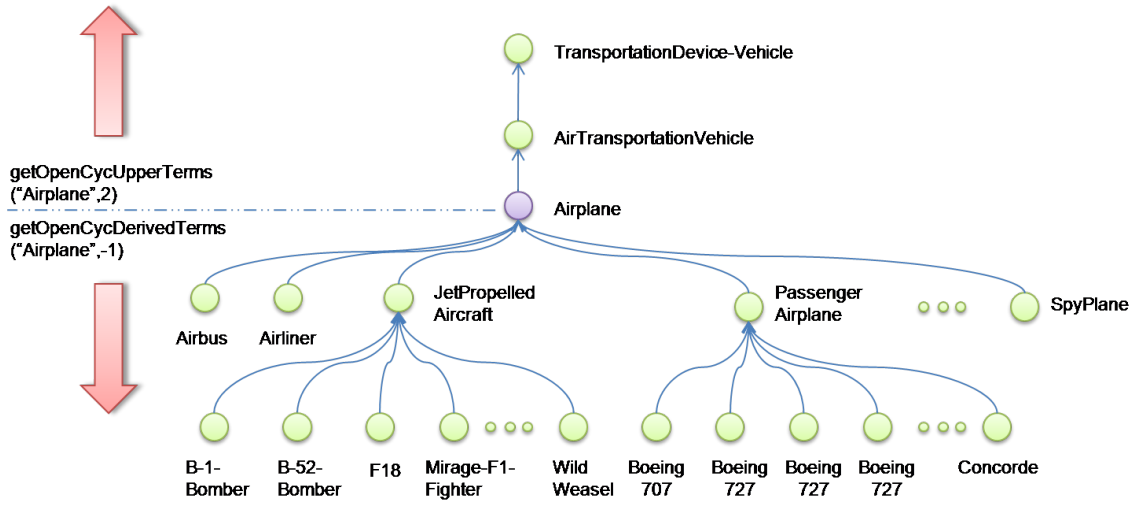

Figura 5.13: Inferencia OpenCyc - clases superiores y derivadas

También se representa de forma gráfica el resultado obtenido tras la ejecución de la función "*getOpenCycDerivedTerms*". Como puede apreciarse se corresponde a todas las clases derivadas de forma directa e indirecta hasta el final de la jerarquía, ya que el proceso no para hasta que llega al final de la jerarquía, debido a que no existe un límite de profundidad definido (parámetro "*deep=-1*").

La funciones comentadas anteriormente son utilizadas por el método de búsqueda "*executeQueryandInference*" de la clase "*QueryManager*". El constructor de la clase recibe como parámetro una lista de objetos del tipo "*InferenceTerm*" que especifican los valores sobre los que se han de ejecutar procesos de razonamiento transitivo necesarios para expandir la consulta. De este modo, en una consulta en la que se deseen obtener todos los objetos de aprendizaje sobre los aviones a reacción que expliquen con alto detalle algún tipo de motor a reacción (en especial los IAEV2500) debe proporcionar, desde la interfaz de usuario de búsqueda con apoyo del módulo de búsqueda "*Search*" la siguiente sentencia:

```
PREFIX edu: <http://slor.sourceforge.net/ontology/edutest.owl>
PREFIX egn: <http://slor.sourceforge.net/ontology/engines.owl>
SELECT ?object ?title
    WHERE {    ?object :hasAssociatedMetadataRecord ?mt .
               ?mt :keywords ?ocl .
               ?ocl  rdf:type :oc_ConceptLink .
               ?ocl  :concept "JetPropelledAircraft" .
               ?mt :description ?owl .
               ?owl  rdf:type :owl_SemanticLink .
               ?owl  :property   edu:explainHighDetail .
               ?owl  :property_value egn:IAEV2500 .
    }
```

Donde el término "*oc:JetPropelledAircraft*" es el valor a procesar para realizar una inferencia de tipo TOP-DOWN, sin nivel de profundidad establecido, y el término "*eng:IAEV2500*" como término a procesar con inferencia de tipo BOTTOM-UP con límite de profundidad 1. Ambos construidos con la siguiente función proporcionada por el módulo de búsqueda:

```
public void addNewInferenceTerm(String IDOntology,
                                String term,
                                InferenceType inftype,
                                int deep ){
    InferenceTerm inf=new InferenceTerm(term);
    inf.setOntology(IDOntology);
    inf.setInferenceType(inftype);
    inf.setDeep(deep);
    lstInfTerms.add(inf);
}
```

El módulo de búsqueda posteriormente invoca al método "*semanticLOSearch*" de la interfaz "*ISearch*" proporcionada por el núcleo, con el objetivo de construir la sentencia final apoyándose del componente "*OWLOpenCycReactor*" del núcleo. El método "*semanticLOSearch*" (ver siguiente sección) invoca al método interno

"*executeQueryAndInference*" tras construir una instancia de la clase "*QueryBuilder*", por cada término se analiza el tipo de inferencia a realizar, la ontología a la que corresponde y los parámetros necesarios. En el caso del ejemplo mencionado anteriormente, para el término "*oc:JetPropelledAircraft*" de OpenCyc se realiza una consulta transitiva sin límite de profundidad, al especificar el tipo de inferencia "*TOP-DOWN*" y para el término "*IAEV2500*" se realiza una consulta de tipo "*BOTTOM-UP*" obteniendo como resultado las clases que corresponden al nivel directo de la jerarquía superior. Con los resultados obtenidos posteriormente se anexan todas las clases derivadas a la consulta SPARQL original:

```
PREFIX edu: <http://slor.sourceforge.net/edutest.owl>
PREFIX egn: <http://slor.sourceforge.net/engines.owl>
SELECT ?object ?title
    WHERE {    ?object :hasAssociatedMetadataRecord ?mt .
               ?mt :keywords ?ocl .
               ?ocl  rdf:type :oc_ConceptLink .
               {?ocl  :concept "JetPropelledAircraft"}
          UNION{?ocl :concept "B-1-Bomber"}
          UNION{?ocl :concept "B-52-Bomber"}
          UNION{?ocl :concept "F18"}
          UNION{?ocl :concept "Mirage-F1-Fighter"}
          ...
          UNION{?ocl :concept "WildWeasel"}.
               ?mt :description ?owl .
               ?owl  rdf:type :owl_SemanticLink .
               ?owl  :property   edu:explainHighDetail .
               {?owl  :property_value egn:IAEV2500}
          UNION{?owl :property_value egn:JetEngine}.
    }
```

La combinación de otros lenguajes de consulta y razonadores OWL pueden simplificar los procesos descritos anteriormente, aún así es necesario establecer el puente de comunicación con la base de conocimiento OpenCyc para que el repositorio semántico de objetos de aprendizaje posea un modelo consistente.

En la implementación realizada del prototipo se han construido los procesos básicos de razonamiento transitivo para consultas realizadas con el lenguaje SPARQL. El componente "*OpenCyc-OWLReactor*" posee una estructura de clases extensible que permite ampliar las funciones de inferencia y deducción combinadas con razonadores OWL y otros motores de inferencia.

5.3.2. Componente "*Core*"

El componente "*Core*" aporta las operaciones de gestión (a nivel abstracto) de los objetos de aprendizaje albergados en el repositorio. Todas las operaciones descritas a continuación utilizan los objetos de aprendizaje de la jerarquía de clases de la ontología de SLOR (ver las interfaces representadas en la figura 5.8). El componente "*Core*" posee las interfaces:

- IStore: interface encargada del almacenamiento de los objetos RDF/OWL creados por las interfaces de los módulos.

- ICreateLO: incluye las operaciones básicas de construcción de objetos de aprendizaje para los objetos que no poseen un esquema definido utilizando las definiciones "*LearningObject-AsAnything, LearningObject-AsAnythingDigital, LearningObject- AsAnythingWithEducationalPurpose*".

- IManOP: aporta las operaciones de gestión de los objetos de aprendizaje albergardos en el repositorio.

- ISearch: interfaz que incluye las operaciones de búsqueda sobre los objetos albergados en el repositorio de objetos de aprendizaje. Por un lado permite obtener objetos que pertenecen a una clase determinada, y por otro ejecutar consultas SPARQL para la localización de recursos.

Almacenar

Cuando se utiliza el modelo persistente descrito, todos los cambios reflejados en las instancias de las clases quedan automáticamente almacenados en la base de datos si se utiliza la factoría de construcción de objetos "*OWLDatabaseKnowledgeBaseFactory*". Esta situación genera un problema a la hora de operar con las clases del componente "*SLOR Model*" si se inicializa la factoría "*OntoSlorFactory*" con el modelo persistente. Cada valor actualizado por una orden de la interfaz genera automáticamente los cambios en el sistema de almacenamiento persistente.

Para aportar mayor versatilidad en la construcción del objeto de aprendizaje que se desea insertar por parte del módulo responsable de la generación del objeto, se ha decidido utilizar una factoría que construye objetos en memoria y posteriormente los actualiza en el sistema persistente utilizando el método "*insertMemoryModel*". De forma gráfica, en la figura 5.14 se representa la construcción de un modelo OWL/RDF en memoria de la instancia "*INDV1-OA1*" de la clase "*LearningObject-LOM*" con el registro de metadatos asociado "*REC-01*". Desde la interfaz de usuario se han ido añadiendo determinados valores y expresiones en las propiedades de los campos del registro, como es el caso de la descripción, donde se han enlazado los valores de la propiedad "*explainHighDetail*" con términos del tesauro NCI-OWL[9]. En el caso de la propiedad "*keyword*", desde la interfaz de usuario se ha insertado una expresión que indica el enlace al concepto DNA de la ontología OpenCyc, por ello en el grafo representado en la figura aparece enlazado a través del valor de la clase "*oc_concept*". En cambio en la propiedad "*coverage*" se ha decidido insertar una aserción utilizando la clase "*oc_conceptLink*" para enlazar con un predicado de OpenCyc. Como puede apreciarse, la complejidad del modelo aconseja utilizar en lugar de una instancia en memoria de un modelo OWL, otras estructuras de programación más básicas.

[9]ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Thesaurus/Thesaurus_07.10d.OWL.zip

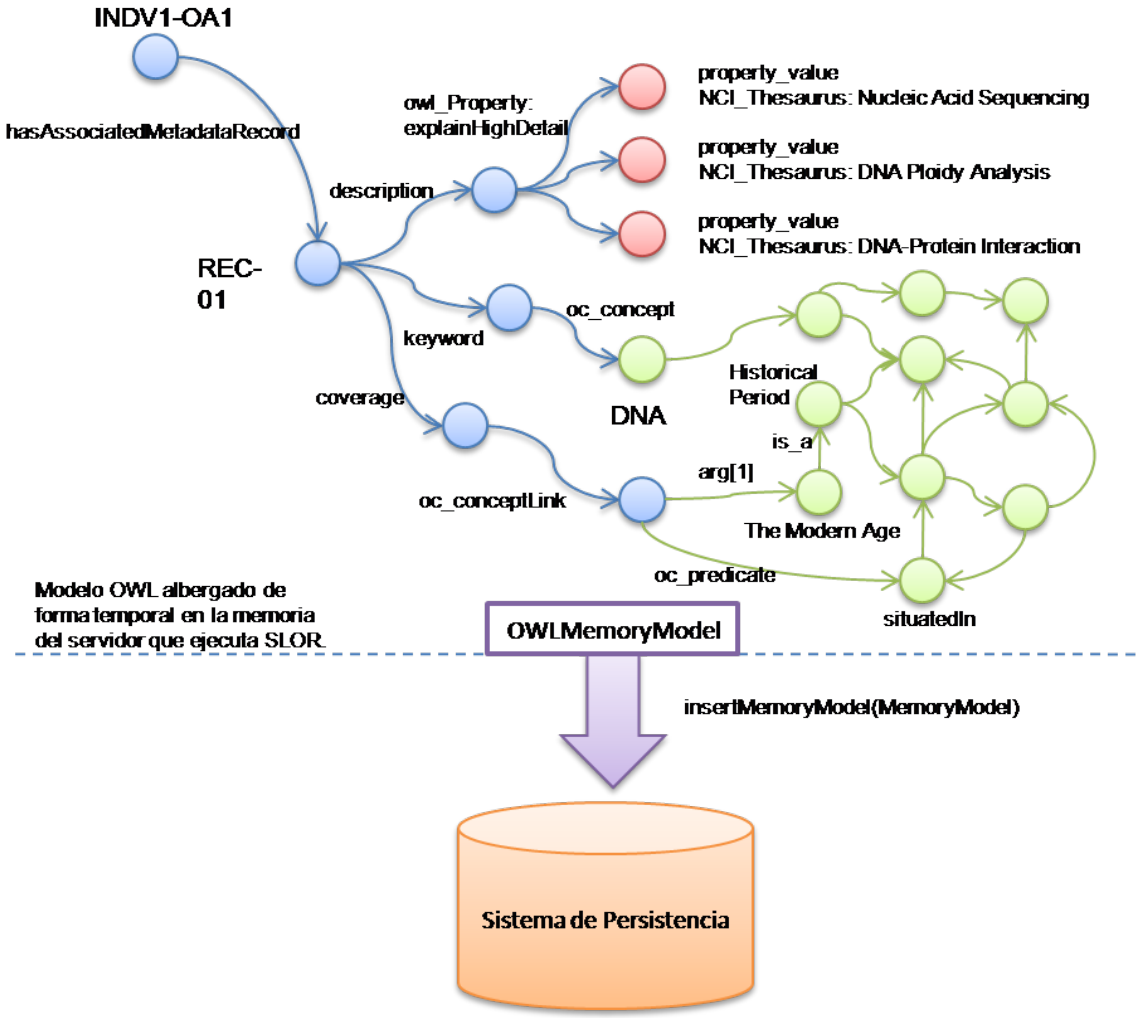

Figura 5.14: Inserción de objetos OWL/RDF en el sistema de persistencia.

La instancia de una factoría que permita construir un objeto OWL/RDF como el representado en la figura 5.14 se realiza con el siguiente código:

```
String uri = "http://localhost:8080/slor/ontology/slor.owl";
OWLModel owlMemoryModel;
slor.ontology.OntoSlorFactory owlMemoryFactory;
owlMemoryModel=ProtegeOWL.createJenaOWLModelFromURI(uri)
owlMemoryFactory=new slor.ontology.ontoSlorFactory(owlMemoryModel);
```

En la primera línea se especifica la localización del esquema del modelo. En la tercera línea se construye el modelo en memoria. Finalmente en la última línea se instancia la factoría "*owlMemoryFactory*" que permite construir los objetos RDF/OWL (ver sección 5.2.3), como los descritos en la figura 5.14.

Las operaciones de construcción del objeto son proporcionadas por el esquema del modelo. La responsabilidad de construcción se delega sobre el módulo que desee realizar una operación de inserción en el sistema. En último término se lleva a cabo la construcción del modelo en memoria tras todas las comprobaciones y validaciones realizadas sobre los datos obtenidos de la interfaz, para lo que se ha de ejecutar el método "*insertMemoryModel*" proporcionado por la interfaz "*IStore*". El código se muestra en el siguiente listado:

```
1 public void insertMemoryModel(OWLModel memory){
2     OWLModel persistentstorage=app1.getPersistentOWLModel();
3     OWL2OWLCopier cpr=new OWL2OWLCopier(memory,persistentstorage);
4     cpr.run();
5 }
```

La clase "*OWL2OWLCopier*" implementa los procesos de copia entre dos modelos OWL, pertenece al marco de trabajo de Protégé en el paquete de gestión de almacenamiento "*edu.stanford.smi.protegex.owl.storage*".

Búsqueda semántica

Una de las operaciones más complejas de cuantas proporciona el núcleo es la realización de una búsqueda semántica de objetos de aprendizaje. La estructura del modelo sobre el que se realiza una consulta es bastante flexible y además soporta diferentes tipos de axiomas entre propiedades que mantienen consistente el modelo. A continuación se describen los cuatro tipos de búsqueda definidos:

- Obtención directa de registros: métodos que reciben un identificador unívoco que permite recuperar el objeto de aprendizaje directamente del sistema de persistencia.

- Búsqueda semántica simple: implementada en un conjunto métodos que permiten consultar campos relevantes, como son el título ("*Title*"), las palabras clave ("*Keywords*") o la clasificación ("*Classification*"), este último muy utilizado en la navegación de contenidos.

- Búsqueda semántica sobre aserciones: se aporta un método para construir y ejecutar una consulta que evalúe las expresiones semánticas insertadas en el modelo.

- Búsqueda semántica e inferencia: similar al anterior pero se especifica sobre cada expresión que forma la consulta si se ha de desencadenar algún proceso de inferencia.

Búsqueda directa

La interfaz de la factoría "*OntoSlorFactory*" del subsistema de persistencia aporta un conjunto de funciones básicas que permiten recuperar la estructura y contenido de un registro de metadatos de un objeto de aprendizaje a partir del identificador interno. Desde el núcleo, los módulos pueden invocar este tipo de funciones a través de la interfaz "*IManOP*", que permite obtener la referencia utilizando el método "*OntoSlorFactory getSLORFactoryObjects()*". Ejemplos de los prototipos de estas funciones incluidas en la interfaz son:

```
1 public LearningObject_Generic
2     getLearningObject_Generic(String name);
3 public LearningObject_AsAnything
4     getLearningObject_AsAnything(String name);
```

```
5 public LearningObject_AsAnythingDigital
6     getLearningObject_AsAnythingDigital(String name);
7 public LearningObject_AsAnythingWithEducationalPurpose
8     createLearningObject_AsAnythingWithEducationalPurpose(String name)
          ;
9 public LearningObject_LOM getLearningObject_LOM(String name);
```

Búsqueda semántica simple

Si se desea realizar una simple consulta utilizando la propiedad "*Title*" del objeto de aprendizaje se ha de invocar al método "*searchLORbyTitle*" de la interfaz "*ISearch*". El código del método (listado 5.19) que define una secuencia de instrucciones similar utilizada en el resto de métodos de tipo de busqueda simple sigue la siguiente lógica:

Entre las líneas 6 y 9 se define la consulta SPARQL, como puede apreciarse en la línea 9 se concatena la expresión de consulta con el parámetro recibido "*paramTitle*". Posteriormente en la línea 10 se obtiene la referencia al subsistema de persistencia (componente "*SLOROntoManager*"), seguidamente se obtiene la referencia a la interfaz de gestión y proceso de consultas "*QueryManager*", finalmente en la línea 12 se ejecuta la consulta invocando al método "*executeQuery*" especificando en la consulta el lenguaje utilizado.

La función devuelve una lista de objetos de tipo "*Result*" que posteriormente será mostrada en un componente visual de tipo tabla. Desde la línea 13 hasta la 26 del método se realizan las siguiente acciones:

1. En la línea 13 se instancia la lista de resultados.
2. Entre las líneas 14 y 26 se ejecuta el grupo de instrucciones que van a rellenar la lista de resultados.

3. En la línea 15 se obtiene uno de los resultados contenido en el objeto "*r*" instancia de la clase "*QueryResults*". El tipo devuelto corresponde al tipo "*Map*".

4. En la línea 16, se obtiene la instancia contenida dentro del mapa utilizando el identificador de la variable SPARQL "*object*".

5. En la línea 18 se realiza la conversión al tipo genérico de objeto de aprendizaje.

6. Entre las líneas 19 y 26 se recogen los datos generales del objeto de aprendizaje y se insertan dentro de la lista de resultados.

Listado 5.19: Búsqueda de un objeto de aprendizaje por la propiedad "*Title*"

```
1  public LinkedList<Result> searchLORbyTitle(String paramTitle){
2      if (SlorParameters.QueryLanguage==SlorParameters.LanguageType.
           SPARQL)
3      {
4          QueryResults r;
5          try{
6              String query="SELECT ?object ?title WHERE
7                         {  ?object :Title ?title . " +
8                             "FILTER(regex(str(?title),\""
9                             + paramTitle + "\")) } " ;
10             PersistentSubsystem ps=getAppContext().
                   getPersistentSubsystem();
11             QueryManager qm=ps.getQueryManager()
12             r=qm.executeQuery(query,SlorParameters.LanguageType.SPARQL)
                   ;
13             LinkedList<Result> lst=new LinkedList<Result>();
14             while (r.hasNext()){
15                 java.util.Map m=r.next();
16                 OWLIndividual x=(OWLIndividual)m.get("object");
17                 slor.ontology.LearningObject_Generic g=null;
18                 g=(slor.ontology.LearningObject_Generic)x.as(slor.
                       ontology.LearningObject_Generic.class);
19                 Result zr=new Result();
```

```
                    if (g.hasIdentifier())
                        zr.setID(g.getIdentifier());
                    else zr.setID("");
                    if (g.hasTitle())
                        zr.setName(g.getTitle());
                    else zr.setName("");
                    lst.add(zr);
                }
            }catch(Exception err){
                throw(new QueryError());
                return null;
            }
        }else if (SlorParameters.QueryLanguage==SlorParameters.
            LanguageType.SeRQL){
            // Código para la construcción de la consulta de una sentencia
                SeRQL
        }
    }
```

Si el usuario desea obtener más información sobre un objeto de aprendizaje, se aporta un conjunto de funciones que permiten recuperar un objeto de aprendizaje a partir del identificador público:

```
LearningObject_Generic getLearningObject(String ID);
```

Este tipo de funciones realizan una consulta similar a "*searchLORbyTitle*", sobre la propiedad genérica "*Identifier*" definida en el término de mayor nivel de abstracción del modelo de SLOR ("*LearningObject_Generic*"). Posteriormente, como se describirá en la siguiente sección, se ejecutarán las instrucciones que permiten obtener el tipo al cual pertenece el objeto y se podrá mostrar el contenido del registro de metadatos.

Búsqueda semántica sobre aserciones

En este tipo de búsqueda, el usuario compone la consulta utilizando una interfaz de usuario como asistente (ver sección 5.5). Desde la interfaz se pueden consultar los esquemas de ontologías insertados dentro del repositorio (tanto OWL como la base de conocimiento OpenCyc), así como la inclusión en las condiciones de búsqueda de enlaces con términos, propiedades o predicados existentes de este tipo de esquemas.

La consulta se construye en el módulo "*Search*" y se invoca a través del método "*queryLORs*" de la interfaz "*ISearch*" del núcleo. El método posee la misma forma que el expresado en el listado de código 5.19, con la peculiaridad de que la consulta y el lenguaje son recibidos como parámetro.

Los diferentes tipos de estructuras de restricciones de existencia tenidos en cuenta en la construcción de consultas son:

- Restricción existencial - Términos OWL: permite asignar un término como restricción de existencia sobre una propiedad asociada a algún campo de un registro de metadatos. Por ejemplo, en el caso de la propiedad "*keywords*" asociada al tipo de registro de metadatos "*LOM_Record*" se puede incluir la restricción de existencia de asociación sobre el concepto "*nci:DNA_Analysis*" de la ontología Nacional Cancer Institute. A continuación se muestra un fragmento de consulta escrita en lenguaje SPARQL generada por este tipo de restricción:

```
1       ?object :hasAssociatedMetadataRecord ?mt .
2               ?mt  rdf:type  :LOM_Record .
3               ?mt :keywords ?owlc .
4               ?owlc  rdf:type :owl_ConceptLink .
5               ?owlc  :ref_Term nci:DNA_Analysis .
```

- Restricción existencial - Términos OpenCyc: este tipo de restricción incluye

un valor directo de correspondencia sobre el valor del rango de la propiedad "*concept*" de tipo "*xsd:String*". En el ejemplo mostrado en el siguiente listado, se aprecia la correspondencia del concepto "*B-1-Bomber*" con el valor de la propiedad "*concept*" asociada al tipo "*oc_ConceptLink*":

```
1       ?object :hasAssociatedMetadataRecord ?mt .
2              ?mt  rdf:type  :LOM_Record .
3              ?mt :keywords ?ocl .
4              ?ocl  rdf:type :oc_ConceptLink .
5              ?ocl  :concept "B-1-Bomber" .
```

- Restricción existencial relacional - Propiedades OWL: gracias a este tipo de restricción se permiten generar expresiones de consulta sobre las relaciones existentes definidas en el modelo del repositorio. Por ejemplo, puede generarse desde la interfaz de usuario la restricción existencia de búsqueda "*edu:historicalPeriod* $\exists$ *aat:Baroque*" sobre la propiedad coverage de un tipo de registro "*LOM_Record*". El código incluido en la consulta SPARQL al ejecutar el método "*queryLORs*" corresponde con el mostrado en el siguiente listado de código:

```
1       ?object :hasAssociatedMetadataRecord ?mt .
2              ?mt  rdf:type  :LOM_Record .
3              ?mt :coverage ?owlr .
4              ?owlr  rdf:type :owl_SemanticLink .
5              ?owlr  :property   edu:historicalPeriod .
6              ?owlr  :property_value aat:Baroque
```

- Restricción existencial relacional - Predicados OpenCyc: con este tipo de restricción se incluyen sobre la consulta expresiones de restricción de existencia sobre las aserciones realizadas con predicados OpenCyc incluidas en el modelo del repositorio como instancias de la clase "*oc_SemanticLink*". A modo de ejemplo, un usuario puede construir una expresión de consulta desde la interfaz de usuario indicando obtener todos los registros de metadatos que describen

una imagen por satélite.

```
1   ?object :hasAssociatedMetadataRecord ?mt .
2             ?mt  rdf:type  :LOM_Record .
3             ?mt :description ?ocl .
4             ?ocl  rdf:type :oc_SemanticLink .
5             ?ocl  :predicate   "isa" .
6             ?ocl  :oc_Args    ?arg .
7             ?arg  :norderArg   2 .
8             ?arg  :termArg     "SatelliteImage".
```

Cada uno de los ejemplos expuestos anteriormente muestran la generación del código de la consulta tras la primera inserción de una restricción. Por ello se incluyen al principio las líneas (1 y 2) donde se obtiene la referencia a los registros de metadatos. En sucesivas inserciones de restricciones sobre un mismo tipo de registro no se incluyen más.

Búsqueda semántica e inferencia

Desde la interfaz de usuario, se indican las instrucciones para la construcción de las consultas con las restricciones explicadas anteriormente. También sobre cada restricción incluida se puede especificar un tipo de inferencia a realizar con el parámetro, en el caso del prototipo se han implementado restricciones existenciales transitivas en dos tipos de dirección ascendente ("*BOTTOM-UP*") o descendente ("*TOP-DOWN*").

Las restricciones existenciales transitivas en dirección "*BOTTOM-UP*", si es necesario y el lenguaje de consulta no lo soporta (como es en el caso de SPARQL), realizan una consulta transitiva sobre la jerarquía superior del modelo hasta el nivel de profundidad indicado. De esta forma se obtienen todos los elementos que posteriormente serán utilizados en el método de generación de la consulta final, como es en el caso de la generación de la consulta con el lenguaje SPARQL ya que este lenguaje carece

de capacidades de inferencia.

En el caso de las restricciones existenciales transitivas en dirección "*TOP-DOWN*", al ser procesadas se ejecutan las instrucciones de consulta sobre la jerarquía de clases derivadas hasta el nivel de profundidad indicado. De esta forma se obtienen todos los elementos que posteriormente son utilizados en el proceso de construcción de la consulta final.

Si el razonador OWL utilizado soporta un lenguaje de consulta capaz de desencadenar este tipo de procesos, no será necesario ejecutar parte de las acciones comentadas, ya que él mismo se encarga de procesarlas. Los métodos de tratamiento de este tipo de consultas han de tener en cuenta este tipo de restricciones existenciales transitivas y otro tipo de inferencias que pueden de ser procesadas por el razonador para la construcción de la consulta final.

En la interfaz "*ISearch*" del componente "*Core*" del núcleo de SLOR se ha incluido el método "*inferenceQueryLORs*" que recibe los parámetros construidos en la interfaz de usuario a través del módulo de búsqueda:

- La consulta a ejecutar.
- El tipo de lenguaje utilizado en la definición de la consulta.
- Una lista de elementos que referencian los términos de la consulta sobre los que se ha de realizar una operación de inferencia.

El método "*inferenceQueryLORs*" utiliza el componente "*Opencyc-OWLReactor*" (comentado en la sección 5.3.1) para realizar las operaciones necesarias de inferencia sobre cada restricción.

Rol	**Permisos**
GUEST	SEARCH
META-CREATOR	SEARCH,CREATE,MODIFY, DELETE
PEER-REVIEW	SEARCH, REVIEW

Tabla 5.1: Roles de usuario

5.3.3. Administración y gestión de usuarios

El núcleo de SLOR proporciona un sistema de administración y gestión de los distintos usuarios que utilizan el repositorio. En el diseño e implementación del prototipo se han tenido en cuenta los componentes "*Users*", "*Privileges*", "*FOAF Interpreter*" y "*Login*".

El componente "*Users*" proporciona las operaciones principales de creación de usuarios, búsqueda y verificación de permisos. En el proceso de registro, el usuario puede especificar un archivo de descripción FOAF. Se ha proporcionado esta característica para poder obtener mayor información de los diferentes usuarios que utilicen el sistema.

Por otro lado, el componente "*Privileges*" habilita las funciones de gestión de roles de SLOR. Para gestionar el control de acceso de los distintos elementos administrados por el repositorio, se han definido las acciones de creación ("*CREATE*"), borrado ("*DELETE*") y modificación ("*MODIFY*"), búsqueda y navegación ("*SEARCH*"), e inserción de comentarios de un revisor ("*REVIEW*"). También se han definido 3 tipos de roles, usuario sin registrar ("*GUEST*"), creador de metadatos ("*META-CREATOR*"), revisor ("*PEER-REVIEW*"). Los permisos asociados se describen en la tabla 5.1.

En general todos los que deseen publicar sus contenidos didácticos en el repositorio necesitan registrarse previamente para adquirir el rol de creador de metadatos "*META-CREATOR*". El administrador del sistema puede seleccionar los revisores de

contenidos previa solicitud de aquellos usuarios que lo deseen y posean un cierto criterio de evaluación demostrado (ej. profesores). La revisión puede asignarse sobre las categorías de contenidos gestionadas por el repositorio, así un revisor de objetos de aprendizaje de Bioquímica no podrá realizar comentarios sobre los objetos de aprendizaje de Informática.

Finalmente, el componente "*Login*" proporciona las operaciones de validación (interfaz "*IVerify*") para que un usuario pueda acceder al repositorio de objetos de aprendizaje y obtener todos los permisos asignados.

5.4. Módulos

Los módulos son entidades diseñadas para separar las diferentes funcionalidades ofrecidas por el repositorio. Cada módulo aporta un conjunto de interfaces de usuario y clases que se comunican con el núcleo de SLOR. En la figura 5.15 se representa el diagrama de componentes que incluye los primeros módulos e interfaces de usuario planteados en el diseño del prototipo SLOR.

Como puede apreciarse en la figura 5.15 se muestran los módulos, interfaces de usuario y las dependencias existentes entre los módulos. Existen diferentes tipos de módulos: los correspondientes a la gestión de registros de metadatos, como son "*ISBD-Module*" o "*LOM-Module*", y los módulos que aportan funcionalidades especificas para todos los registros de metadatos albergados en el repositorio, como son el módulo "*Reviews-Module*" o "*Security-Module*".

Cada módulo relacionado con la gestión y administración de registros de metadatos con un tipo de esquema específico, incluye una interfaz de usuario que contiene

todos los campos del esquema y además un conjunto de clases que permiten verificar y procesar la información recibida de los usuarios a través de la interfaz para posteriormente ser enviada al núcleo para ser almacenado en el sistema de persistencia. Este tipo de módulos gestiona un modelo OWL en memoria como el descrito en la sección 5.3.2, mientras que la interfaz permite construir las expresiones semánticas que son verificadas y añadidas en el modelo representado en memoria. Posteriormente, cuando el usuario desea hacer persistente el registro insertado, invoca la operación que será enviada al núcleo para copiar el modelo escrito en memoria en el subsistema de persistencia.

El módulo "*Reviews-Module*" incluye la funcionalidad específica para insertar comentarios y evaluaciones sobre los objetos de aprendizaje descritos por los registros de metadatos del repositorio. El usuario revisor puede insertar un comentario utilizando la interfaz "*AddReview*", que invoca las operaciones de la interfaz "*set*" para mantener persistentes los comentarios.

Todos los módulos representados poseen relaciones con el módulo de seguridad "*Security-Module*", esto es debido a que para cada operación recibida desde una interfaz de usuario se verifica si el usuario que la ha desencadenado posee privilegios para invocar a la operación. La gestión de la sesión de un usuario en el repositorio se inicia a partir de la validación aportada por este módulo, que utiliza el componente "*Login*" del núcleo.

5.4.1. SLOR IModule Interface

Cada módulo expone una interfaz común llamada "*IModule*". Esta interfaz aporta las operaciones necesarias de administración del componente; en principio para el prototipo se han definido las operaciones "*install*" y "*unInstall*". La operación

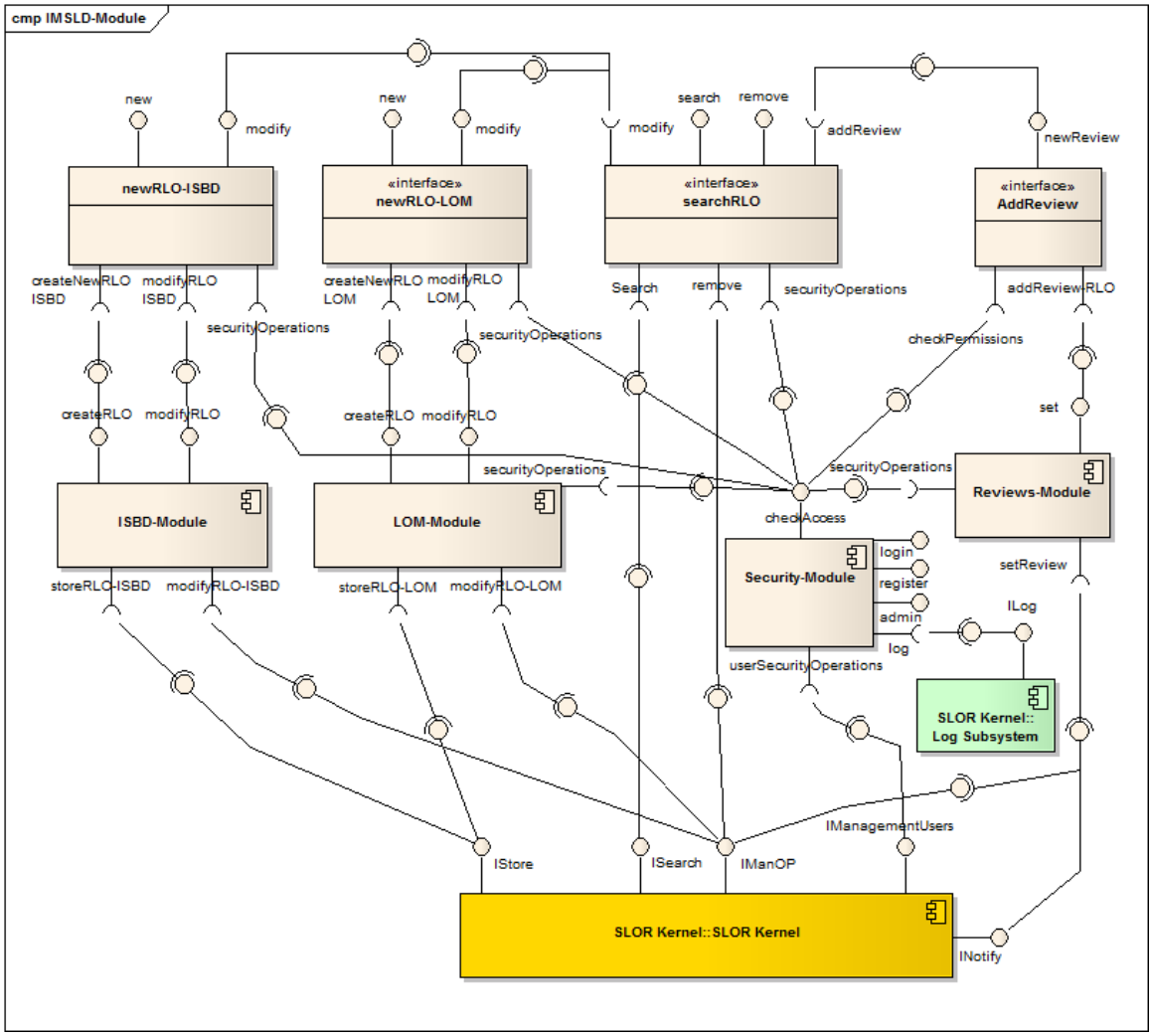

Figura 5.15: Diagrama de componentes de los módulos LOM, ISBD y seguridad.

"*install*" utiliza un fichero XML denominado descriptor del módulo, que indica las dependencias del módulo con otros módulos de SLOR. Al instalar un nuevo módulo en el sistema se copian el conjunto de interfaces de usuario y se despliegan las clases, posteriormente se recarga la aplicación web. Por ejemplo, el módulo "*newRLO-ISBD*" representado en la figura 5.15 depende del módulo "*Security-Module*", por lo que en el proceso de instalación del módulo se verifica que previamente esté cargado el módulo Security-Module.

La operación "*unInstall*" elimina el módulo del repositorio, quitando las interfaces de usuario y clases de programación de las que se compone.

5.4.2. Interacción Módulo-Interfaz

Los módulos exponen una serie de operaciones invocadas por los eventos lanzados desde las interfaces de usuario. Cada interfaz posee la propiedad de lanzar eventos como "*action*", cuando se hace clic sobre un botón, o "*valueChangeListener*", cuando cambia el valor de una lista desplegable. Estos eventos invocan las operaciones correspondientes implementadas en un módulo, desencadenando una acción de ida y vuelta[10]. Otra parte de la funcionalidad es invocada por eventos lanzados desde el componente Intellisense SLOR-AJAX (descrito más adelante en la sección 5.5) lo que permite evitar una recarga de página por cada acción a ejecutar.

Al crear una instancia de un módulo es necesario especificar el tipo de operaciones a realizar. Estas pueden ser operaciones de ejecución de una instalación inicial ("*INSTALL*", valor enumerado de tipo "*behaviourType*"), desinstalación ("*UNINSTALL*") o de operación normal ("*OPERATIONAL*").

Los módulos de tipo gestión de registros de metadatos poseen un ciclo de vida asociado a la sesión del cliente, por lo que cuando un cliente solicita una interfaz de usuario de inserción de un registro de metadatos, el código del servidor incluye dentro de la sesión del usuario una nueva instancia operacional del módulo de gestión correspondiente. Esto es debido a la gestión realizada en memoria de la construcción de los registros de metadatos, que ha de ser individual y separada en cada inserción

[10]En inglés - PostBack

realizada por un usuario. Cuando acaba el proceso de inserción, la instancia asociada al módulo desaparece. En cambio los módulos encargados de realizar operaciones comunes dentro del conjunto de registros de metadatos del repositorio, se cargan al inicio de la aplicación web y permanecen latentes en memoria en el ámbito de aplicación hasta que la aplicación se descarga del servidor.

Para entender la creación y ciclo de ejecución de un módulo, a continuación se describe la estructura e implementación de las funciones más relevantes del módulo "*LOM-Module*".

LOM-Module

En general, la implementación de las funciones del módulo requiere de los componentes descritos del núcleo y del sistema de persistencia. En el caso de un módulo LOM de tipo gestión de registros de metadatos, se han implementado las operaciones genéricas de inserción y modificación de registros descritas en la interfaz "*IMetadataRecordManagementOperations*", que son "*createRLO*" para la creación de registros de metadatos de objetos de aprendizaje y "*modifyRLO*" para su modificación.

Para poder realizar la construcción de un registro de metadatos, es necesario gestionar en memoria el objeto que contiene toda la estructura del registro. Por ello en el módulo se han definido diversas colecciones que contienen elementos de tipo "*descriptivePropertive*", como es el caso de las colecciones "*description*", "*keywords*", "*coverage*" o "*classification*", entre otras. Este grupo de colecciones de tipo "*java.util.Hashtable*" permiten mantener un modelo dinámico de gestión en memoria de toda la estructura final del registro LOM, con las expresiones

semánticas añadidas, antes de ser almacenado en el sistema persistente (ver figura 5.14).

Al solicitar la interfaz de usuario asociada al módulo que permite crear un nuevo registro de metadatos de tipo "*LOM_Record*", se crea una nueva instancia operacional del módulo y se inserta dentro del ámbito de la sesión del usuario. El constructor del módulo se muestra en el siguiente listado de código (5.20):

Listado 5.20: Módulo LOM

```
1  public class LOM_Module implements IModule,
       IMetadataRecordManagementOperations{
2       slor.ontology.ontoSlorFactory sf;
3       String uri = slor.kernel.SlorParameters.URI;
4       OWLModel owlMemoryModel;
5       slor.persistence.PersistentSubsystem pm;
6       slor.persistence.oc.HCIQueryComponent oc;
7       slor.kernel.IKernel kernel;
8       // LOM Descriptor properties
9       String LORID,textID,title;
10      String [] languages;
11      //..
12      //  LOM multiple properties - single strings & semantic
           expressions
13      private java.util.Hashtable description;
14      private java.util.Hashtable keywords;
15      //..
16      private java.util.Hashtable classification;
17      //..
18
19      public LOM_Module(slor.kernel.IKernel _k,behaviourType behaviour)
           {
20        switch(behaviour){
21            case INSTALL:  install();
22                   break;
23            case UNINSTALL: unInstall();
24                   break;
```

```
25            case OPERATIONAL:
26               descriptions=new java.util.Hashtable();
27               keywords=new java.util.Hashtable();
28               coverage=new java.util.Hashtable();
29               //.. All multiple properties
30               classification=new java.util.Hashtable();
31               try {
32                  kernel=_k;
33                  pm=kernel.getIMapOP().getPersistentSubsystem();
34                  oc=pm.getHCIQueryComponent();
35                  owlMemoryModel= ProtegeOWL.createJenaOWLModelFromURI(
                       uri);
36                  sf=new slor.ontology.ontoSlorFactory(owlModel);
37               } catch (Exception ex) {
38                  ex.printStackTrace();
39               }
40         }
41      }
42   }
```

En las líneas iniciales del listado 5.20 se encuentra la definición de la clase y las definiciones de las propiedades del módulo, primero las correspondientes a las referencias de los objetos de gestión del núcleo "*kernel*" y del sistema de persistencia, entre los que se requiere la referencia a la interfaz del modelo en memoria "*owlMemoryModel*", la referencia a la interfaz del sistema de gestión de persistencia "*pm*", la referencia al puente de operaciones HCI de OpenCyc "*oc*" y finalmente la referencia a la factoría de construcción del modelo "*sf*". Seguidamente se declaran todas las estructuras de tipo HashMap gestionadas por el módulo que almacenan los componentes del registro de metadatos "*LOM_Record*".

El constructor del módulo recibe dos parámetros, la referencia a la interfaz del kernel de SLOR (" *_k* ") y el tipo de comportamiento de la instancia del módulo

("*behaviour*"). Si el comportamiento es de tipo "*INSTALL*", el módulo inicia el proceso de instalación sobre el repositorio invocando al método "*install*"; si es de tipo "*UNINSTALL*", el módulo inicia el proceso de desinstalación del repositorio invocando al método "*unInstall*". En cambio, si el comportamiento es de tipo "*OPERATIONAL*", se inicializan los valores de las referencias enunciadas anteriormente.

Entre las líneas 22 y 26 se inicializan los elementos "*HashTable*", seguidamente se obtiene la referencia al subsistema de persistencia y al puente de operaciones HCI de OpenCyc, y finalmente en la línea 31 se crea una nueva instancia de un modelo OWL según el esquema de la ontología de SLOR (especificado en el parámetro uri - "*slor.kernel.SlorParameters.URI*"). Sobre este modelo, en la línea 32 se construye la factoría de creación de objetos.

Como hemos visto en el código anterior, en la construcción de una instancia operacional del módulo "*LOM_Record*" se crea un modelo OWL en memoria que albergará de forma temporal un conjunto de objetos RDF/OWL que almacenan el contenido de los campos del registro de metadatos. La construcción se realiza de forma incremental, el módulo aporta las operaciones invocadas por los eventos de la interfaz de tal forma que en cada campo LOM representado por una propiedad múltiple en el modelo OWL, se inserta, borra o modifica una cadena simple o expresión semántica. Las propiedades múltiples "*description*", "*keywords*", "*coverage*", ... y "*classification*" van completándose con las expresiones introducidas en la interfaz de usuario.

En el listado de código 5.21 se muestra el conjunto de métodos que permiten operar con los valores asignados a las propiedades en el modelo de memoria. El mecanismo es similar al modelo desconectado propuesto por la tecnología ADO.NET, el

usuario opera con el registro en memoria, añadiendo, quitando o modificando las propiedades insertadas. Cuando el registro queda listo para la inserción se invoca al método "*createRLO*". A continuación se explica la estructura de los métodos que permiten operar con los valores asignados a una propiedad, en el ejemplo expuesto se explican los métodos que permiten insertar una nueva palabra clave simple - método "*addSimpleStringKeyword*" o expresión semántica - método "*addSemanticKeyword*", o eliminarla - método "*removeKeyword*".

El método "*addSemanticKeyword*" permite añadir un enlace semántico a un término (existente en una ontología definida en OWL o en la base de conocimiento OpenCyc), a la propiedad "*keyword*" de un registro de metadatos del objeto de aprendizaje "*LOID*", construido en la interfaz web y almacenado en un modelo OWL en memoria que utiliza el mismo esquema que el persistente de SLOR (consultar la operación almacenar descrita en la sección 5.3.2).

El método recibe como parámetros el identificador de la ontología de la cual se ha obtenido el término elegido "*ontology*", el identificador del objeto de aprendizaje "*LOID*" sobre el que se ha de anexar el enlace del concepto sobre la propiedad "*keywords*" y finalmente el término elegido a enlazar "*IDConcept*". En el método se verifica si se ha de tratar el término recibido como un término de la base de conocimiento OpenCyc, o si ha de procesar el término como un elemento descrito en alguno de los esquemas OWL gestionados por el repositorio. En la línea 18 se realiza la comprobación: si es un término OpenCyc se anexa al esquema en memoria construyendo la expresión semántica utilizando la clase "*OCSemanticLinkBuilder*" e incluyéndola en el HashTable "*keywords*", y si no se construye la expresión de enlace a un concepto relacionado con una ontología descrita en OWL gestionada por el repositorio, utilizando la clase "*OWLSemanticBuilder*", y finalmente también se

incluye la expresión en el Hashtable "*keywords*".

Listado 5.21: Gestión de la propiedad múltiple "*keywords*" del módulo "*LOM-Module*"

```
1  import slor.kernel.OCSemanticLinkBuilder;
2  import slor.kernel.OWLSemanticLinkBuilder;
3  //..
4
5  class LOM_Module implements IModule,
       IMetadataRecordManagementOperations{
6  //..
7      public void addSimpleStringKeyword(String LORID,String language,
           String keyword){
8          slor.ontology.simpleString ss=
9                     sf.createsimpleString("sk_"+ LORID + "_" + java.util
                          .Calendar.getInstance().getTimeInMillis());
10         ss.setLanguage(language);
11         slor.ontology.impl.DefaultsimpleString dss=
12                           new slor.ontology.impl.DefaultsimpleString();
13         ss.setValue(keyword);
14         keywords.put(keyword,ss);
15     }
16
17     public void addSemanticKeyword(String ontology, String LORID,
           String IDConcept)
18     {
19
20         if (ontology.equals(oc.ID_ONTOLOGY) && oc.isConcept(IDConcept))
               {
21             //OpenCyc Concept oc_ConceptLink.. <--
22             OCSemanticLinkBuilder ocsl=new OCSemanticLinkBuilder(sf);
23             slor.ontology.oc_ConceptLink koccl=
24                 ocsl.createOC_ConceptLink("kcl_"+ LORID,IDConcept);
25             keywords.put(koccl.getConcept(),koccl);
26         }else{
27             // OWL Concept Link
28             OWLSemanticLinkBuilder owlsl=new OWLSemanticLinkBuilder(sf)
                   ;
29             slor.ontology.owl_ConceptLink kowlcl=
```

```
30                owlsl.createOWL_ConceptLink("kcl_"+ LORID,IDConcept);
31            keywords.put(kowlcl.getConcept(),kowlcl);
32        }
33    }
34
35    public void removeKeyword(String ID){
36        keywords.remove(ID);
37    }
38//..
39}
```

Una vez rellenados todos los campos del registro de metadatos, el usuario puede desencadenar la inserción persistente del registro dentro del repositorio. La operación de inserción del registro de metadatos de tipo "*LearningObject_LOM*" se implementa en el método "*createRLO*" definido en la interfaz de administración de registros "*IMetadataRecordManagementOperations*". La secuencia de operaciones hace persistente el registro de metadatos dentro del modelo.

La inserción se completa en tres fases diferentes (ver listado 5.22):

- En la primera fase se verifican los permisos del usuario que ha lanzado la operación. Si el usuario no puede ejecutar la operación, el método lanza la excepción "*OperationAccessException*".

- En la segunda fase se crean, completan y asocian todos los componentes del registro de metadatos del objeto de aprendizaje. En el caso del módulo "*LOM-Module*", en la línea 7 se crea el descriptor del objeto de aprendizaje en el modelo "*owlMemoryModel*" a través de la referencia "*sf*" de la factoría, seguidamente en la línea 9 se crea el registro de metadatos "*lr*" de tipo "*LOM_Record*". Posteriormente en las líneas 11 y 12 se asocian los valores correpondientes a las propiedades del descriptor del objeto de aprendizaje "*lor*", a continuación desde la línea 14 hasta la 36 se incluyen todas las propiedades asociadas al registro

de metadatos albergadas en las tablas "*HashTable*" gestionadas por el módulo. Finalmente en la línea 37 se asocia el registro de metadatos al descriptor del objeto de aprendizaje.

- En la tercera fase se hace persistente el modelo OWL gestionado y almacenado de forma temporal en memoria por el módulo. Para realizar esta acción, en la línea 38 se invoca al método "*insertMemoryModel*" proporcionado por la interfaz "*IStore*" del núcleo.

Listado 5.22: Método createRLO

```
1  public void createRLO(User refUser) throws OperationAccessException{
2
3     if (securityModule.checkAccess(slor.modules.Operations.CREATELOR,
4                                    refUser)){
5
6             slor.ontology.LearningObject_LOM lor=
7                    sf.createLearningObject_LOM("lom_"+ LORID);
8             slor.ontology.LOM_Record lr=
9                    sf.createLOM_Record("lor_" + LORID);
10
11            lor.setIdentifier(textID);
12            lor.setTitle(title);
13            // Languages..
14            for (int i=0;i<languages.length;i++){
15                lr.addLanguages(languages[i].getValue().toString());
16            }
17
18            slor.ontology.descriptiveProperty dp;
19            java.util.Enumeration e=descriptions.elements();
20            while (e.hasMoreElements()){
21               dp=(slor.ontology.descriptiveProperty)e.nextElement();
22               lr.addDescription(dp);
23            }
24
25            e=keywords.elements();
26            while (e.hasMoreElements()){
27               dp=(slor.ontology.descriptiveProperty)e.nextElement();
```

```
28              lr.addKeywords(dp);
29           }
30
31           e=coverage.elements();
32           while (e.hasMoreElements()){
33              dp=(slor.ontology.descriptiveProperty)e.nextElement();
34              lr.addCoverage(dp);
35           }
36           //..
37           lor.setHasAssociatedMetadataRecord(lr);
38           kernel.getIStore().insertMemoryModel(owlMemoryModel);
39    }else throw(new(OperationAccessException());
40 }
```

A lo largo de esta sección se ha explicado la estructura general de un módulo, exponiendo como ejemplo las funciones más relevantes aportadas por el módulo "*LOM_Module*" de gestión de registros de metadatos LOM. El esquema de implementación es similar para la gestión de todos los tipos de registros de metadatos.

5.5. Interfaces

SLOR posee un diseño debilmente acoplado, donde las operaciones que permiten interactuar con el repositorio se invocan en los eventos de las interfaces de usuario. Las interfaces pueden ir orientadas a usuarios humanos o agentes software.

Desde el punto de vista de las interfaces orientadas a usuarios humanos, el diseño de SLOR ha tenido en cuenta las características necesarias para poder ofrecer los componentes visuales que permitan un buen nivel de interacción hombre-máquina. Desde el primer momento, se han encontrado problemas en la usabilidad del sistema. Además, el gran número de campos de las especificaciones (LOM, IMS-CP, etc..) sumado a la dificultad existente en la creación de expresiones semánticas, hacen del uso continuo de este repositorio por parte de los creadores de metadatos, una tediosa tarea.

La tecnología "*Java Server Faces*" ofrece una filosofía de creación de Interfaces Web basadas en componentes visuales que pueden lanzar eventos, similar al modo tradicional de la creación de aplicaciones de escritorio. Gracias a la introducción de esta tecnología, pueden implementarse Interfaces Web del repositorio de una manera más sencilla que la especificada en otro tipo de tecnologías (como Struts), permitiendo añadir de forma cómoda para el programador un mayor grado de funcionalidad gracias al modelo de gestión de eventos mediante acciones de ida y vuelta. Utilizando esta tecnología se han creado para SLOR interfaces usables que utilizan la funcionalidad implementada en los módulos.

Aún con las ventajas ofrecidas por este tipo de tecnología en el proceso de desarrollo de una interfaz web, se han encontrado problemas a la hora de trabajar con expresiones semánticas. Para crear enlaces a conceptos de otras ontologías, se requiere conocer bien el esquema y los términos definidos. De cara a un humano, este tipo de operaciones son muy complicadas ya que requiere conocer muy bien (o haber consultado previamente) los términos, propiedades y predicados utilizados para construir una expresión.

Otro tipo de interfaces son las que no poseen ningún tipo de componente visual, y únicamente exponen operaciones que pueden ser invocadas de forma remota por algún otro repositorio de objetos de aprendizaje o agente software. En SLOR se ha optado por utilizar las tecnologías ofrecidas por los Servicios Web, debido al amplio uso y aceptación dentro de la industria de desarrollo de componentes distribuidos. Los protocolos ofrecidos permiten una comunicación flexible sin barreras de arquitecturas

distribuidas complejas de comunicación[11]. Por ello se ha implementado un Servicio Web que da soporte a las operaciones básicas descritas en la especificación IMS-DRI.

A lo largo de las próximas secciones se explican en detalle los diferentes tipos de interfaces implementados en el repositorio, tanto Interfaces Web como Servicios Web.

5.5.1. Interfaces Web

En esta sección se pretende mostrar los dos tipos de interfaces relevantes utilizados en un repositorio semántico de objetos de aprendizaje:

- *Inserción de un registro de metadatos*: este tipo de interfaz contiene los componentes necesarios para completar los diferentes campos de un registro de metadatos. Posee una estructura similar a la interfaz de inserción de registros del módulo LOM (ver figura 4.16, en la sección 4.3.1 del capítulo anterior). Están preparadas a nivel general para la inserción de expresiones semánticas manejadas por el modelo de SLOR.

- *Búsqueda semántica*: este tipo de interfaz proporciona los componentes visuales que permiten construir expresiones semánticas de búsqueda.

Componentes visuales utilizados para la inserción de un registro de metadatos

Respecto a la inserción de un registro de metadatos, se han tenido en cuenta en el diseño de los componentes visuales dos tipos diferentes de propiedades, las que contienen un único valor asociado de un tipo concreto (cadena, entero, real, etc...) y las que contienen una colección posible de expresiones de descripción (correspondientes a la

[11]referidas a la configuración de firewalls para que permitan pasar tramas especificas de un protocolo como IIOP o MSRPC, y a la ejecución aplicaciones en un servidor que aporten servicios de tipo contenedor de aplicaciones distribudas DCOM+ Server, Corba, entre otros.

clase "*descriptiveProperty*") del modelo de SLOR. Repecto a este último tipo de propiedad, se ha propuesto el diseño de un componente que ayude a introducir y generar los distintos tipos de descripción de propiedades. En la figura 5.16 se representa el componente visual que permite realizar la inserción de una propiedad múltiple de rango "*descriptiveProperty*" dentro de un registro de metadatos.

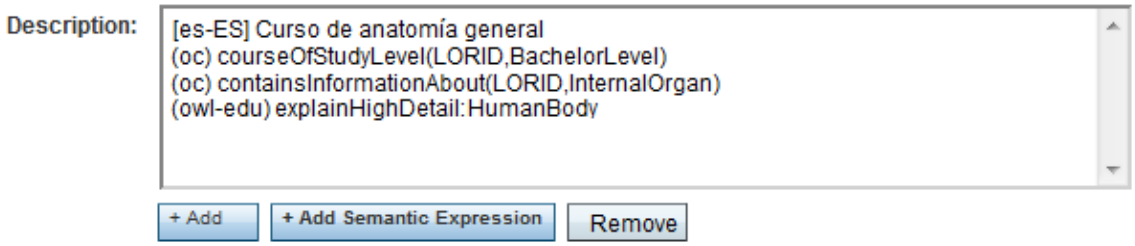

Figura 5.16: Componente visual de inserción de propiedades múltiples.

El componente visual de propiedades múltiples se compone de los siguientes elementos:

- La lista donde se muestran todas las expresiones insertadas. Cada expresión posee una forma de serialización. Así en el caso de las propiedades de tipo "*simpleString*", se representan con el valor del lenguaje asociado entre corchetes y el valor de la propiedad a continuación. Para las expresiones que enlazan con otras ontologías se siguen las reglas siguientes:
  - En el caso de enlaces a ontologías descritas en OWL existen dos modos de representación: únicamente para términos con la sintaxis de inclusión "*(ID-ontología) término*" y aserción de una propiedad OWL con la sintaxis "*(ID-ontología) propiedad: término*".
  - Para OpenCyc también hay dos posibles modos de representación: en el caso de enlazar únicamente términos se expresarían "*(oc) término*", mientras que en el caso de incluir alguna aserción con un predicado se utiliza

la sintaxis"*(oc) predicado(arg1,arg2,...,argn)*".

- El botón "*+Add*" muestra el formulario representado en la figura 5.17 para insertar un valor de tipo "*simpleString*" asociado del componente.
- El botón "*+AddSemanticExpresion*" muestra el formulario representado en la figura 5.18 para insertar una expresión semántica utilizando el componente Intellisense SLOR-AJAX.
- El botón "*Remove*" elimina el elemento seleccionado de la lista, también invoca al método que elimina el objeto RDF/OWL asociado en el modelo OWL almacenado en memoria y gestionado por el módulo correspondiente.

En la figura 5.17 se muestra el formulario de entrada de las propiedades de tipo "*simpleString*" descritas en el modelo SLOR.

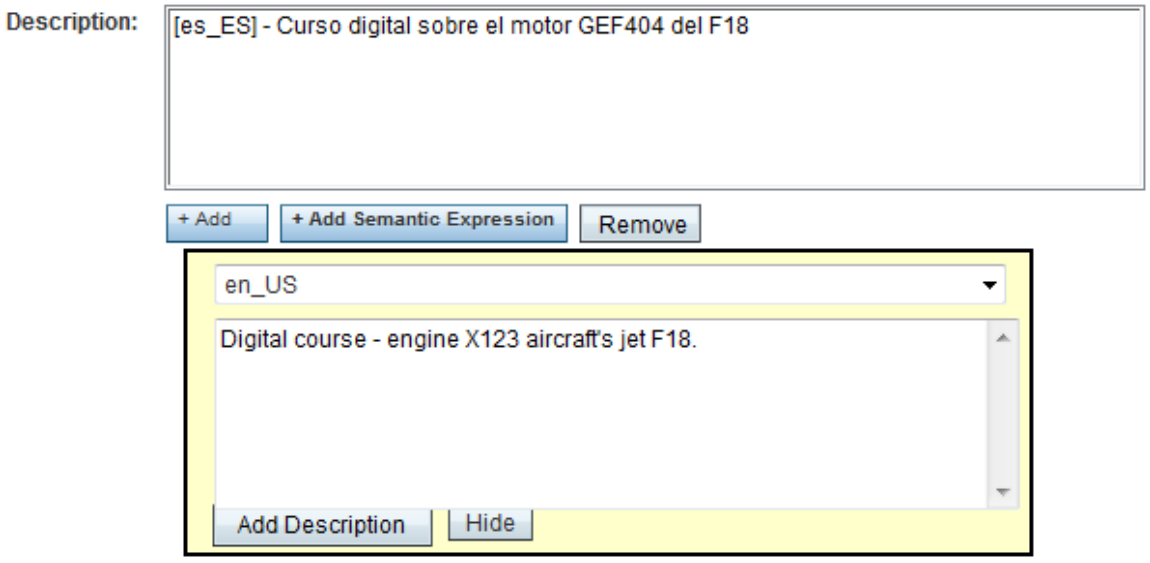

Figura 5.17: Componente visual de inserción de valores "*simpleString*".

El listado 5.23 muestra el código de evento "*action*" del botón "*AddDescription*". Como puede apreciarse se recoge el valor escrito por el usuario en la caja de texto

(nombre del objeto "*txtNewDescription*") y el lenguaje seleccionado en la lista ("*ddlActiveDescriptionLanguage*"). Al final, se invoca al método del módulo que permite añadir un valor de tipo "*simpleString*" a la propiedad "*description*".

Como ya se ha comentado en las secciones anteriores de este capítulo, la inserción se realiza en el modelo del registro de tipo "*LOM_Record*" en la tabla hash "*descriptions*". Si el objeto se desea eliminar, únicamente se ha de seleccionar de la lista múltiple de descripciones y pulsar el botón "*Remove*". Este botón invoca a la operación "*remove[Propiedad]*" del módulo asociado a la interfaz de creación de registros de metadatos.

Listado 5.23: Evento AddDescription_action

```
1 public String btnAddDescription_action() {
2     SessionBean1 sb1=getSessionBean1();
3     slor.modules.LOM_Module lom=sb1.getLOM_Module();
4     String selectedLanguage=(String) this.ddlActiveDescriptionLanguage
          .getSelected();
5     String text=this.txtNewDescription.getText().toString();
6     lomModule.addSimpleStringDescription(sb1.getLORID(),
          selectedLanguage,text);
7 }
```

Para añadir una expresión semántica en el formulario representado en la figura 5.18 se ha de pulsar el botón "*Add Semantic Expression*". El usuario se apoya en el control "*Intellisense SLOR-AJAX*" para construir los distintos tipos de expresiones. El código ejecutado mostrado en el listado 5.24 corresponde al evento "*action*" asociado al botón "*AddSemanticDescription*". Existen dos tipos de expresiones: referencias a conceptos únicos y aserciones de predicados o valores de propiedades.

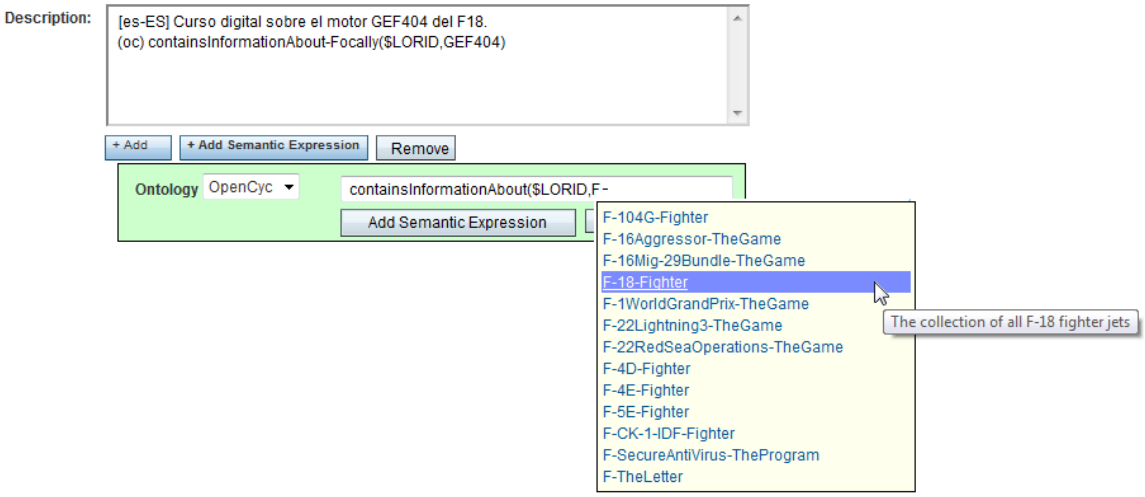

Figura 5.18: Componente visual de inserción de expresiones semánticas

Listado 5.24: Método btnAddSemanticDescription_action

```
1  public String btnAddSemanticDescription_action() {
2      SessionBean1 sb1=getSessionBean1();
3      slor.modules.LOM_Module lomModule=sb1.getLOM_Module();
4      String expression=this.autoCompleteDescription.getText().toString
           ();
5      String ontology=this.ddlSemanticDescription().getSelected();
6      if (ontology.equals(lomModule.getocID())){
7          String predicate=slor.util.Predicates.getPredicate(expression);
8          String arguments=slor.util.Predicates.getArguments(expression);
9          lomModule.addSemanticDescription(ontology,sb1.getLORID(),
               predicate,arguments);
10     }else{
11         String property=slor.util.Property.getProperty(expression);
12         String property_value=slor.util.Property.getValue(expression);
13         lomModule.addSemanticDescription(ontology,sb1.getLORID(),
               property,property_value);
14     }
15 }
```

Intellisense SLOR-AJAX

El componente "*Intellisense SLOR-AJAX*" ha sido construido a partir del componente "*Auto Complete Text Field*" de la biblioteca de componentes visuales "*Blue Prints AJAX Components*"[12]. Ha sido necesario incorporar las siguientes modificaciones:

- Indicador de ejecución de operación: debido al tiempo de espera existente en los procesos de los esquemas de ontologías y la base de conocimiento OpenCyc ha sido necesario incorporar una imagen indicadora de la actividad realizada del servidor.
- Control de la posición de la lista de resultados: cuando se crea una expresión semántica como una aserción utilizando un predicado, es necesario colocar sobre el argumento que se esta escribiendo la lista de los términos que ayudan a autocompletar la expresión de forma correcta.
- Incluir la propiedad "*tooltip*": necesaria para aportar una explicación del término o estructura de un predicado o propiedad.

Existen cuatro formas de operar con este tipo de control en función del tipo de expresión que se esté tratando. El proceso de obtener las listas de valores adecuados para autocompletar una expresión se realiza en el servidor, en un método con la forma:

```
1 public void autoCompleteAJAXWebControl_complete(FacesContext fc,
    String s, CompletionResult cr);
```

Este método se ejecuta cada vez que se escribe una letra sobre la expresión, de tal forma que siempre se va actualizando de forma dinámica la interfaz a través de la

[12]Componentes AJAX BluePrints - https://blueprints.dev.java.net/

tecnología JSON[13] que permite la invocación de llamadas a la operación del servidor desde un cliente Web javascript.

En función de la ontología seleccionada en la lista desplegable del control de inserción de expresiones semánticas, el código del evento "*complete*" verifica las operaciones que ha de realizar: procesar un término o predicado OpenCyc, o procesar las clases, individuales o propiedades de una ontología OWL gestionada por el repositorio.

En el listado de código 5.25 se muestra el conjunto de instrucciones que permite obtener un conjunto de clases o colecciones de la base de conocimiento OpenCyc o de una ontología OWL registrada en el repositorio en función del texto escrito en el control "*autoCompleteKeywords*":

Listado 5.25: AutoCompletar AJAX

```
1  public void autoCompleteAJAXWebControl_complete(FacesContext fc,
       String s, CompletionResult cr)
2  {
3      String ontology=this.ddlSemanticKeyword().getSelected();
4      IKernel kernel=getSLORApplicationBean().getKernel();
5      ISearch query=kernel.getISearch();
6      String[] z;
7      if (s.length()>1){
8          if (ontology.equals(lomModule.getocID())){
9              z=query.getOpencycCollections(s,20,true);
10         }else{
11             z=query.getOWLClasses(s,20,true);
12         }
13         cr.addItems(z);
14     }
15 }
```

[13] http://json.org/

En cambio si en el campo del registro de metadatos se pueden insertar expresiones de tipo predicado ("*OpenCyc_Predicate*") o propiedad ("*Owl_Property*"), el método ha de procesar la cadena recibida en el parámetro "*s*". En el caso de un predicado OpenCyc el análisis de la cadena recibida ha de descomponer el nombre del predicado y los argumentos de los que se compone.

```
1 predicado(arg1,arg2,...,argn)
```

En el servidor se van recibiendo las invocaciones al evento "*complete*" lanzadas por el componente AJAX. Según se va escribiendo, para cada argumento se invoca al método "*getOpenCycInstancesOf*" que procesa el tipo ("*isa*") e intenta recuperar todas las instancias.

El análisis de una propiedad OWL únicamente requiere de conocer el rango de la misma para obtener el conjunto de resultados. La obtención de la lista de propiedades de una ontología determinada se consigue invocando al método "*getOWLProperties*" de la interfaz "*ISearch*" del núcleo. Una vez elegida la propiedad, se rellena la lista de instancias que corresponde con el rango de la propiedad, facilitando de esta forma el proceso de completar una expresión.

Componentes visuales utilizados para la búsqueda de registros de metadatos

En la figura 5.19 se muestra el componente visual utilizado como herramienta de construcción de expresiones para búsquedas semánticas. En la imagen pueden distinguirse tres tipos de zonas diferentes. La primera, correspondiente a la parte superior, que contiene los componentes necesarios para navegar entre la estructura del esquema flexible del repositorio y construir las restricciones semánticas que desean aplicarse sobre la búsqueda a realizar. La segunda zona corresponde a la tabla resumen de

las expresiones introducidas por el usuario, posee el botón "*execute*" que permite lanzar la consulta al núcleo de SLOR. Finalmente la tabla situada en la parte inferior contiene los resultados obtenidos.

El proceso de construcción de una expresión semántica sigue los siguientes pasos:

- Primero se selecciona el nivel de clase sobre el cual quiere consultarse. Sobre ese nivel puede consultarse algún valor de propiedad o registro de metadatos gestionado por el repositorio.

- Seguidamente se selecciona el valor de alguna propiedad de un registro de metadatos, como puede ser "*LOM-Record.keywords*" o "*ISBD.Area*".

- Una vez seleccionada la propiedad, si es de tipo "*descriptiveProperty*" aparecen los distintos tipos en el control "*Tree*" de la parte izquierda superior de la pantalla, para seleccionar el tipo de enlace sobre el que se desea consultar. El usuario selecciona un tipo de propiedad y establece los valores asociados en la parte inferior.

- Sobre esos valores, para la ayuda de búsqueda dentro del esquema de una ontología, se ha establecido el control visual "*Value*" como un control de tipo "*Intellisense SLOR-AJAX*". En el caso de seleccionar una propiedad de tipo "*Owl Link*" aparece una lista desplegable con todas las ontologías gestionadas por el repositorio, de esta forma el control "*Intellisense SLOR-AJAX*" utiliza como parámetro la ontología seleccionada por el usuario para completar la lista de términos o propiedades mostrada. En el caso de seleccionar una propiedad de tipo "*OpenCyc Link*", se muestra únicamente el control "*Value*" que adquiere el comportamiento de completar la lista de apoyo con términos o predicados OpenCyc.

- Una vez escrito el valor y seleccionada la ontología sobre la que se desea realizar una búsqueda, se pueden establecer los parámetros de inferencia. En la figura 5.19 se muestra un ejemplo de restricción con inferencia sobre la propiedad "*Area*" de un registro de metadatos tipo "*ISBD*" enlazado a un concepto OpenCyc ("*oc_ConceptLink*"), con el valor asignado al término OpenCyc "*EngineeringField*". Como puede apreciarse en la figura, se establece la casilla "*Inference*" y se selecciona el tipo de inferencia (en el caso del ejemplo se ha seleccionado el tipo "*TRANSITIVE_BOTTOMUP*"). Si el proceso de inferencia requiere de algún parámetro aparecerán los controles en la parte inferior, como es el caso de la profundidad en el proceso de búsqueda transitiva "*deep*".

- Al finalizar la configuración de la restricción se pulsa el botón "*Add restriction*" y se incluye en la tabla "*Query Restrictions*" situada en la segunda zona de la interfaz.

- Una vez creadas todas las restricciones el usuario pulsa el botón "*execute*" que lanza la consulta al núcleo de SLOR por medio de la interfaz "*ISearch*".

- Al terminar de procesar la consulta, el servidor devuelve la página con los resultados incluidos en tabla "*Results*".

5.5.2. Comunicación distribuida

SLOR es un servicio con capacidad de procesar y recibir operaciones de un usuario, pero también de otros agentes externos como otros repositorios o herramientas de autor. Desde esta visión operacional, desde una red como internet SLOR puede proveer una serie de servicios a determinadas entidades externas que deseen interactuar para conseguir un objetivo definido. En este escenario LMSs externos como Moodle o Atutor, herramientas de autor u otros repositorios, pueden utilizar los servicios

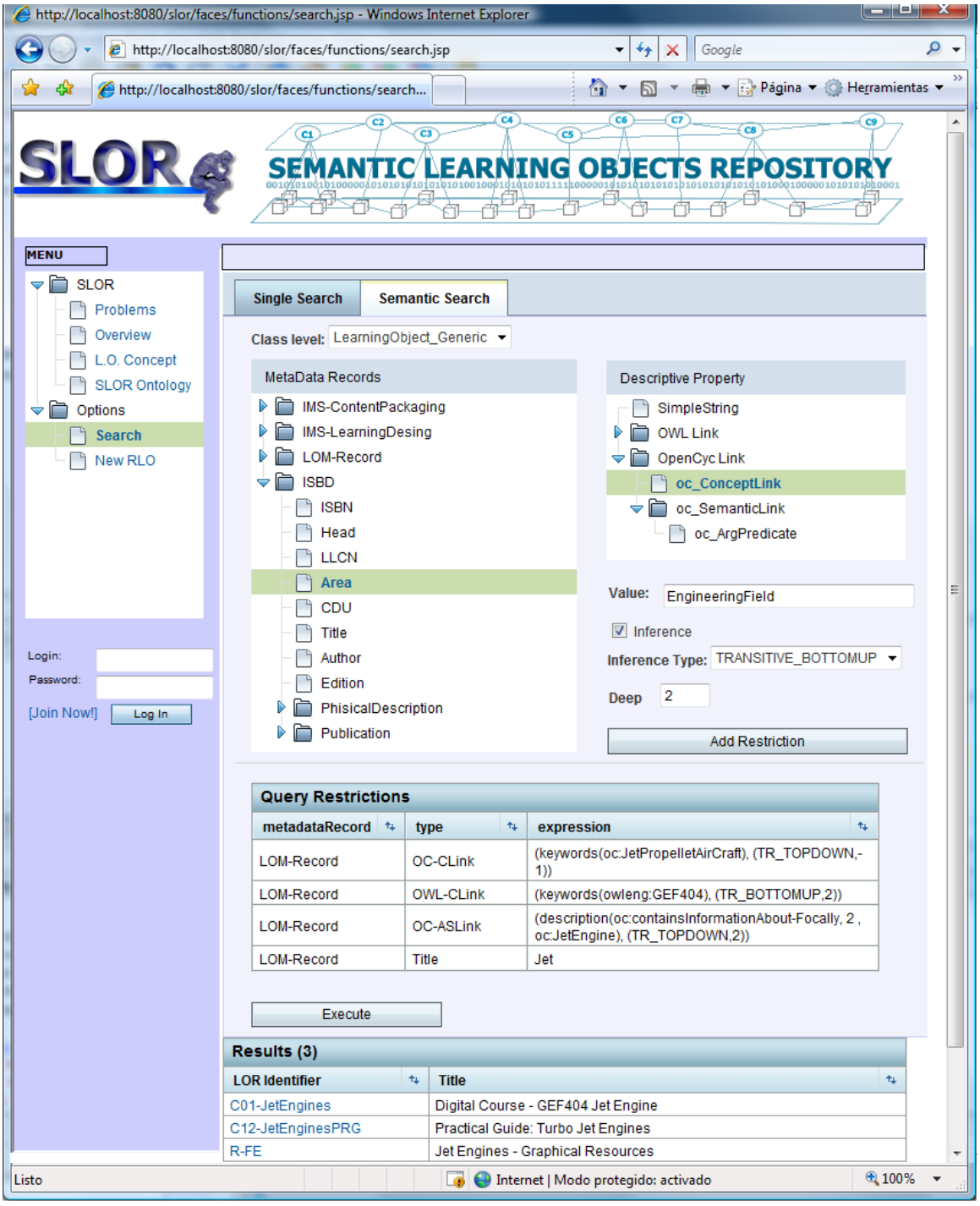

Figura 5.19: Componente visual de inserción de restricciones de búsqueda

expuestos por SLOR para consultar su contenido y obtener determinados materiales (referenciados por los registros de metadatos) que posteriormente pueden ser incluidos en algún curso gestionado por sistema de gestión de contenidos.

Para establecer el diseño de componentes de comunicación con otras aplicaciones, se ha utilizado la especificación IMS-DRI, las propuestas de implementación con el protocolo SOAP ofrecidas por la guía de buenas prácticas [IMS03a] y las ideas aportadas por las especificaciones de los lenguajes XQuery y Z39.50 para el diseño del modo de operación sobre la consulta del contenido del repositorio.

En la figura 5.20 se presenta el diagrama de interacción entre la comunicación del repositorio con otras aplicaciones. Como puede apreciarse el componente SLOR posee 4 interfaces diferentes que concuerdan con los Servicios Web expuestos:

- "*Expose*": expone las operaciones de búsqueda, actualización y alerta sobre el estado del repositorio.
- "*Store*": incluye el conjunto de operaciones remotas necesarias para insertar un registro de metadatos de un objeto de aprendizaje dentro del modelo conceptual de SLOR.
- "*Manage*": aporta las operaciones que permiten administrar el contenido del repositorio de objetos de aprendizaje.
- "*Deliver*": incluye las funciones que entregan el contenido de un registro de metadatos almacenado en el repositorio.

Algunas aplicaciones externas necesitarán para interactuar con el repositorio SLOR algún traductor del modelo conceptual de SLOR al esquema de metadatos que utilice. Por ello se ha definido el componente "*Mediator*" como un traductor de los esquemas

manejados por los módulos instalados en SLOR, como puede ser el caso de traducción LOM-XSD → LOM OWL y LOM OWL → LOM XSD, si una herramienta no manipula el esquema OWL de metadatos "*LOM_Record*" utilizado por el repositorio.

A continuación se presentan los agentes software externos más representativos que pueden interoperar con el repositorio SLOR:

- Herramientas de autor: son las herramientas de construcción de los objetos de aprendizaje. Este tipo de herramientas interactúan con el repositorio en la labor de buscar objetos de aprendizaje para su posterior reutilización. También pueden almacenar nuevos registros de metadatos invocando la operación remota de almacenamiento "*Store*", o gestionar de forma remota el contenido del repositorio utilizando la interfaz "*Manage*". Estas herramientas pueden utilizar un esquema de metadatos que difiere del modelo conceptual gestionado por SLOR. El "*Mediador*" es la propuesta de componente software encargado de hacer la traducción del esquema manejado por la herramienta de autor a un esquema válido dentro del modelo conceptual de SLOR. Ejemplo: un archivo XML bajo el esquema XSD de LOM, el mediador traduce el código XML a código OWL bajo el esquema de metadatos "*LOM_Record*" incluido en el modelo conceptual de SLOR.

- Repositorios: pueden interactuar con SLOR para invocar operaciones de búsqueda de aquellos objetos de aprendizaje que no posean referenciados. Se pueden establecer mecanismos de búsqueda federada entre diferentes repositorios bajo una transacción compuesta 2PC (Two Phase Commit). El mecanismo de búsqueda sigue el mismo principio que para el anterior, el repositorio envía la consulta utilizando el servicio web "*expose*", una vez obtenidos los resultados el repositorio puede invocar la operación "*deliver*" si necesita recuperar algún

registro de metadatos en concreto.

- Sistemas gestores de contenidos de aprendizaje (LCMS): aplicaciones como Moodle o Atutor pueden acceder al repositorio para reutilizar los contenidos en múltiples cursos publicados y gestionados por este tipo de sistemas.

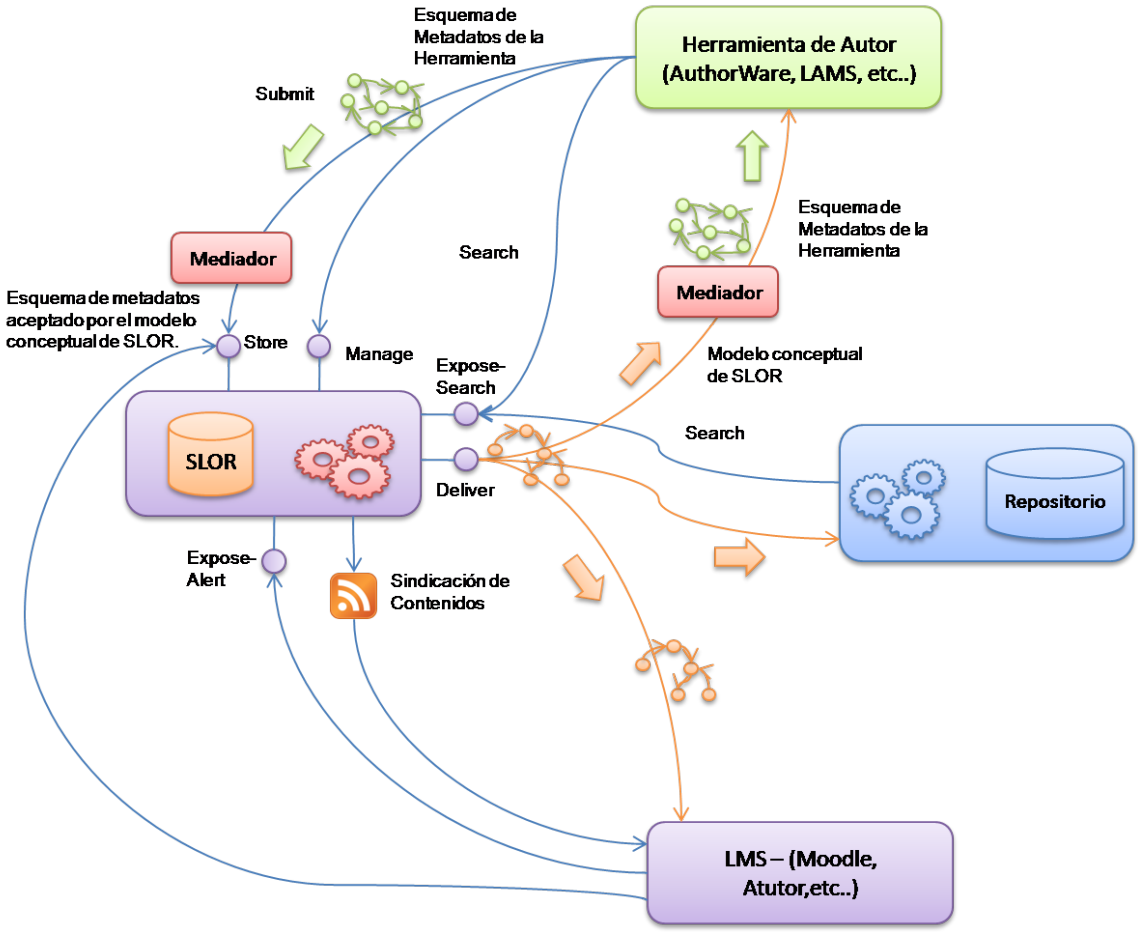

Figura 5.20: Interoperabilidad de SLOR

En el diseño del repositorio, se ha incluido como nuevo método de alertas complementario al propuesto por IMS DRI. Los usuarios o agentes externos pueden suscribirse a una RSS que contenga publicadas las nuevas inserciones y comentarios realizados

en el repositorio. De esta forma usuarios asiduos al contenido del mismo pueden configurar las categorías sobre las cuales quieren ser informados a través de la interfaz "*ALERT*" para poder recibir en formato RSS la información en su escritorio o sistema LCMS. Si se inserta un nuevo registro de metadatos y se establece la categoría (mediante un término de una ontología gestionada por el repositorio), se pública en la RSS la inserción realizada del nuevo registro. El proceso de obtención se realiza a través de la URL relativa:

```
/alert/service?user=usuario&ref=refnumber}
```

Donde el parámetro "*user*" es el nombre de usuario del repositorio que posee una lista de suscripción y "*refnumber*" el código de la lista de suscripción. Cada usuario puede configurar una lista RSS con una serie de categorías para que contenga únicamente las inserciones y comentarios realizados sobre un grupo de registros de metadatos de objetos de aprendizaje que pertenecen a las categorías definidas.

Desde el servicio "*Expose*" se pueden registrar alertas de comunicación desde otro tipo de sistemas como el correo electrónico. Si un usuario del repositorio desea recibir por correo electrónico mensajes de notificación de las actualizaciones realizadas, se puede generar un mensaje XML con una estructura basada en las propuestas de buenas prácticas de la guía IMS-DRI [IMS03b]. En el mensaje de registro de alerta se proponen tres tipos de estructuras:

- $< dr : event >$: donde se notifican las acciones a realizar por un evento. En la propuesta de alertas de la guía de buenas prácticas se especifica el uso de "*XQuery*" para lanzar una serie de consultas sobre el repositorio. En el modelo de SLOR se ha ampliado esta definición incluyendo una serie de acciones especificas de la clase "*oc:Action*":

- "*NewRLORegister*": para indicar la consulta sobre las operaciones de nueva inserción en el repositorio. Sobre esta clase de acción se pueden especificar las categorías que ha de analizar para la construcción del mensaje de notificación (propiedad "*category*"). Si se desea, se pueden especificar las categorías como términos OpenCyc o de otra ontología utilizando las clases "*oc_ConceptLink*" y "*owl_ConceptLink*" ya comentadas en la sección 5.3.1 del modelo de SLOR.
- "*NewCommentRegister*": indica la recogida de las acciones sobre la inserción de comentarios. Si se establece la propiedad "*onlyUsersRLOs*" con el valor "*true*", se indica al proceso de búsqueda que recoja los comentarios realizados únicamente sobre los objetos de aprendizaje insertados por el usuario "*user*".

- $< dr : frecuency >$: indica la frecuencia de ejecución del grupo de acciones especificado en $< dr : event >$.
- $< dr : notification >$: indica el mecanismo de notificación.

A continuación, en el siguiente listado de código, se muestra un ejemplo de registro de alertas para el aviso de nuevos objetos de aprendizaje que pertenezcan a la categoría "*oc:JetEngine*" y de aviso de nuevos comentarios sobre los objetos insertados por el usuario "*jesus.soto*". Cada 14 días se envía un email con la recogida de todas las acciones a la dirección "*jesus.soto@imai-software.com*".

```
<dr:alert id="urn:example.com:alerts:100201:110"
 xmlns:dr="http://imsproject.org/schema/dr/2001"
 xmlns:slor="http://slor.sourceforge.net/ontology/slor.owl/#">
 <dr:event>
       <slor:Actions rdf:ID="actions_trigger01">
          <has_Action>
              <NewRLORegister rdf:ID="alertnRLOr01">
```

```
                    <category>
                         <slor:oc_ConceptLink>
                            <concept rdf:datatype="&xsd;string">
                                JetEngine
                            </concept>
                         </slor:oc_ConceptLink>
                    </category>
                </NewRLORegister>
            </has_Action>
            <has_Action>
            <NewCommentRegister rdf:ID="alertcr01">
                <user rdf:datatype="&xsd;string">jesus.soto</user>
                <onlyUsersRLOs rdf:datatype="&xsd;boolean">true</
                    onlyUsersRLOs>
            </NewCommentRegister>
           </has_Action>
        </slor:Actions>
 </dr:event>

 <dr:frequency all-notify="false">
 <dr:period> P14D </dr:period>
 <dr:repeat> 50 </dr:repeat>
 </dr:frequency>

 <dr:notification>
 <dr:method> email </dr:method>
 <dr:method-data> jesus.soto@imai-software.com </dr:method-data>
 </dr:notification>
</dr:alert>
```

Como caso práctico se plantea el siguiente escenario de interacción: desde Moodle puede crearse un curso de Inteligencia Artificial que necesite alguna lección de Estadística, con ayuda de un "***plugin***" implementado para acceso a repositorios por SOAP IMS-DRI podría invocarse a la operación remota de búsqueda del servicio web "***Expose***" de SLOR y obtener los resultados deseados, en caso de solicitar un registro completo de metadatos desde una operación del plugin se invocaría al servicio web

"*Deliver*". También en el repositorio utilizando el plugin "*RSS-Module*" podría incluirse en la interfaz de los usuarios con el rol predefinido de Moodle "*creador de contenidos*" un bloque HTML con la lista de sindicación.

5.6. Resumen

En el presente capítulo se ha presentado el diseño de la arquitectura de SLOR y la implementación realizada de un prototipo que ha servido para la demostración y estudio de los principios de diseño de la arquitectura. La descripción ha sido realizada desde un nivel general hasta el detalle de cada componente incluyendo el código fuente relevante. El prototipo implementado permite la inclusión de expresiones semánticas y diferentes tipos de esquemas de registros, al basarse en el modelo flexible descrito en la sección 4.4.1. En el próximo capítulo se evaluarán las características del diseño que permitan corroborar la consecución de los objetivos definidos en la presente investigación.

Capítulo 6

Evaluación

The human creativity has produced many useless and even some harmful theories. How should we decide in which theories to trust? How are we to know? Because we use theories to help us predict, we should evaluate them by how well they actually make predictions. A good theory is a theory that predicts well. The same thing goes for our everyday beliefs.

Nils J. Nilsson

A lo largo de este capítulo se describen el conjunto de casos de estudio y actividades de prueba realizadas con el objeto de contrastar el buen funcionamiento del prototipo creado y la consecución de los objetivos planteados en el primer capítulo.

6.1. Introducción

La evaluación de los mecanismos semánticos orientados a la flexibilidad de los repositorios para objetos de aprendizaje propuestos en el presente trabajo se lleva a cabo en varios ámbitos, con las siguientes tareas de evaluación:

1. Evaluar la compleción del modelo de datos del repositorio, mediante actividades

de evaluación encaminadas a la comprobación de que el modelo de representación de datos del repositorio permite albergar registros de metadatos completos según el esquema de metadatos para objetos de aprendizaje más utilizado, IEEE LOM.

2. Evaluar que el modelo semántico del repositorio amplía los esquemas de metadatos de objetos de aprendizaje existentes con semántica computacional. Se describen las actividades de evaluación encaminadas a comprobar que el modelo semántico del repositorio en OWL proporciona soporte para la inclusión de expresiones semánticas adicionales (no sustitutorias) en los registros de metadatos definidos en el modelo conceptual. Hay que remarcar que además de la compatibilidad hacia atrás con modelos de metadatos existentes (IEEE LOM) es posible añadir información específicamente semántica.

3. Evaluar la flexibilidad del modelo semántico del repositorio mediante actividades de evaluación encaminadas a comprobar que el modelo semántico del repositorio permite almacenar registros de metadatos en OWL según diferentes conceptualizaciones presentes (IEEE LOM, Dublin Core, LearningObjectAsAnything) y futuras (SpanishLOMCore, etc.)

En forma de esquema, en la tabla 6.1 se enuncian los códigos con los que se hará referencia a las evaluaciones realizadas.

En el resto del capítulo se describe, desde los distintos puntos de vista aludidos, la evaluación de los mecanismos semánticos basados en la arquitectura propuesta e implementados en el prototipo del repositorio descrito en el capítulo anterior. Esta evaluación se utilizará para comprobar que los objetivos de la investigación han sido alcanzados.

E_1	Compleción
E_1A_1	Comprobar que el modelo de datos del repositorio permite crear, almacenar, recuperar y modificar registros de metadatos IEEE LOM completos.
E_2	Semántica computacional
E_2A_1	Comprobar que es posible la creación de instancias de metadatos con información semántica adicional sobre una base IEEE LOM, así como su almacenamiento y recuperación del repositorio.
E_2A_2	Diseño y ejecución de casos de prueba que demuestren que es posible llevar a cabo operaciones estrictamente semánticas (tales como búsquedas) sobre las instancias creadas en E_2A_1
E_3	Evaluar la flexibilidad del modelo semántico del repositorio
E_3A_1	Comprobar que el modelo de datos del repositorio permite albergar registros de metadatos completos según otros esquemas de metadatos no LOM.
E_3A_2	Coexistencia entre modelos. Soporte de perfiles de aplicación IEEE LOM.

Tabla 6.1: Actividades de evaluación

6.2. Compleción (E_1)

Para conseguir evaluar la compleción del modelo presentado en esta investigación, se ha elegido la conceptualización IEEE LOM por ser la más utilizada en la mayoría de los sistemas gestores de aprendizaje (LMS). Este estándar, descrito en la sección 2.4.2 del capítulo 2, esta formado por más de 70 campos de metadatos agrupados en diferentes categorías que permiten describir un objeto de aprendizaje.

6.2.1. Actividad (E_1A_1): compleción IEEE LOM

En esta actividad se presentan las siguientes tareas a realizar:

1. Definición del esquema de metadatos IEEE LOM en OWL basado en el esquema aportado por SLOR.

2. Inclusión de un registro IEEE LOM completo según el esquema desarrollado.

Para conseguir el primer objetivo, se irá describiendo a lo largo de esta sección la correlación de tipos, registros y propiedades definidos en IEEE LOM con el nuevo esquema LOM OWL. Seguidamente, se presentará una prueba de inclusión de un registro de metadatos representado en IEEE LOM bajo el esquema definido LOM OWL dentro del repositorio SLOR. Finalmente, y a modo resumen, se presentarán las conclusiones del desarrollo y la ejecución de la actividad.

Definición LOM OWL

Un objeto de aprendizaje descrito con un registro de metadatos IEEE LOM encaja dentro de la definición "*LearningObject_AsAnythingDigital*" definida en la sección 4.2.2 del capítulo 4. Se ha creado el término "*LOR_LOM*" que agrupa todas las conceptualizaciones de objeto de aprendizaje descritas con un registro de metadatos "*LOMOWL_Record*" asociado.

Correlacion de tipos IEEE LOM - LOM OWL

- LOM CharacterStrings: tipo de dato que representa un conjunto de caracteres. Este tipo de dato posee una correspondencia directa con el tipo "*xsd:string*" que representa una cadena de caracteres.

IEEE LOM datatype	**LOM OWL**
CharacterString	xsd:string

- LOM LangString: tipo de dato que representa una o varias cadenas de caracteres escritas en un lenguaje humano (español, inglés, etc..). Según la definición un "*LangString*" puede ser formado por una estructura de múltiples pares "*(Language,CharacterString)*". Dado que OpenCyc define el término "*oc:HumanLanguage*" como:

  > *"Una especialización del concepto Lenguaje. Cada instancia es un lenguaje utilizado en los actos de comunicación humana. Incluye el conjunto de lenguajes naturales en actual uso y también los que están en desuso (lenguajes muertos), como el griego antiguo o el latín, también incluye los inventados por el hombre como el esperanto."*

  La definición del lenguaje en un LangString puede ser correlacionada con una instancia del término "*oc:HumanLanguage*".
  La correlación con el modelo de SLOR en el rango de la propiedad "*LangString*" es con la clase "*slor:simpleString*" que representa cualquier cadena escrita en un lenguaje humano, explicada en la sección 5.3.1 del capítulo 5.

IEEE LOM datatype	**LOM OWL**
LangString	slor:simpleString

- LOM Date_Time: tipo de dato que representa un punto en el tiempo expresado con una precisión de hasta un segundo.

$$YYYY - MM - DDThh : mm : ss[Z|(+|-)hh : mm]$$

  La correlación se puede establecer con el tipo de dato "*xsd:dateTime*". En algunos registros en IEEE LOM también se permite incluir una cadena escrita, por lo que para mantener la compatibilidad se ha diseñado la clase "*lom:DateTime*" con dos propiedades asociadas "*lom:dateTimeValue*" de rango "*xsd:dateTime*"

con multiplicidad [1..1] y "*lom:dateTimeDescription*" con rango "*LangString*" y multiplicidad [0..*].

IEEE LOM datatype	**LOM OWL**
DateTime	lom:DateTime

- LOM Duration: tipo de dato que representa un intervalo de tiempo con un error menor de un segundo.

$$P[aA][mM[dD][T[hH][nM][s[.s]S]]$$

La correlación puede establecerse con el tipo de dato "*xsd:duration*". Al igual que con el tipo "*lom:DateTime*", en algunos registros en IEEE LOM también se permite incluir una cadena escrita, por lo que se ha definido la clase "*lom:Duration*" con dos propiedades asociadas, "*lom:durationValue*" de rango "*xsd:duration*" con multiplicidad [1..1] y "*lom:durationDescription*" de rango "*lom:LangString*" y multiplicidad [0..n].

IEEE LOM datatype	**LOM OWL**
DateTime	lom:Duration

- LOM Vocabularies: los elementos de vocabulario de IEEE LOM son definidos como pares de cadenas de caracteres "*CharacterString*" con la estructura "*(isource,value)*" siendo "*isource*" un URI que identifica el tipo de vocabulario. Este tipo de elementos pueden ser correlacionados con el término definido "*ConceptLink*" de la forma:

$$Conceptual\_Link \equiv oc\_ConceptLink \cup owl\_ConceptLink$$

Con el término "*Conceptual_Link*" se hacen referencia a las instancias correspondientes con enlaces a otros vocabularios OWL o a términos de la base

de conocimiento OpenCyc. En el capítulo anterior en la sección 5.3.1 han sido explicados previamente. Por ello puede establecer la siguiente relación de equivalencia:

$$lom : Vocabulary \equiv Conceptual\_Link$$

IEEE LOM datatype	**LOM OWL**
Vocabulary	lom:Vocabulary

Inclusión de semántica en la correlación de tipos

La correlación de tipos se ha realizado atendiendo a las siguientes características:

- Compatibilidad completa con el esquema IEEE LOM.
- Posibilidad de inclusión de expresiones semánticas.

Para incluir diferentes tipos de expresiones semánticas se ha definido la conceptualización "*Semantic_Expression*" de la siguiente forma:

$$Semantic\_Expression \equiv Conceptual\_Link \cup SemanticProperty\_Link$$

Que incluye cualquier tipo de expresión semántica posible que haya sido definida bajo el esquema de SLOR descrito en la sección 5.3.1 del capítulo anterior. Las propiedades OWL definidas en el esquema de correlación IEEE LOM pueden incluir enlaces a los tipos de datos básicos o a este tipo de expresiones.

El tipo de dato "*lom:Vocabulary*" define cualquier tipo de término incluido en un vocabulario con posibilidad de ser referenciado en campos IEEE LOM. Varias propiedades de la correlación LOM-OWL tienen como rango este tipo de elemento. En la tarea de creación de registros utilizando herramientas como "*SLOR Intellisense*" es necesario conocer los vocabularios incluidos por IEEE LOM "*LOMv1.0*" u otros

manejados por el repositorio. La propiedad de anotación "*propertyContext*" ayuda a este tipo de herramientas a reconocer los vocabularios que puede manejar para ayudar al usuario en su tarea de creación o búsqueda de registros de metadatos.

A continuación se presenta la correlación de IEEE LOM con la ontología LOM OWL utilizada para la inserción de registros LOM completos dentro de SLOR. Se presentan las nueve categorías definidas en el estándar LOM.

Categoría 1. GENERAL

Las propiedades IEEE LOM "*1.1 Identifier*" y "*1.2 Title*" se correlacionan con las propiedades básicas del registro de metadatos "*LOMOWL_Record*" indicadas a continuación:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.1 Identifier	lom:Identifier [1..1]	lom:Identifier
1.2 Title	lom:Title[1..1]	lom:LangString

La clase "*lom:Identifier*" incluye dos propiedades, "*lom:catalog*" que establece el catálogo utilizado en la definición del identificador y "*lom:entry*" que incluye el valor del identificador. Ambas propiedades poseen de rango valores de tipo "*xsd:string*"

El lenguaje es una propiedad con rango "*oc:HumanLanguage*" que especifica los lenguajes con los que ha sido descrito el objeto de aprendizaje.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.3 Language	slor:language [1..*]	oc:HumanLanguage

El elemento "*1.4 Description*" incluye las descripciones realizadas en lenguaje

natural sobre el objeto de aprendizaje. Como correlación directa se puede establecer el elemento "*lom:LangString*", si bien en esta propiedad pueden asociarse a su vez expresiones semánticas "*slor:Semantic_Expression*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.4 Description	lom:description	lom:LangString ∪ slor:SemanticExpression

Una palabra clave proporciona un medio rápido para localizar el objeto de aprendizaje, en IEEE LOM quedan contempladas en el elemento "*1.5 Keyword*". La correlación directa es con un elemento de tipo "*lom:LangString*", pero también pueden incluirse referencias a conceptos de otras ontologías definidas en OWL o de la base de conocimiento OpenCyc incluyendo en el rango el término definido "*slor:Concept_Link*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.5 Keyword	lom:keyword [0..*]	lom:LangString ∪ slor:Concept_Link

El elemento "*1.6 Coverage*" se utiliza para describir el tiempo, la cultura, geografía o región en donde se utiliza el objeto de aprendizaje, para lo cual el estándar propone incluir localizaciones espaciales (nombre de un lugar o coordenadas geográficas), periodos temporales (identificador del periodo, fecha o rango de fechas) o jurisdicciones (como un nombre de una entidad administrativa).

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.6 Coverage	lom:coverage [0..*]	lom:LangString ∪ oc:GeographicalRegion ∪ oc:TimeInterval ∪ oc:AdministrativeUnit ∪ slor:OwlLink

Para incluir referencias a conceptos de otras ontologías definidas en OWL se ha incluido la clase "*slor:OwlLink*" dentro del rango de la propiedad "*lom:coverage*". A la hora de cargar una ontología dentro del sistema puede indicarse la correlación de rangos explícitos a asociar, de esta forma es posible facilitar las tareas del control "*Intellisense*".

A continuación se describen las definiciones OpenCyc de los términos utilizados en el rango de la propiedad:

- oc:GeographicalRegion : *"Una región espacial real que incluye alguna parte de una superficie de un planeta (normalmente del planeta tierra "*`PlanetEarth`*"), y debe poder ser representada sobre un mapa del planeta. Este concepto incluye regiones topográficas como montañas y espacios sumergidos por agua, lugares definidos por demografía (ej. areas de territorio por lenguaje) y cualquier otro tipo de demarcacion territorial".*

- oc:TimeInterval : *"Un intangible temporal caracterizado por su extensión temporal. De esta forma, el año 1969 DC es un "*`TimeInterval`*"; por todas las cosas interesantes que ocurrieron en ese año, el año queda definido completamente por su extensión temporal."*

- oc:AdministrativeUnit : cada instancia es una unidad con responsabilidades administrativas.

El elemento LOM "*1.7 Structure*" se refiere a la estructura de organización de un objeto de aprendizaje. Este valor puede ser especificado por uno de los múltiples términos definidos en una ontología OWL.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.7 Structure	lom:structure [1..1]	lom:Vocabulary

En el caso de la ontología diseñada, se han incluido los términos definidos por LOM ("*atomic*", "*collection*", "*networked*", "*hierarchical*" y "*linear*") como instancias de la clase "*lom:StructureVocabularyItem*". Para incluir este tipo de conceptos se ha de crear una instancia de la clase "*lom:StructureVocabulary*" que establezca en la propiedad "*ref_Term*" el término al que referencia. A continuación se describe con lógica de descripciones la clase "*lom:StructureVocabulary*" asociada al rango de la propiedad "*lom:structure*":

$$lom : StructureVocabulary \equiv owl\_ConceptLink$$

$$lom : StructureVocabulary \sqsubseteq lom : Vocabulary$$

$$ontologySchemaLink \epsilon \{\text{“}LOMv1{,}0''\}$$

$$ref\_Term \exists (lom : StructureVocabularyItem)$$

Es una clase equivalente a un enlace de tipo "*owl_ConceptLink*" en el que la propiedad "*ontologySchemaLink*" de tipo "*xsd:anyURI*" ha de contener el valor "*LOMv1.0*" y la propiedad "*ref_Term*" ha de referenciar algún valor de tipo "*lom:StructureVocabularyItem*".

Finalmente, el elemento "*1.8 Aggregation level*" permite especificar la granularidad del objeto de aprendizaje. Sus valores son definidos en varios términos de una ontología. En la ontología presentada se definen los términos como instancias de

la clase "*lom:AggregationLevelItem*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
1.8 Aggregation Level	lom:aggregationLevel[1..1]	lom:Vocabulary

De la misma forma que el elemento anterior, se ha creado la clase conceptual "*lom:AggregationLevelVocabulary*" utilizada para incluir las referencias a los elementos "*lom:AggregationLevelItem*" en la propiedad "*lom:aggregationLevel*". La clase se describe en lógica de descripciones de la siguiente forma:

$$lom : AggregationLevelVocabulary \equiv owl\_ConceptLink$$
$$lom : AggregationLevelVocabulary \sqsubseteq lom : Vocabulary$$
$$ontologySchemaLink \in \{\text{“}LOMv1{,}0''\}$$
$$ref\_Term\exists(lom : AggregationLevelItem)$$

En el que la propiedad "*ontologySchemaLink*" de tipo "*xsd:anyURI*" ha de contener el valor "*LOMv1.0*" y la propiedad "*ref_Term*" ha de referenciar algún valor de tipo "*lom:AggregationLevelVocabulary*".

Categoría 2. LIFECYCLE

El primer elemento de esta categoría es "*2.1 Version*", que define la información referente a la edición del objeto de aprendizaje.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
2.1 Version	lom:version[0..1]	lom:LangString

La propiedad "*2.2 Status*" referencia valores de un vocabulario determinado que permiten indicar el estado de edición del objeto de aprendizaje.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
2.2 Status	lom:status[0..1]	lom:Vocabulary

Los elementos definidos en la ontología propuesta son instancias de la clase definida "*lom:StatusVocabularyItem*", que pueden ser enlazados en la propiedad a través de una instancia de la clase "*lom:StatusVocabulary*". A continuación se expone la definición de la clase en lógica de descripciones:

$$lom : StatusVocabulary \equiv owl\_ConceptLink$$
$$lom : StatusVocabulary \sqsubseteq lom : Vocabulary$$
$$ontologySchemaLink \in \{\text{“}LOMv1{,}0''\}$$
$$ref\_Term \exists (lom : StatusVocabularyItem)$$

Según las restricciones descritas, la propiedad "*ontologySchemaLink*" de tipo "*xsd:anyURI*" ha de contener el valor "*LOMv1.0*" y la propiedad "*ref_Term*" ha de referenciar algún valor de tipo "*lom:StatusVocabulary*".

Finalmente, el campo "*2.3 contribute*" permite describir en un registro de metadatos las entidades (personas u organizaciones) que han contribuido al desarrollo del objeto de aprendizaje durante su ciclo de vida (ej. creación, edición, publicación). Las contribuciones pueden ser consideradas en sentido general, como todas las acciones que afectan al estado de un objeto de aprendizaje.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
2.3 Contribute	lom:contributeLifeCycle [0..*]	lom:Contribute

La clase "*lom:Contribute*" representa la información de algún tipo de contribución (como autor, revisor, etc...). Las propiedades utilizadas en estos términos son las siguientes:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
2.3.1 Role	lom:contributeRole [1..1]	lom:Vocabulary
2.3.2 Entity	lom:entity [1..*]	vCard ∪ oc:Organization
2.3.3 Date	lom:contributeDate [0..1]	lom:DateTime

Al igual que en la correlación de otros términos de vocabulario específicos de LOM, se ha creado la clase "*lom:RolesVocabularyItem*" junto con las instancias correspondientes ("*author*", "*review*", "*graphic*" "*designer*", etcetera). Estos términos son referenciados a través de la clase definida "*lom:RolesVocabulary*". Las instancias incluidas en el diseño con correlación directa según el vocabulario LOMv1.0 aparecen representadas en la figura 6.1.

Como términos de vocabulario pueden incluirse de la base de conocimiento las instancias de la clase "*oc:Role*".

Respecto a la entidad, se ha decidido utilizar el término "*oc:Organization*", según OpenCyc cada instancia es: "un grupo de miembros de tipo agente inteligente "*oc:IntelligentAgent*". En cada una de estas instancias existen ciertas relaciones y obligaciones entre los miembros de una organización. La instancias de "*oc:Organization*" incluyen cualquier tipo de organización. También cada una de estas instancias pueden realizar proyectos, establecer acuerdos y otro tipo de tareas características de los agentes que la componen.

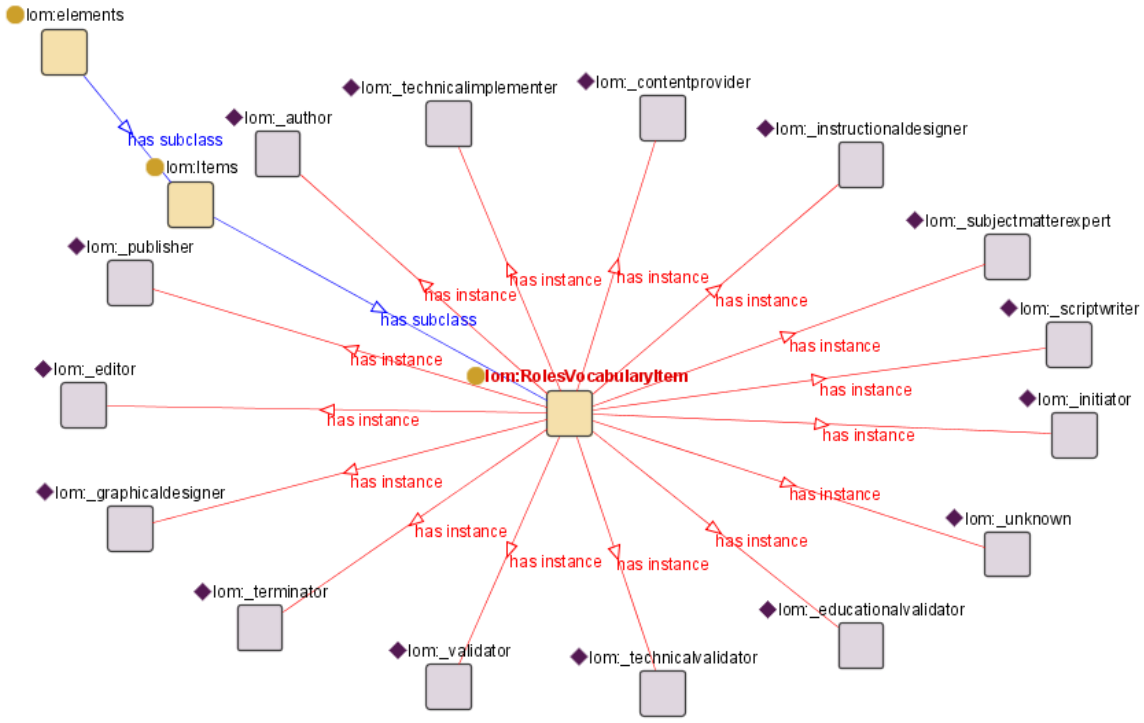

Figura 6.1: Jerarquía e instancias de la clase "*lom:RolesVocabularyItem*"

Categoría 3. META-METADATA

Esta categoría incluye las propiedades que permiten describir aspectos específicos de un registro de metadatos LOM de un objeto de aprendizaje. Cada registro posee un identificador, el campo LOM "*3.1 Identifier*", que se correlaciona con la propiedad "*lom:Identifier*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
3.1 Identifier	lom:identifier [0..1]	lom:Identifier
3.2 Contribute	lom:contributeMetadata [0..1]	lom:Contribute
3.3 Schema	lom:metadataSchema[0..1]	lom:MetadataSchema
3.4 Language	lom:language[0..1]	oc:HumanLanguage

Categoría 4. TECHNICAL

El elemento "*4.1 Format*" viene representado en IEEE LOM como un conjunto de caracteres que describen el formato técnico del objeto de aprendizaje. Este tipo de información puede ser representada según los tipos MIME basados en el registro RFC2048:1996 de IANA [FKP96]. La correlación de este elemento ha sido definida con la propiedad "*lom:format*", con rango "*iana:mimeType*" o "*lom:Non-Digital*". La clase "*iana:MimeType*" ha sido diseñada siguiendo el modelo descrito en el RFC2048:1996, incluyendo la propiedad "*iana:type*" que representa la información sobre el tipo MIME base. Para facilitar el tratamiento automático se ha incluido la propiedad "*iana:MimeEntry*" que describe el identificador completo del tipo MIME (ej: application/zip).

Además de utilizar la definición de tipos MIME, LOM permite especificar el tipo "non-digital" para objetos de aprendizaje no digitales. Según el esquema de SLOR, las individualizaciones de registros de objetos de aprendizaje se correlacionan con el tipo "*LearningObject-AsAnyThingWithEducationalPurpose*". Para establecer la correlación con este tipo de información y mantener la compatibilidad con IEEE LOM se ha definido el término "*lom:Non-Digital*" para ser incluido dentro del rango de la propiedad "*lom:format*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
4.1 Format	lom:format [0..*]	iana:MimeType *cup* lom:Non-Digital

El campo "*4.2 Size*" indica el tamaño del objeto de aprendizaje en bytes. La correlación realizada ha sido con el tipo de dato "*xsd:long*".

El elemento "*4.3 Location*" especifica el modo de acceso al objeto por alguna

URI o URL, por lo que se ha correlacionado con el tipo de dato "*xsd:anyURI*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
4.2 Size	lom:size [0..1]	xsd:long
4.3 Location	lom:location [0..*]	xsd:anyURI

El valor semántico de la propiedad "*lom:location*" posee la siguiente equivalencia:

$$lom : location \equiv oc : fileFromURI$$

El elemento "*4.4 Requirement*" sirve para indicar las características técnicas necesarias para utilizar el objeto de aprendizaje descrito. La inclusión de este tipo de características se realiza mediante estructuras de composición de requisitos técnicos (elemento "*4.4.1 OrComposite*" de IEEE LOM). En el diseño de la ontología se ha correlacionado esta estructura de composición con la clase "*lom:OrComposite*" y la propiedad asociada "*lom:hasTechnical Requirement*". Para incluir cada uno de los requisitos técnicos en el registro de metadatos LOM se ha creado la clase "*lom:TechnicalRequirement*", que permite describir este tipo de características con las propiedades correlacionadas (4.3.1.1, 4.3.1.2, 4.3.1.3 y 4.3.1.4).

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
4.4 Requirement	lom:requirement [0..*]	lom:OrComposite
4.4.1 OrComposite	lom:hasTechnical-Requirement[0..*]	lom:TechnicalRequirement
4.4.1.1 Type	lom:requirementType [0..1]	lom:Vocabulary
4.4.1.2 Name	lom:requirementName [0..1]	lom:Vocabulary
4.4.1.3 Minimum Version	lom:minVersion [0..1]	xsd:string
4.4.1.4 Maximum Version	lom:maxVersion [0..1]	xsd:string

De acuerdo con el estándar IEEE LOM se ha creado el vocabulario indicado para la definición de los términos de las propiedades 4.3.1.1 y 4.3.1.2. De acuerdo al vocabulario LOMv1.0 se han creado las clases "*lom:RequirementTypesVocabulary*" y "*lom:RequirementValuesVocabulary*".

La clase "*lom:RequirementTypesVocabulary*" posee la siguiente definición en lógica de descripciones:

$$lom : RequirementTypesVocabulary \sqsubseteq lom : RequirementVocabulary$$
$$ref\_Term \in (lom : browser) \sqcup ref\_Term \in (lom : operating\_system)$$

La clase "*lom:RequirementValuesVocabulary*" se define en lógica de descripciones de la siguiente forma:

$$lom : RequirementValuesVocabulary \sqsubseteq lom : RequirementVocabulary$$
$$ref\_Term \in (lom : RequirementOperatingSystemValue) \sqcup$$
$$ref\_Term \in (lom : RequirementBrowserValue)$$

Ambas definiciones hacen referencia a instancias de elementos del término que define un tipo de requisito "*lom:RequirementType*", y sus especializaciones directas "*lom:RequirementOperatingSystemValue*" o "*lom:Requirement BrowserValue*" como puede apreciarse en la figura 6.2. Ejemplos de instancias de requisitos referentes al tipo de sistema operativo "*lom:RequirementOperatingSystemValue*" son "*lom:macos*", "*lom:ms-windows*" o "*lom:unix*". Ejemplos de instancias correspondientes al tipo de requisito navegador web "*lom: RequirementBrowserValue*" son "*lom:ms-internet_explorer*", "*lom:amaya*" o "*lom:opera*".

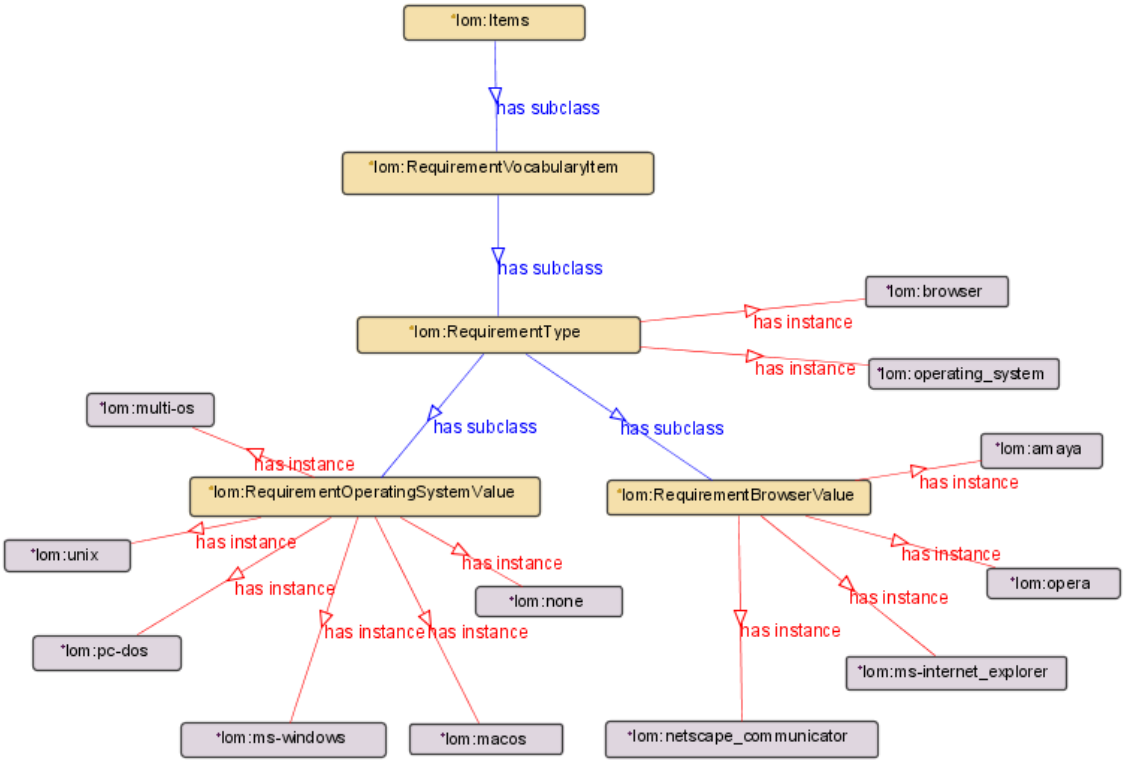

Figura 6.2: Jerarquía e instancias de la clase "*lom:RequirementVocabularyItem*"

Para describir los procedimientos de instalación IEEE LOM tiene definido el elemento "*4.5 Installation Remarks*". También se pueden indicar algunos requisitos específicos utilizando el elemento "*4.6 Other platform requirements*". Si el objeto de aprendizaje posee una duración continua en el tiempo puede indicarse con el elemento "*4.7 Duration*". Estos elementos se correlacionan con las propiedades OWL de la siguiente forma:

IEEE LOM element	LOM OWL - Propiedades LOR_LOM	Rango
4.5 Installation Remarks	lom:installationRemarks [0..*]	lom:LangString
4.6 Other platform requirements	lom:otherPlatformRequirements [0..*]	lom:LangString
4.7 Duration	lom:duration[0..1]	lom:Duration

Categoría 5. EDUCATIONAL

El primer elemento de esta categoría definido en el estándar es "*5.1 Interativity type*", que sirve para describir la información sobre el tipo de interactividad del objeto de aprendizaje. En la correlación de propiedades y tipos de la ontología se ha creado el término "*lom:InteractivityTypeVocabularyItem*", cuyas instancias son los distintos tipos de interactividad especificados en el esquema LOMv1.0 (ver figura 6.3.

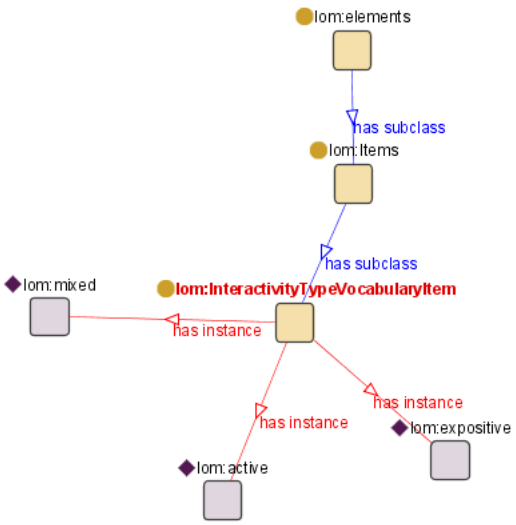

Figura 6.3: Jerarquía e instancias de la clase "*lom:InteractivityTypeVocabulary-Item*"

La correlación establecida de este elemento con la ontología LOM OWL es la siguiente:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.1 Interactivity Type	lom:interactivityType [0..*]	lom:Vocabulary

Al igual que con los anteriores elementos que hacen referencia a un vocabulario, se ha creado la clase definida "*lom:InteractivityTypeVocabulary*" que representa el conjunto de términos definidos para los distintos tipos de interactividad.

$$lom : InteractivityTypeVocabulary \equiv owl\_ConceptLink$$
$$lom : InteractivityTypeVocabulary \sqsubseteq lom : Vocabulary$$
$$ontologySchemaLink \epsilon \{"LOMv1{,}0"\}$$
$$ref\_Term \exists (lom : InteractivityTypeVocabularyItem)$$

El elemento "*5.2 Learning Resource Type*" indica el tipo de recurso de aprendizaje. En la ontología LOM OWL se ha creado una clase cuyas instancias son los distintos tipos de recursos de aprendizaje contemplados por el vocabulario LOMv1.0, la clase es "*lom:LearningResourceTypeVocabularyItem*".

También se ha definido la clase "*lom:LearningResourceTypeVocabulary*" que especifica el tipo de enlace en la propiedad "*slor:ref_Term*" con los términos de la clase "*lom:LearningResourceVocabulatyItem*". Se han utilizado dos propiedades para definir la correlación establecida con el elemento 5.2 de IEEE LOM. La primera es "*lom:learningResourceType*", que especifica los tipos de recursos descritos por el registro de metadatos. Si un objeto incluye diferentes tipos de recursos, se utiliza la segunda propiedad para indicar el tipo de recurso predominante estableciendo el

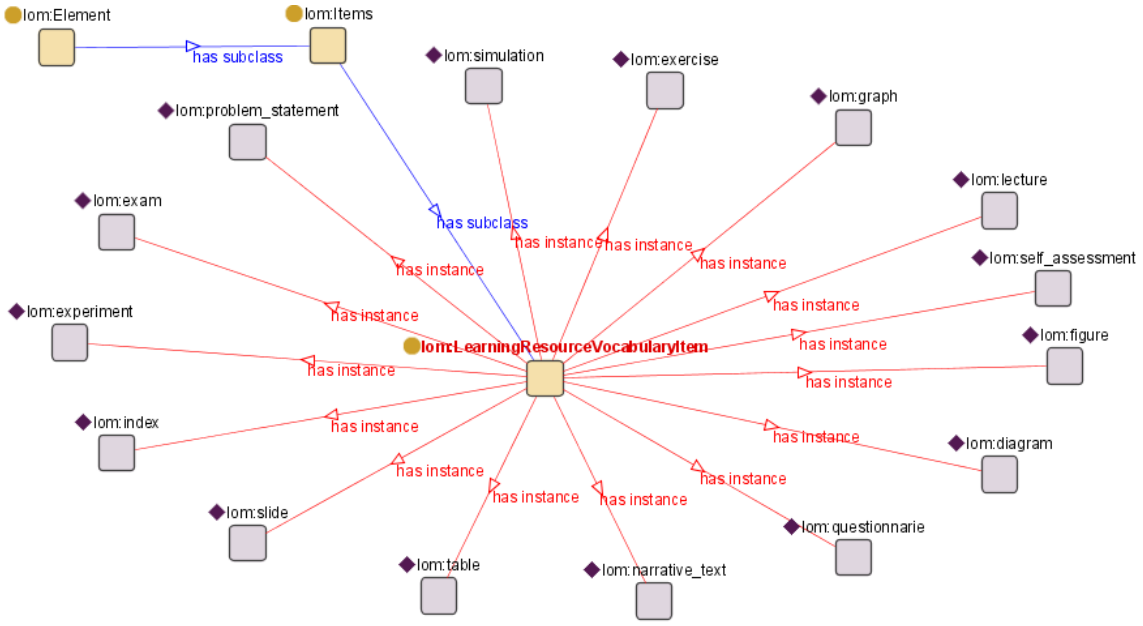

Figura 6.4: Jerarquía e instancias de la clase "*lom:LearningResourceVocabulary-Item*"

valor correspondiente en la propiedad "*lom:dominantLearningResourceType*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.2 Learning Resource type	lom:learningResourceType [0..*] y lom:dominantLearning ResourceType[0..1]	lom:Vocabulary

El elemento "*5.3 Interactivity level*" sirve para indicar el grado de interactividad del objeto de aprendizaje según las características de su diseño. El estándar aporta un conjunto de términos como vocabulario posible a utilizar en la definición de este elemento. En la ontología LOM OWL se ha definido la clase "*lom:LevelVocabularyItem*" cuyas instancias representan a estos términos. También se ha creado la clase "*lom:InteractivityLevelVocabulary*" que define los enlaces a términos LOMv1.0 asociados con el elemento 5.3.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.3 Interativity Level	lom:interactivityLevel VocabularyItem[0..1]	lom:Vocabulary

El elemento "*5.4 Semantic density*" sirve para describir el grado de concisión de un objeto de aprendizaje. Los niveles de densidad son los mismos términos que los especificados en el elemento anterior, por tanto se pueden utilizar los definidos en la clase "*lom:LevelVocabularyItem*". Por otro lado, se ha creado la clase definida "*lom:SemanticDensityVocabulary*" utilizada para la definición de los enlaces a los términos LOMv1.0 asociados con el elemento 5.4.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.4 Semantic Density	lom:semanticDensity [0..1]	lom:Vocabulary

El elemento "*5.5 Intended end user role*" sirve para indicar el usuario principal para el que ha sido diseñado el objeto de aprendizaje. El estándar aporta el conjunto de términos como vocabulario posible de utilizar en la definición de este elemento. Se ha definido la clase "*lom:IntendedEndUserRoleVocabularyItem*" cuyas instancias representan a estos términos. Como con los anteriores elementos que hacen referencia a un término de un vocabulario específico, se ha creado la clase "*lom:IntendedEndUserRoleVocabulary*" para definir los enlaces a términos LOMv1.0 asociados con el elemento 5.5.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.5 Intended end user role	lom:intendedEndUserRole [0..*]	lom:Vocabulary

La información sobre el entorno principal en el que se utilizará el objeto educativo

se recoge en el elemento IEEE LOM "*5.6 Context*".

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.6 Context	lom:context [0..*]	lom:Vocabulary

Los términos del vocabulario LOMv1.0 respecto a este elemento se correlacionan con las instancias de la clase "*lom:ContextVocabularyItem*". También se ha creado la clase "*lom:ContextVocabulary*" para definir los enlaces a términos LOMv1.0 asociados con el elemento 5.6 aunque también se pueden incluir enlaces a microteorías de OpenCyc "*oc:MicroTheory*". Una microteoría en OpenCyc se define en un dominio específico de conocimiento.

El elemento "*5.7 Typical age range*" se refiere a la edad de desarrollo intelectual necesaria para el destinatario.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.7 Typical Age Range	lom:typicalAgeRange [0..*]	lom:LangString o lom:SemanticAgeRange

Desde la perspectiva semántica, se delimitan de forma estructurada las posibles interpretaciones de edad de desarrollo intelectual en las instancias del término "*lom:SemanticAgeRage*" con 3 propiedades:

- "*lom:descriptionAgeRange[0..*]*" de rango "*lom:LangString*": descripción del rango de edad.
- "*lom:minAgeRange[0..1]*" de rango "*xsd:int*": límite mínimo del intervalo cerrado del rango de edad.
- "*lom:maxAgeRange[0..1]*" de rango "*xsd:int*": límite máximo del intervalo cerrado del rango de edad.

El grado de dificultad del objeto de aprendizaje se define con el elemento "*5.8 Difficulty*". Los términos correspondientes al vocabulario LOMv1.0 de este elemento han sido definidos como instancias de la clase "*lom:DifficultyVocabularyItem*". También se ha creado la clase "*lom:DifficultyVocabulary*" para identificar los enlaces a este tipo de términos.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
5.8 Difficulty	lom:difficulty [0..*]	lom:Vocabulary

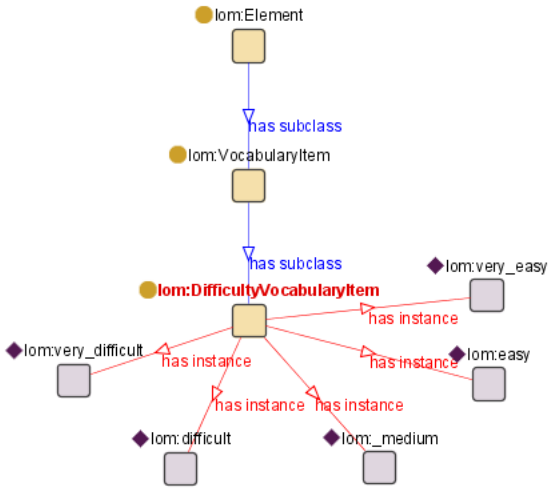

Figura 6.5: Jerarquía e instancias de la clase "*lom:DifficultyVocabularyItem*"

El elemento "*5.9 Typical Learning Time*" define el tiempo aproximado o típico que necesitan para asimilar el objeto educativo los destinatarios objetivo típicos. El elemento "*5.10 Description*" sirve para indicar mediante comentarios cómo debe utilizarse el objeto. Finalmente, el término "*5.11 Language*" expone el idioma utilizado por el destinatario típico de este objeto educativo. La correlación de estas propiedades se muestra a continuación:

IEEE LOM element	LOM OWL - Propiedades LOR_LOM	Rango
5.9 Typical Learning Time	lom:typicalLearningTime [0..1]	lom:Duration
5.10 Description	lom:educationalDescription [0..*]	lom:LangString
5.11 Language	lom:userLanguage [0..*]	oc:HumanLanguage

Categoría 6. RIGHTS

En la categoría "*6. Rights*" se definen tres campos relacionados con los derechos de autor del objeto de aprendizaje descrito. El campo "*6.1 Cost*" permite indicar si el objeto de aprendizaje requiere de algún tipo de pago. El campo "*6.2 Copyright and other restrictions*" indica si existen derechos de autor u otras restricciones sobre el objeto educativo. Los términos definidos en LOMv1.0 para indicar los valores de estas propiedades han sido correlacionados como instancias de la clase "*lom:YesNoVocabularyItem*". Según el esquema definido previamente, se ha creado la conceptualización "*lom:YesNoVocabulary*" para indicar el enlace a los conceptos de este vocabulario.

IEEE LOM element	LOM OWL - Propiedades LOR_LOM	Rango
6.1 Cost	lom:cost [0..1]	lom:Vocabulary
6.2 Copyright and other restrictions	lom:lom:copyrightAnd OtherRestrictions [0..1]	lom:Vocabulary

Según el carácter semántico del esquema LOM OWL, este tipo de información puede ser especificada en mayor detalle con otros términos de vocabularios especializados en la descripción de derechos de autor y tipos de coste.

El elemento "*6.3 Description*" permite incluir comentarios sobre las condiciones de utilización del objeto de aprendizaje.

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
6.3 Description	lom:rightsDescription [0..*]	lom:LangString

Categoría 7. RELATION

En esta categoría se definen los elementos que permiten especificar el tipo de relaciones existentes con otros objetos de aprendizaje. Para indicar los diferentes tipos de relación con otros objetos de aprendizaje se ha creado la clase "*lom:Relation*", que permite a través de la propiedad "*lom:kindOfRelation*" indicar el tipo de relación. IEEE LOM aporta un conjunto de términos basados en Dublin Core, estos términos han sido correlacionados con las instancias de la clase "*lom:RelationVocabularyItem*" como puede apreciarse en la figura 6.6.

La correlación de elementos IEEE LOM establecida es la siguiente:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
7 Relation	lom:relation [0..*]	lom:Relation
7.1 Kind	lom:kindOfRelation [0..1]	lom:Vocabulary
7.2.1 Resource Identifier	lom:relatedLO [0..1]	lom:Identifier *cup* slor:LearningObject AsAnyThing
7.2.2 Resource Description	lom:relationLODescription [0..*]	lom:LangString

Categoría 8. ANNOTATION

Esta categoría proporciona los elementos necesarios para poder realizar comentarios sobre la utilización del objeto de aprendizaje, incluyendo la información sobre la creación del comentario. Una anotación IEEE LOM en el modelo LOM OWL es una instancia de la clase "*lom:Annotation*" que en su dominio posee asociadas las

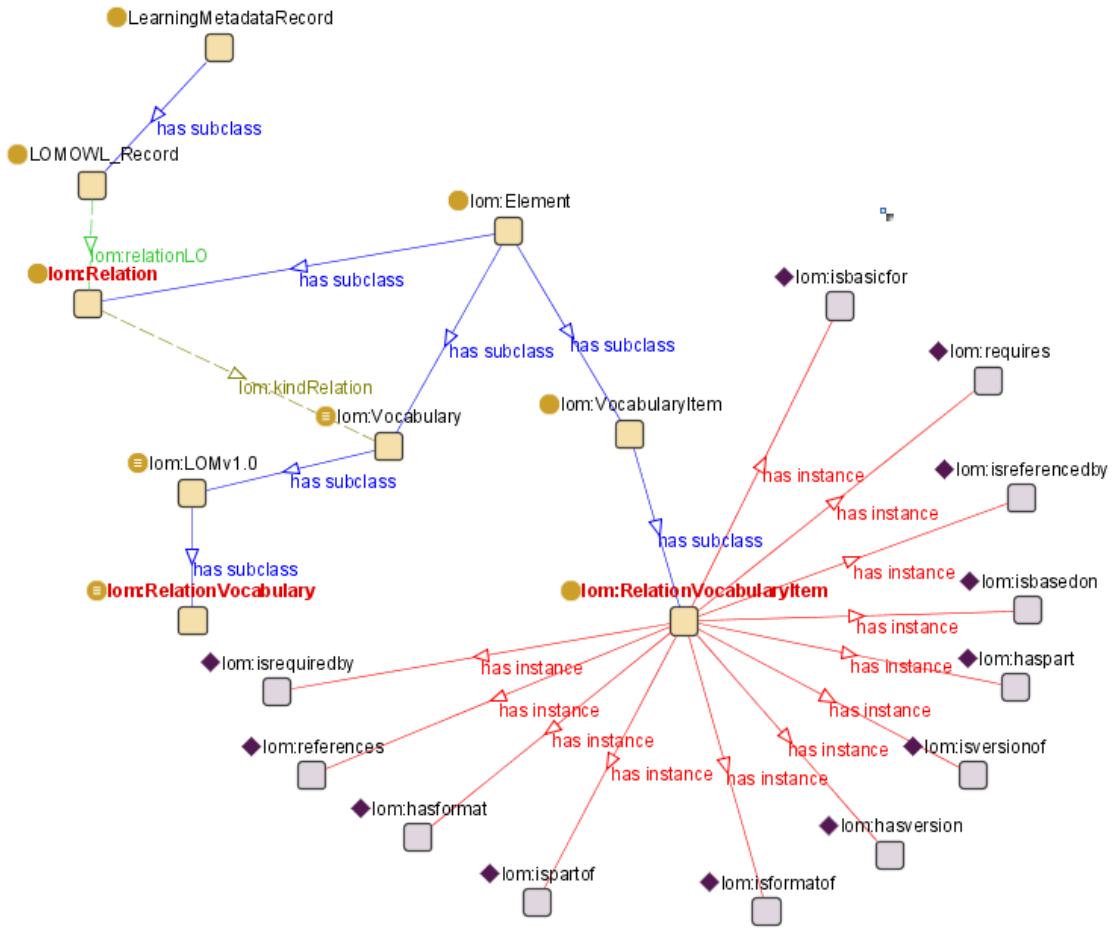

Figura 6.6: Jerarquía e instancias de la clase "*lom:RelationVocabularyItem*"

siguientes propiedades:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
8.1 Entity	lom:annotationEntity [0..1]	oc:Organization o vCard
8.2 Date	lom:annotationDate [0..1]	lom:DateTime
8.3 Description	lom:annotationDescription [0..*]	lom:LangString

Las anotaciones se realizan por medio de la propiedad "*lom:annotation*" con

dominio "*LOMOWL_Record*" definida en el esquema LOM OWL con rango de tipo "*lom:Annotation*".

Categoría 9. CLASSIFICATION

Esta categoría permite clasificar el objeto de aprendizaje dentro de un cierto sistema de clasificación, para ello en IEEE LOM es necesario especificar el propósito de clasificación y la ruta taxonómica. En la correlación establecida, los elementos definidos en el vocabulario LOMv1.0 son instancias de la clase que representa los términos de *propósito de clasificación* del vocabulario "*lom:PurposeVocabularyItem*". Al igual que en el resto de correlaciones definidas en los elementos anteriores, la clase "*lom:PurposeVocabulary*" define las relaciones de este tipo de conceptos.

La propiedad "*lom:classification*" de rango "*lom:Classification*" permite incluir los diferentes modos de clasificación del objeto de aprendizaje. Cada instancia de la clase "*lom:Classification*" incluye las propiedades siguientes:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
9.1 Purpose	lom:purpose [0..1]	oc:Organization *cup* vCard
9.2 Taxon	lom:taxon [0..1]	lom:TaxonID *cup* slor:conceptual_Link

La propiedad "*9.2 Taxon*" puede incluir una semántica más elaborada que la definida en el estándar LOM, permitiendo el enlace a diversos conceptos de otros vocabularios. Por ello, se ha incluido en el rango la clase "*slor:conceptual_Link*". Por ejemplo pueden utilizarse los términos definidos en la base de conocimiento OpenCyc tales como "*oc:Biological*" u "*oc:History*". Para mantener la compatibilidad con IEEE LOM e incluir descripciones sobre los elementos referenciados de un vocabulario, se ha definido la clase "*lom:Taxon*" con las siguiente correlación de propiedades:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
9.2.1 Source	lom:taxonSource [0..1]	lom:Identifier
9.2.2 Taxon	lom:refTaxon [0..1]	lom:TaxonID *cup* slor:conceptual_Link
9.2.3 Description	lom:taxonDescription[0..*]	lom:LangString
9.2.4 Keyword	lom:taxonKeyword[0..*]	lom:LangString *cup* slor:conceptual_Link

Tanto en la propiedad "*lom:refTaxon*" como en la propiedad "*lom:taxonKeyword*" pueden incluirse conceptos de otras ontologías e incluso de la base de conocimiento OpenCyc. Es útil para los mecanismos de consulta o navegación que requieran solicitar un conjunto de elementos. Todos los elementos descritos de clasificación poseen un identificador, instancia de la clase "*TaxonID*" que se define con las siguientes propiedades:

IEEE LOM element	**LOM OWL - Propiedades LOR_LOM**	**Rango**
9.2.2.1 Id	slor:identifier [0..1]	xsd:string
9.2.2.2 Entry	lom:taxonEntry [0..1]	lom:LangString

En la sección B.1.1 del apéndice B se ha incluido la ontología correspondiente con la correlación del esquema de metadatos IEEE LOM descrita en esta sección.

6.2.2. Definición de un registro de metadatos según el modelo LOM OWL

A continuación se exponen dos casos de prueba de inclusión de un registro LOM-OWL según la correlación establecida en la sección anterior. Se contempla la inclusión de un registro completo según todas las categorías de propiedades definidas por el estándar y la inclusión de un registro sólo con propiedades relevantes. Es importante

resaltar la compatibilidad hacia atrás soportada con los registros IEEE LOM por el nuevo esquema LOM-OWL.

Caso de prueba 1 - IEEE LOM Completo

Debido a las dimensiones del ejemplo, en la sección B.1.2 del apéndice B se ha incluido el código y el esquema correspondiente al objeto de aprendizaje "*IEEE LOM*" tomado como referencia de IMS[1]. La información ha sido correctamente representada con la ontología LOM-OWL, quedando almacenada en la base de datos del sistema de persistencia de SLOR.

Caso de prueba 2 - IEEE LOM Parcial

Se ha incluido como ejemplo de compleción parcial IEEE LOM un registro de metadatos que describe el tutorial *DNA From The Beginning* [2]. En la figura 6.7 se representan las clases, propiedades e individuales de la declaración del registro "*LOR_DNAFrom TheBeginning*". En el apéndice B se ha incluido en la sección B.4 el esquema y el código de la instancia correspondiente representado en lenguaje OWL.

6.3. Semántica computacional (E_2)

En este apartado se verifica el soporte de inclusión de expresiones semánticas sobre las propiedades de los registros de metadatos definidos en el modelo conceptual. Como se ha comentado en la sección 5.3.1 del capítulo 5, se ha diseñado un modelo que permite incluir enlaces a conceptos de una ontología descrita en OWL u OpenCyc, así como la posibilidad de incluir también predicados definidos en el modelo lógico de OpenCyc o relaciones con otros.

[1] http://www.imsglobal.org/metadata/mdv1p2p2/samples/ims/imsmdexample_schema.xml
[2] http://www.dnaftb.org/dnaftb/

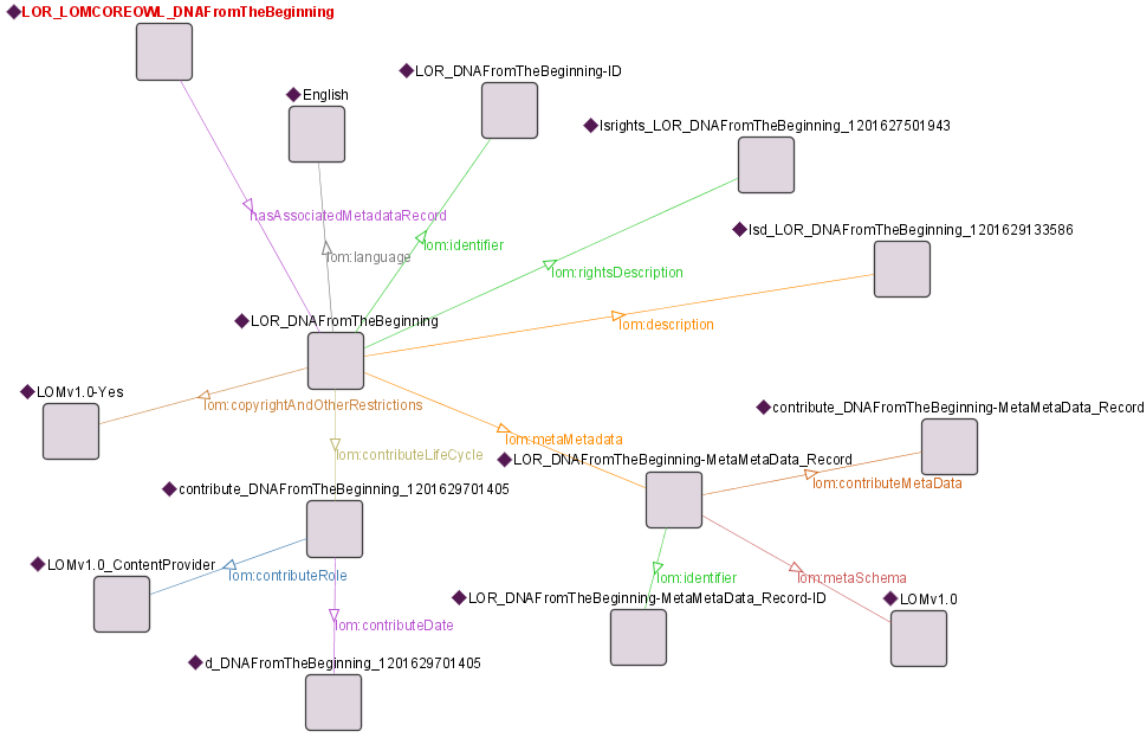

Figura 6.7: Registro de metadatos "*LOR_DNAFromTheBeginning*"

Para demostrar la utilidad de este diseño se han definido las siguientes actividades de evaluación:

- Comprobar que es posible la creación de instancias de metadatos con información semántica adicional sobre una base IEEE LOM, así como su almacenamiento y recuperación del repositorio.

- Diseño y ejecución de una batería de casos de prueba que demuestren que es posible llevar a cabo operaciones estrictamente semánticas (tales como búsquedas) sobre las instancias creadas en E_2A_1.

6.3.1. Inserción de diferentes expresiones formales que describen la semántica de los metadatos (E_2A_1)

Esta actividad de evaluación ha sido diseñada para demostrar la posibilidad de incluir diferentes tipos de expresiones semánticas sobre un registro de metadatos LOM definido formalmente según la ontología descrita en la sección 6.2.1 del presente capítulo, así como la utilidad de las mismas. En concreto, se han incluido expresiones semánticas en la instancia definida en la sección 6.2.2 sobre las propiedades:

- lom:description
- lom:keyword
- lom:coverage
- lom:classification

Inserción de expresiones semánticas en 1.4 Description

En la propiedad "*lom:description*" se pueden incluir expresiones semánticas que ayuden a procesar el lenguaje natural definido en las instancias de tipo cadena "*lom:LangString*" o como aserciones independientes complementarias que aportan mayor grado de semántica.

Por ejemplo en el siguiente texto definido en lenguaje natural sobre una instancia "*lsd_LOR_ExampleLOM_1201627137504*" de tipo "*lom:LangString*"[3]:

> *Metadata is information about an object, be it physical or digital. As the number of objects grows exponentially and our needs for learning expand equally dramatically, the lack of information or metadata about objects*

[3]Ver apéndice sección B.1.2

places a critical and fundamental constraint on our ability to discover, manage and use objects.

Pueden incluirse expresiones definidas por predicados y conceptos OpenCyc según las siguientes expresiones:

```
(isa LOR_ExampleLOM SpecificationDocument)
(isa LOM-Record-Structure LogicalSchema)
(thingSpecified LOR_ExampleLOM LOM-Record-Structure)
(usedIn MakingSomethingAbstractAvailable)
```

También puede utilizarse la propiedad externa "*enable*" de la ontología "*edu*"[4] para describir la utilidad del estándar mencionada en la descripción dada en lenguaje natural. Por ejemplo pueden referenciarse las acciones definidas en la ontología "*LO-Discover*" o "*LO-Manage*".

Traducido según el modelo de inclusión de expresiones semánticas de SLOR, en la base de conocimiento quedarían incluidas con las instancias definidas según la figura 6.8. Los individuales representados en color morado que empiezan con el prefijo "*ocsld*", representan los enlaces semánticos creados sobre predicados OpenCyc, el individual que empieza con el prefijo "*owlsld*" representa la inclusión de una propiedad "*OWL*" utilizada para la descripción del recurso. El resto de individuales que empiezan con el prefijo "*lsd*" indican los "*LangString*" utilizados para la descripción del recurso en lenguaje natural. Cada elemento va seguido de un identificador unívoco definido por el proceso interno de almacenamiento de expresiones semánticas

[4]Ontología utilizada para realizar las pruebas de verificación, localizada en http://slor.sourceforge.net/ontology/edutest.owl

de SLOR (descrito en los métodos del capítulo anterior).

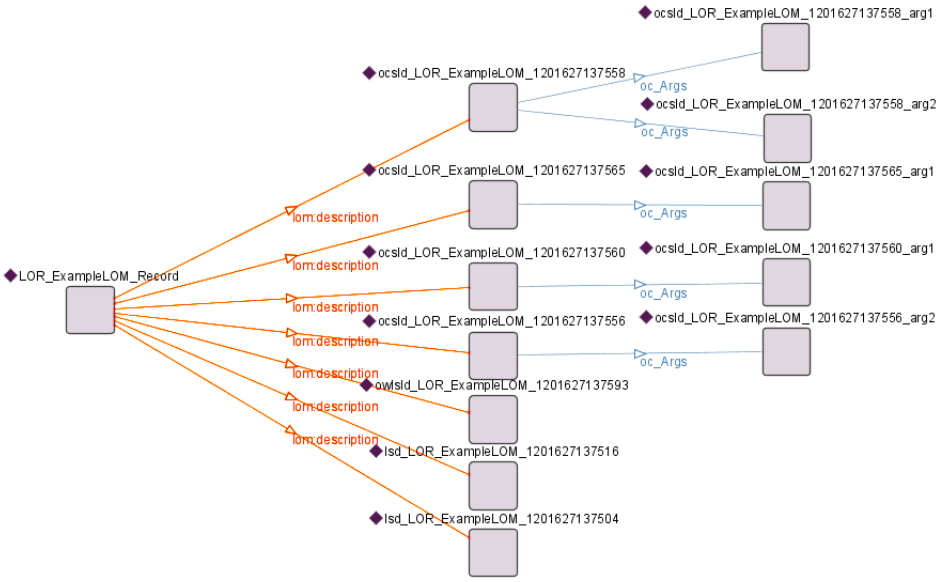

Figura 6.8: Inclusión de expresiones semánticas en la propiedad "*lom:Description*"

A continuación se expone a modo de ejemplo la descripción en sintaxis abreviada [PSHH04] de la aserción OpenCyc "*(isa LORObject SpecificationDocument)*" y "*(edu:enable edu:LO-Discover)*" según el esquema definido en la ontología de SLOR:

```
Individual(LOR_ExampleLOM_Record type(LOMOWL_Record)
   ...
   value(lom:description ocsld_LOR_ExampleLOM_1201627137556)
   value(lom:description owlsld_LOR_ExampleLOM_1201627137593)
   ...)
```

```
Individual(ocsld_LOR_ExampleLOM_1201627137556 type(oc_SemanticLink)
   value(arity "2"^^<http://xsd:int>)
   value(oc_Predicate "isa"^^<xsd:string>)
   value(oc_Args LORID_arg1)
   value(oc_Args ocsld_LOR_ExampleLOM_1201627137556_arg2))

Individual(ocsld_LOR_ExampleLOM_1201627137556_arg2 type(oc_ArgPredicate)
   value(termArg "SpecificationDocument"^^<xsd:string>)
   value(norderArg "2"^^<xsd:int>))

Individual(owlsld_LOR_ExampleLOM_1201627137593 type(owl_SemanticLink)
   value(ontologySchemaLink
     "http://slor.sourceforge.net/ontology/edutest.owl"^^<xsd:anyURI>)
   value(property edu:enable)
   value(property_value edu:LO-Discover))
```

Como puede apreciarse en el código mostrado según la sintaxis abstracta OWL [PSHH04], la propiedad "*lom:description*" puede incluir expresiones semánticas utilizando predicados OpenCyc o propiedades definidas en otras ontologías OWL. En el ejemplo, se ha utilizado el predicado "*isa*" para indicar que el objeto descrito "*LOR_ExampleLOM*" es un documento de una especificación, referenciando al concepto "*oc:SpecificationDocument*".

También se ha incluido en el ejemplo la expresión basada en el vocabulario "*edu*" "*edu:enable edu:LO-Discover*", que permite añadir expresiones comunes utilizadas en la descripción de objetos de aprendizaje. De esta forma se indica que el objeto de aprendizaje permite el descubrimiento de objetos de aprendizaje , así también pueden incluirse otras acciones como "*edu:LO-Manage*" (*"... ability to discover, manage*

and use objects."). Siempre que exista un vocabulario adaptado a la descripción de un tipo de objeto de aprendizaje es posible incluir cualquiera de sus estructuras básicas (propiedades o conceptos) dentro de las propiedades LOMOWL correlacionadas en la sección anterior con el estándar IEEE LOM. El resto de aserciones definidas se han incluido en lenguaje OWL en el apéndice B sección B.1.3.

El segundo objeto insertado "*LOR_DNA*" en el registro de metadatados creado "*LOR_DNA_Record*", posee el siguiente valor en la propiedad "*lom:description*":

> *"DNA from the Beginning is an animated tutorial on DNA, genes and heredity. The science behind each concept is explained using animations, an image gallery, video interviews, problems, biographies, and links. There are three sections, Classical Genetics, Molecules of Genetics and Organization of Genetic Material. Key features are the clear explanations of classical experiments and the excellent photographs of researchers and their labs.´´*

Para facilitar procesos semánticos de computación de ayuda al tratamiento del lenguaje natural, en la propiedad "*lom:description*" del registro de metadatos se puede indicar que el contenido del curso es un tutorial utilizando la expresión OpenCyc:

```
(isa LOR_DNA Tutorial)
```

También puede indicarse sobre el contenido el tipo de temas que trata utilizando el predicado "*containsInformation-Focally*" de OpenCyc, de la siguiente forma:

```
(containsInformation-Focally LOR_DNA DNAMolecule)
(containsInformation-Focally LOR_DNA Gene-HereditaryUnit)
```

La descripción detallada de las expresiones incluidas dentro del repositorio se han adjuntado en el apéndice B sección B.1.3.

Inserción de expresiones semánticas en 1.5 Keyword

En la propiedad "*lom:keyword*" también pueden incluirse conceptos relacionados con otros vocabularios o incluidos en la base de conocimiento OpenCyc. En el primer ejemplo, sobre el registro "*LOR_ExampleLOM_Record*" se pueden incluir en la propiedad "*lom:keyword*" el término OpenCyc "*SpecificationDocument*" o el término "*edu:IEEELOM*" de la ontología "*edu*". En el segundo ejemplo en el registro "*LOR_DNA*" puede incluirse el término "*DNAMolecule*".

Inserción de expresiones semánticas en 1.6 Coverage

Sobre la propiedad "*lom:coverage*" pueden insertarse términos correspondientes a los siguientes rangos especificados: "*oc:GeographicalRegion*", "*oc:TimeInterval*", "*oc:AdministrativeUnit*" o "*slor:OwlLink*". En los objetos de aprendizaje descritos en la sección 6.2.2 puede incluirse en la propiedad "*lom:coverage*" el término "*oc:TheModernAge*" para indicar que el contenido de los objetos descritos pertenece al contexto de la era moderna.

Inserción de expresiones semánticas en 9 Classification

En este apartado de IEEE LOM pueden utilizarse tesauros u otros vocabularios estructurados según un esquema formal. Como ejemplo, se ha incluido en la propiedad "*lom:classification*" del registro de metadatos "*LOR_DNA_Record*" del objeto de aprendizaje "*LOR_DNA*" expuesto en la sección 6.2.2 la siguiente declaración:

```
Individual(classification_LOR_ExampleLOM_1201627603151
   type(lom:Classification)
   value(lom:purpose LOMv1.0-Discipline)
   value(lom:taxon Taxon_1201627606154)
   value(lom:taxon Taxon_1201627606193))
```

```
Individual(Taxon_1201627606154 type(lom:Taxon)
   value(lom:taxonSource nci:NCIThexaurus)
   value(lom:refTaxon nci:Human-Genetics))

Individual(Taxon_1201627606193 type(lom:Taxon)
   value(lom:taxonSource nci:NCIThexaurus)
   value(lom:refTaxon nci:Biology))
```

Como puede apreciarse, se han incluido dos referencias a términos definidos dentro del tesauro NCI (*National Cancer Institute*), "*nci:Human-Genetics*" y "*nci:Biology*" representada su situación parcial en el árbol del tesauro en la figura 6.9.

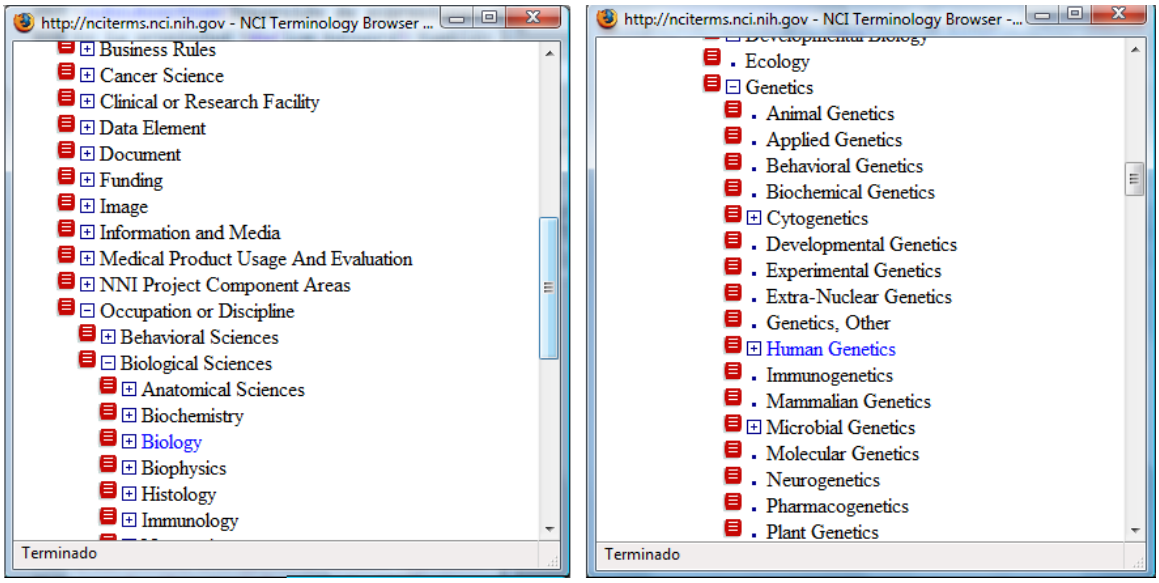

Figura 6.9: Uso del tesauro NCI dentro del campo "*lom:Classification*"

La ventaja ofrecida por este modelo es la de poder procesar parte de los elementos del tesauro dentro de las operaciones de búsqueda y navegación definidas dentro de SLOR. Utilizando términos de un catálogo definido permite determinar sin ambigüedades el tipo de disciplina a la que se refiere un objeto de aprendizaje, e incluso

poder definir los objetivos planteados, las precondiciones o postconditiones, utilizando una ontología específica para tal fin que ayude a establecer de forma automática operaciones basadas en contratos [Alo05].

6.3.2. Operaciones relevantes (E_2A_2)

En esta actividad se han descrito las operaciones relevantes tenidas en cuenta como ventajas frente al modelo de operación de los objetos de aprendizaje almacenados en los repositorios analizados. En concreto, el desarrollo de la actividad se ha centrado en el análisis de las operaciones de búsqueda realizadas por MERLOT, NDSL y MIT OCW que incluye un soporte para SCORM frente a las ventajas aportadas por las operaciones de cómputo semántico realizadas por SLOR.

Se han diseñado 3 casos de prueba preparados para esta actividad de evaluación. El objetivo fue identificar los modos de construcción de restricciones utilizadas por los repositorios seleccionados y evaluarlos frente a los mecanismos semánticos aportados por SLOR.

Caso de prueba 1 - MIT OCW

El formulario de búsqueda avanzada para el repositorio del MIT (ver figura 6.10) permite únicamente definir restricciones sobre la cadena que será consultada de la siguiente forma:

$$search(c_1 \sqcap c_2 \sqcap c_3...) \tag{6.3.1}$$

$$search(c_1 + c_2 + c_3 + ...) \tag{6.3.2}$$

$$search(c_1 \sqcup c_2 \sqcup c_3...) \tag{6.3.3}$$

$$search(\neg c_1 \sqcap \neg c_2 \sqcap \neg c_3...) \tag{6.3.4}$$

La expresión 6.3.5 indica una búsqueda con todas las palabras definidas en un campo de texto. La siguiente expresión 6.3.2 indica la búsqueda con la frase exacta escrita. La expresión 6.3.3 indica como restricción de obtener al menos algún resultado que contenga en sus campos alguna de las cadenas escritas separadas por espacios. Y finalmente la expresión 6.3.4 indica la restricción de no obtener los resultados que contengan las palabras indicadas. Estas restricciones puede combinarse en forma de unión. También se pueden incluir restricciones sobre el tipo de material que contiene el objeto de aprendizaje, como apuntes, videos, laboratorios o exámenes.

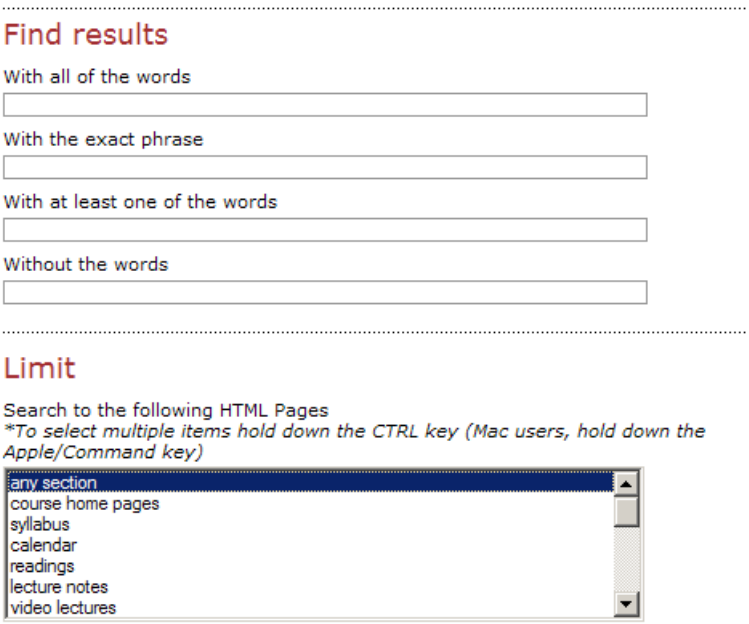

Figura 6.10: Formulario de búsqueda avanzada del repositorio MIT OCW.

Con estas restricciones de búsqueda intentamos construir la siguiente expresión de búsqueda:

Obtener un recurso que explique algún tipo de análisis del ADN.

Lo único que podemos crear es la expresión:

$$search(DNA \sqcap Analysis) \tag{6.3.5}$$

Los resultados obtenidos tras ejecutar la consulta han sido 513. En los 10 primeros aparecen recursos de todo tipo, en concreto el primero de ellos corresponde al resumen de un curso, el segundo a una estructura de un curso, etc. Para filtrar mejor los resultados de la búsqueda se ha incluido una restricción por el tipo "*lecture notes*". En esta segunda búsqueda aparecen 13 resultados. Entre los primeros se encuentran resultados dispersos no encontrados de forma directa ya que en una misma página de tipo "*lecture notes*" aparecen múltiples PDF's que contienen información sobre ADN o algún tipo de análisis.

Como se ha descrito, este tipo de búsquedas entorpece demasiado al usuario cuando se llevan a cabo sobre un vasto conjunto de resultados de búsqueda. Si el repositorio contiene un elevado conjunto de registros de metadatos es necesario obtener de forma sencilla y concreta los resultados deseados. En el caso de SLOR, con simplemente utilizar un vocabulario que evite ambigüedades en la definición de los recursos puede generarse una búsqueda simple por "*lom:keyword*" y "*lom:description*" con una inferencia descendente (TOP-DOWN) para los distintos tipos de análisis. En la lista "*QueryRestrictions*" del formulario de búsqueda tendríamos las expresiones:

```
(and
    ((lom:keyword(nci:DNA_Analysis),(TR_TOPDOWN,-1))),
    (lom:description(oc:containsInformation,2,oc:DNAMolecule))
)
```

Al ejecutar la consulta no sólo se devuelven como resultados aquellos objetos que tengan asociado en la propiedad "*lom:Keyword*" el término "*nci:DNA_Analysis*"

sino que también se recuperan los términos correspondientes a las subclases como "*nci:Allelotype_Analysis*" o "*nci:DNA_Ploidy_Analysis*" entre otros.

Como conclusión tras estas operaciones es posible afirmar que el mecanismo semántico de búsqueda de SLOR aporta un conjunto de operaciones que permite recuperar los resultados de forma más exacta que la definida por el MIT OCW.

Caso de prueba 2 - NSDL

En este caso de prueba se desea obtener un tutorial que explique la molécula del ADN y además esté creado por una universidad.

La búsqueda avanzada en NSDL se realiza utilizando los criterios definidos en el formulario presentado en la figura 6.11. Como puede apreciarse en el formulario aparecen cuatro tipos de restricciones aplicables sobre la búsqueda:

- Por grado académico: se pueden indicar uno o varios niveles según la escala de valores utilizada en el repositorio.
- Por temática: según los acuerdos establecidos por el repositorio NSDL y sus federados, se permite realizar la búsqueda por las siguientes temáticas: educación, salud/medicina, matemáticas, ciencia, estudios sociales, tecnología e ingeniería.
- Por formato: se permite restringir la búsqueda sobre los siguientes formatos: texto, imagen, audio, video, recurso interactivo y datos.
- Por fuente de metadatos: se puede indicar el tipo de fuente de recursos deseada.

Para obtener la búsqueda deseada rellenamos los campos como sigue a continuación:

- Caja de texto: Tutorial DNA.

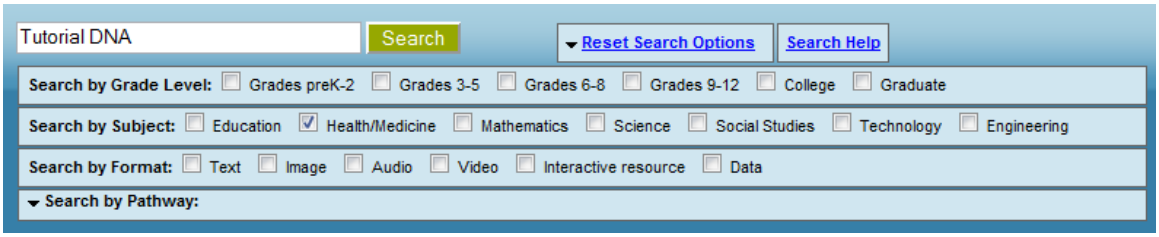

Figura 6.11: Formulario de búsqueda avanzada del repositorio NSDL.

- Subject: Education, Science, Health/Medicine.

Al ejecutar la búsqueda nos aparecen 271 resultados. En los 10 primeros aparecen contenidos muy diferentes entre sí, no correspondientes en algunos casos a lo buscado. En conclusión, la búsqueda ha proporcionado multitud de resultados entre los que aparecen algunas concordancias aproximadas a lo que realmente era nuestra intención de búsqueda.

Con SLOR pueden introducirse, de forma evidente restricciones más directas en los campos deseados. Si se procesa un esquema de metadatos que posee un campo descripción puede incluirse la siguiente expresión semántica de búsqueda:

```
(and
    ((lom:description(oc:isa),2,oc:Tutorial)),
    ((lom:description(oc:containsInformation-Focally,2,oc:DNAMolecule)))
)
```

Cuyos resultados incluyen todos aquellos registros que hagan referencia en el campo descripción al concepto "*Tutorial*". Como se ha visto, además en SLOR pueden elegirse las "propiedades" utilizadas en el modelo conceptual sobre un registro de metadatos, luego este tipo de búsquedas puede configurarse de una forma mucho más aproximada.

Caso de prueba 3 - MERLOT

En este caso de prueba se desean obtener del repositorio MERLOT aquellos objetos que describen algún objeto, hecho o situación que data de la época del Renacimiento.

La búsqueda avanzada en MERLOT se realiza utilizando los campos definidos en el formulario representado en la figura 6.12, en el que no existe ningún campo similar al "*coverage*" de IEEE LOM. Utilizando en su lugar el campo "*description*", o "*keyword*", y realizando la búsqueda con la palabra "*Renaissance*" obtenemos 23 resultados. Estableciendo las categorías "*Humanities*" y "*Humanities/History*" conseguimos reducir el número de resultados, pero se ignoran otros objetos de aprendizaje como por ejemplo los relacionados con los instrumentos de música.

En cambio, con los mecanismos semánticos de búsqueda aportados por SLOR se puede construir una expresión semántica que permita eliminar este problema utilizando un esquema basado como el descrito en la sección 6.2.2 de la siguiente forma:

```
(lom:coverage (oc:TheReinaissance))
```

Otro tipo de búsqueda a la que MERLOT no puede responder de forma directa es la obtención de aquellos objetos de aprendizaje que pertenecen al contexto de una guerra civil y describen alguna batalla militar:

```
(and
    ((lom:description
    (oc:containsInformation-Focally,2,oc:MilitaryBattle)
    (TR_TOPDOWN,-1))),
    (lom:coverage (oc:CivilWar),(TR_TOPDOWN,-1))
)
```

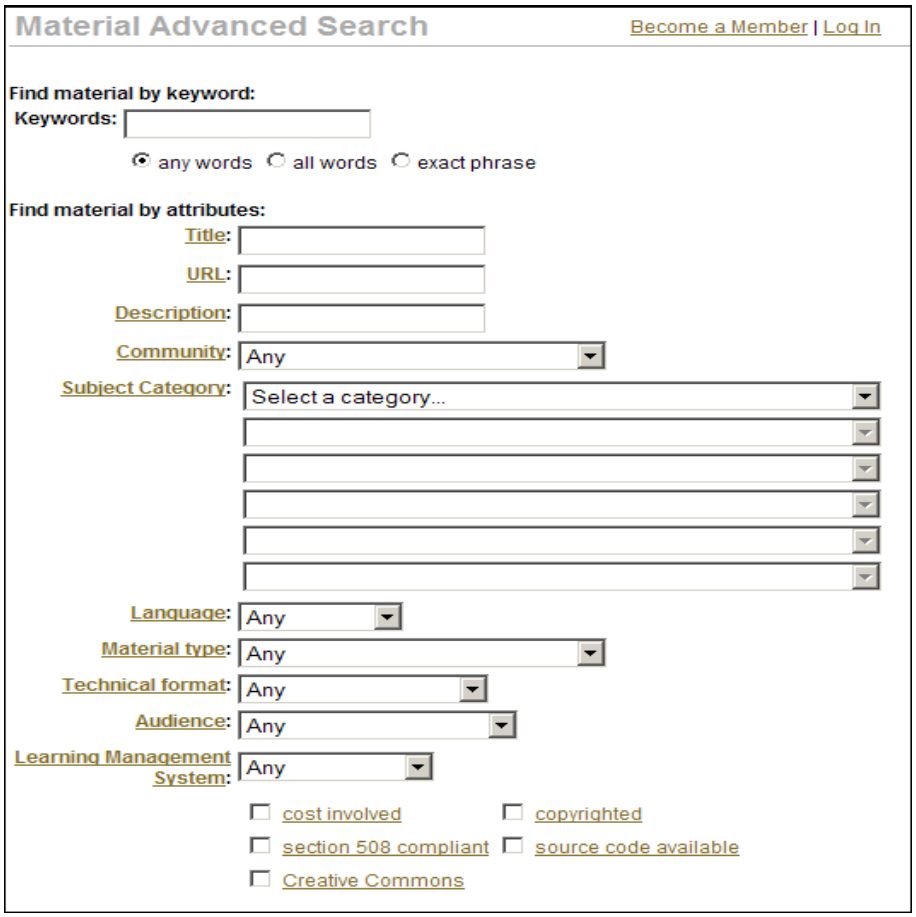

Figura 6.12: Formulario de búsqueda avanzada del repositorio MERLOT.

Estos métodos permiten construir expresiones complejas de búsqueda que eliminan ambigüedades gracias al uso de ontologías específicas utilizadas en la definición de los registros de metadatos.

Conclusiones del análisis

Tras analizar los mecanismos de búsqueda en los casos de prueba descritos, podemos concluir que los repositorios actuales ofrecen mecanismos de búsqueda que no permiten concretar de forma directa los resultados deseados. Esto es debido a que el proceso de ejecución de la consulta realizada es por el análisis sobre el texto plano de

los campos de los registros de metadatos.

El uso de ontologías permite crear expresiones que pueden ser insertadas sobre SLOR, ayudando al proceso de la semántica del lenguaje natural. En el caso de las ambigüedades puede explicarse con un ejemplo. Si quisiéramos localizar objetos de aprendizaje en el contexto geográfico "*La Pampa*", región del Sur de Argentina, utilizando simplemente dicha palabra, surgen multitud de sitios en el mundo que hacen referencia a este nombre, como es el caso de la región de Texas, o el río Pampa de Brasil o la Pampa Grande, región geográfica de Bolivia, entre otros. Con el término tgn:LaPampa-1001375 se referencia directamente al concepto que evoca la provincia "*La Pampa*" de Argentina, y de este modo se elimina todo tipo de ambigüedad.

Los métodos semánticos presentados son una alternativa eficaz a los métodos de búsqueda tradicionales, y permiten localizar con un mayor grado de éxito los resultados deseados, aprovechando el uso de los mecanismos de inferencia proporcionados por SLOR.

6.4. Flexibilidad (E_3)

La evaluación de la flexibilidad aborda la verificación de que los mecanismos semánticos propuestos permiten representar cualquier tipo de conceptualización del término "*objeto de aprendizaje*".

Para conseguir evaluar la flexibilidad se han definido varios casos de prueba, diseñados con el objetivo de mostrar la viabilidad de la representación de cualquier tipo de registro de metadatos almacenado en un repositorio actual de objetos de aprendizaje. Además se pretende verificar que la implementación del modelo basada en la conceptualización del término "*objeto de aprendizaje*" cubre las definiciones

descritas y otras nuevas.

6.4.1. Demostración de la inserción de conceptualizaciones de objetos de aprendizaje no conformes con el estándar LOM (E_3A_1)

Para demostrar que el sistema cubre el requisito de flexibilidad, se han establecido una serie de casos de prueba que consisten en intentar representar en el nuevo modelo conceptual del repositorio, los mismos registros de metadatos de objetos de aprendizaje almacenados en los repositorios descritos en la sección 2.5.2 del capítulo 2. Los casos de prueba utilizados han sido los siguientes:

1. Representación un registro de metadatos [5] almacenado en MERLOT.

2. Representación un registro de metadatos [6] del tipo recurso electrónico almacenado en el repositorio NSDL.

Caso de prueba 1

Para poder representar el esquema de registros de metadatos utilizado por MERLOT dentro del nuevo esquema de almacenamiento semántico definido para SLOR, es necesario generar las clases que contengan las propiedades que permitan describirlo. Para ello, primero se ha realizado un análisis de los campos utilizados en los registros de metadatos de MERLOT. Seguidamente, y con la información obtenida, se ha realizado un modelo conceptual de clases para incluir la conceptualización de un registro de objeto de aprendizaje almacenado por MERLOT. Finalmente, se presentan los resultados tras la inserción del registro de objeto de aprendizaje analizado.

[5] http://www.merlot.org/merlot/viewMaterial.htm?id=89142

[6] http://nsdl.org/search/?q=heart&verb=Search&s=0&n=10&item_num=0&identifier=http%3A%2F%2Fwww.nhlbi.nih.gov%2Fhealth%2Fdci%2FDiseases%2Fhhw%2Fhhw_whatis.html

Los registros de metadatos almacenados en MERLOT utilizan los campos enunciados en las tablas 6.2 y 6.3. La primera tabla describe el cuerpo del registro principal y la segunda tabla contiene los campos utilizados en la descripción de la información complementaria del registro de metadatos.

Campo	**Descripción**
0. Identifier	Identificador interno del registro.
1. Title	Título del objeto de aprendizaje.
2. Material Type	Tipo de material (taxonomía).
3. Technical Format	Descripción del formato.
4. Location	Localización del contenido digital.
5. Date Added	Fecha de inserción del registro.
6. Date Modified	Fecha de modificación del registro.
7. Author*	Autor del recurso.
8. Submitter	Creador del registro de metadatos.
9. Browse in Categories	Categorías del recurso (taxonomía).
10. Descripción	Descripción del recurso.
i. Information	Registro de información complementaria.

Tabla 6.2: Campos de un registro de metadatos del repositorio MERLOT

El campo "*Author*" contiene la información sobre el autor del recurso, aunque no queda especificado formalmente el contenido el campo, incluye la información referente a la dirección de correo electrónico que en este caso de prueba se referencia con el código 7.1 y el centro educativo asociado con referencia 7.2 que identifica a una universidad, instituto o cualquier otro tipo de centro de docencia o investigación.

Principalmente para albergar un registro de metadatos de MERLOT, en función del tipo de objeto de aprendizaje se puede utilizar cualquier conceptualización definida en el modelo de McGreal [McG04]. En este ejemplo se ha creado la conceptualización

Campo	Descripción
i.1 Primary Audience	Audiencia principal a la que va dirigido.
i.2 Technical Requirements	Requisitos técnicos.
i.3 Language	Lenguaje.
i.4 Copyright	Derechos de autor.
i.5 Source Code Available	Código fuente disponible.
i.6 Cost Involved	Coste involucrado.
i.7 Creative Commons	Indicador de licencia Creative Commons.

Tabla 6.3: Campos de un registro de metadatos del repositorio MERLOT

"*LOR_Merlot*" subsumido del término "*LearningObject_AsAnyThingDigital*" al ser un recurso digital con posibilidad de ser utilizado en alguna actividad de aprendizaje de las matemáticas.

En esta conceptualización se pueden realizar las primeras correspondencias con los campos de un registro de Merlot, al ser un registro digital "*oc:ComputerFileCopy*" cada individualización puede especificar en la propiedad heredada "*fileFromURI*" la dirección de ubicación del objeto de aprendizaje. Esta propiedad se corresponde con el campo "*3.Location*" de un registro MERLOT. Por otro lado el título (propiedad "*title*") se corresponde con la propiedad (0.Title) del registro de MERLOT mientras que el identificador (propiedad "*Identifier*") lo hace con el código de identificación interno dentro del repositorio de MERLOT.

En la tabla 6.4 se muestra a modo de resumen la correspondencia de propiedades de la clase "*LOR_Merlot*" con los campos de un registro de metadatos de MERLOT.

Para representar el contenido restante de la información almacenado en un registro de MERLOT se ha definido la clase "*Merlot_Record*", que hereda de la clase

Propiedad	**Rango**	**Correspondencia - MERLOT**
(dp) Identifier	xs:string	0. Identifier
(dp) Title	xs:date	1. Title
(dp) oc:fileFormURI	xs:anyURI	4. Location
(op) hasAssociated MetadataRecord	slor:LearningMetadataRecord	Registro de metadatos asociado al descriptor especificado bajo la definición de la clase "*LearningObject-AsAnyThingDigital*" (ver tabla 6.5)

Tabla 6.4: Correspondencia de las propiedades definidas para la conceptualización LOR_Merlot con las propiedades de un registro MERLOT

"*slor: LearningMetadataRecord*" del modelo de SLOR. Las propiedades asociadas se especifican en la tabla 6.5.

La propiedades señaladas con un asterisco indican que el rango ha sido especificado con un esquema reducido, útil para esta prueba de representación. Si no se conoce el esquema interno del repositorio para este tipo de conceptos, puede definirse la propiedad como "*owl:DataTypeProperty*" y el rango como una cadena "*xs:string*" con objeto de simplificar el tratamiento y la representación.

La clase "*MerlotCategories*" representa el árbol de categorías de MERLOT. Dentro del esquema OWL podría importarse de forma completa si existiese alguna versión pública procesable por una aplicación. Por ello en la tabla se ha especificado con el valor "*(i)*". En esta prueba se ha codificado una taxonomía simple con los elementos indicados en el registro de metadatos. En la figura 6.13 se muestra el árbol

Propiedad	**Rango**	**Correspondencia Merlot**
(op)category*	MerlotCategories(i)	9. Browse in Categories
(op)material_Type*	MerlotMaterialType	2.Material Type
(op)format*	TechnicalFormal	3.Technical Formal
(op)lorauthor	Merlot_LOCreator	7.Author
(op)description	slor:descriptiveProperty	10.Description
(op)information	Merlot_InformationDescriptor	Información extendida (ver tabla 6.6).
(op)metametadata	Merlot_MetaMetadataDescriptor	Información sobre el registro de metadatos (ver tabla 6.7).
(op)review	Merlot_Review	Registro que contiene información sobre la revisión de un usuario de MERLOT.

Tabla 6.5: Correspondencia de los campos del registro de metadatos slor:Merlot_Record con el esquema seguido por MERLOT

de conceptos simplificado utilizado para establecer la categoría de un registro de metadatos de tipo "*Merlot_Record*".

De forma similar, la clase "*MerlotMaterialType*" representa todos los posibles materiales almacenados en Merlot. Por otro lado, la clase "*TechnicalFormal*" ha sido creada para representar el formato del objeto de aprendizaje. Estas clases pueden ser representadas por conceptos de otras ontologías (en la sección 5.3.1 se detalla este tipo de aproximación, que no forma parte del objetivo del presente caso de prueba).

La clase "*Merlot_LOCreator*" representa la información que contiene el repositorio MERLOT en sus registros de metadatos sobre el autor de un objeto de aprendizaje.

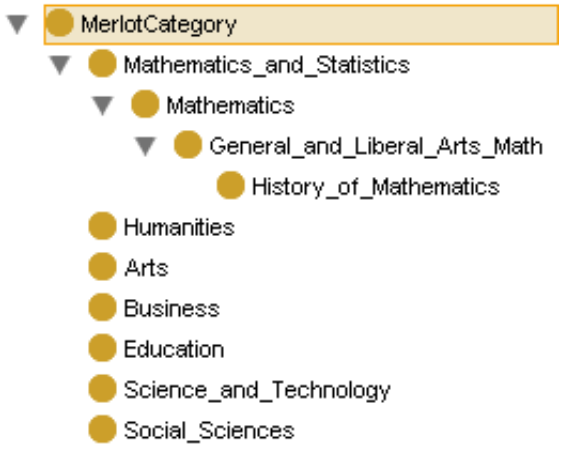

Figura 6.13: Jerarquía de la clase "*MerlotCategories*"

En general se guarda el nombre del autor (propiedad "*LORauthorName*") y la dirección de e-mail (propiedad "*email*"), si bien en algunos registros también se especifica el centro de enseñanza al que pertenece.

Propiedad	**Rango**	**Correspondencia - Merlot**
(dp)LORauthorName	xs:string	7.Author
(dp)email	xs:anyURI	7.1. Email
(dp)education_center	xs:string	7.2. Centro educador

La clase "*Merlot_InformationDescriptor*" incluye la información extendida del registro de metadatos. En la tabla 6.6 se describen las propiedades que contiene.

Finalmente, para completar el modelo del registro de metadatos "*Merlot_Record*", se ha creado clase "*Merlot_MetaMetadataDescriptor*" que representa la información del propio registro de metadatos (ver tabla 6.7), en el que se especifica el creador

Propiedad	Rango	Correspondencia - MERLOT
(dp) primary_audience	xs:string	i.1 Primary Audience
(dp) technical_Requeriments	xs:string	i.2 Technical Requirements
(dp) language	oc:LivingLanguage	i.3 Language
(dp) copyright	xs:boolean	i.4 Copyright
(dp) sourceCode	xs:boolean	i.5 Source Code Aviable
(dp) Cost Involved	xs:boolean	i.6 Cost Involved
(dp) Creative Commons	xs:boolean	i.7 Creative Commons

Tabla 6.6: Correpondencia de los campos de un registro de metadatos MERLOT con el descriptor de información

(propiedad "*submitter*"), la fecha en la que se incluyó el registro dentro del repositorio MERLOT (propiedad "*date_Added*") y la última fecha de modificación (propiedad "*date_Modified*").

Propiedad	Rango	Correspondencia - MERLOT
(op)submitter	merlot:User	8.Submitter
(dp)date_Added	xs:date	5.Date Added
(dp)date_Modified	xs:date	6.Date Modified

Tabla 6.7: Clase slor:Merlot_MetaMetaDataDescriptor

A modo de resumen se presenta en la figura 6.14 el esquema de las relaciones entre las propiedades del registro "*Merlot_Record*". La sección B.2 del apéndice B contiene el código OWL de la definición de la clase "*Merlot_Record*" y todas sus propiedades y clases relacionadas.

Una vez definido el esquema, se modifica el modelo del repositorio introduciendo los términos y propiedades anteriormente comentados. A partir de este momento ya pueden incluirse nuevos registros de objetos de aprendizaje que puedan tener asociado

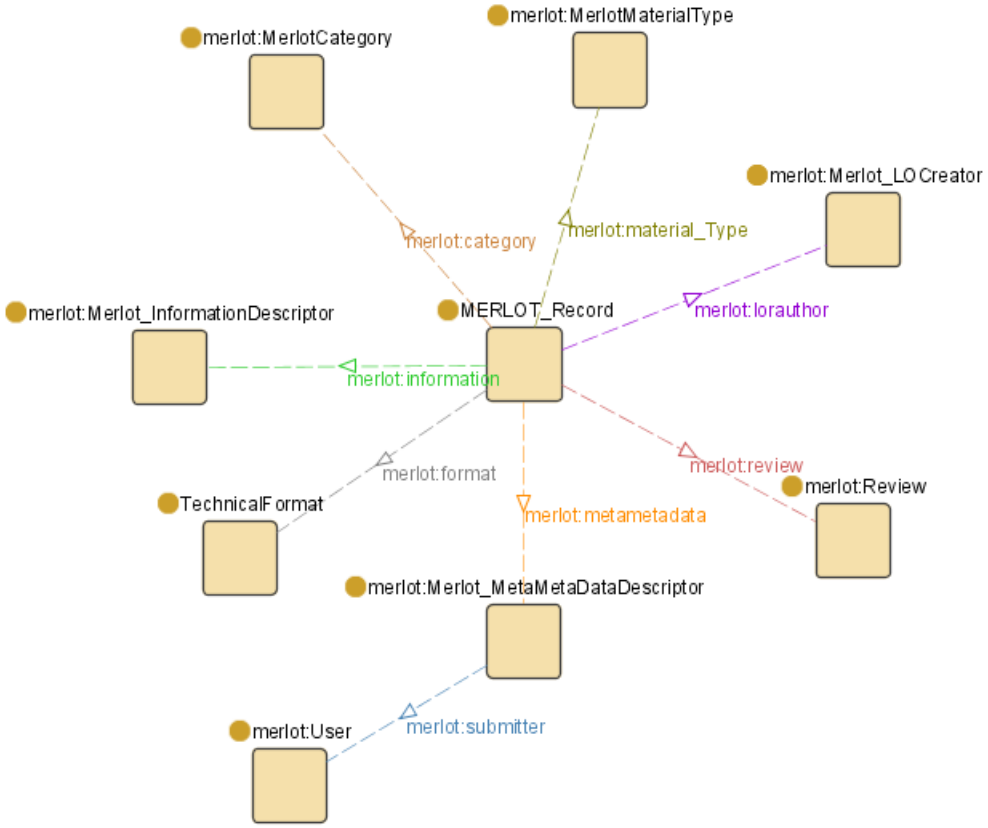

Figura 6.14: Registro de metadatos "*Merlot_Record*"

un registro de metadatos de tipo "*Merlot_Record*".

Respecto a la programación, se ha de actualizar el esquema de clases e interfaces, también es necesario actualizar la factoría "*OntoSlorFactory*". Una vez que se realicen estas operaciones, se ha de codificar un módulo con las funciones básicas de gestión (inserción, borrado, modificación, etcétera) y las interfaces que permitan al usuario manipular el contenido del repositorio con el uso de estos formatos.

El diseño de la arquitectura de SLOR está basado en la ampliación de funcionalidad por módulos. De esta forma la inclusión de un nuevo esquema supone la creación

de un módulo junto con las interfaces que den soporte a la inserción y administración del registro. Como todo el núcleo de administración del almacenamiento persistente está creado, sólo se ha de contemplar el esfuerzo en la creación de las interfaces y la codificación de sus métodos.

Para verificar el cumplimiento de requisitos mínimos en este caso de prueba, se procede a la inserción del registro del objeto de aprendizaje estudiado "*Merlot_Record*" en el repositorio SLOR. Como puede apreciarse en el código del apéndice B en la sección B.2, el sistema da soporte a que cualquier descriptor de registro de metadatos de un objeto de aprendizaje de tipo "*LearningObject_AsAnyThing*" pueda incluir asociado un registro de metadatos tipo "*Merlot_Record*".

Caso de prueba 2

Al igual que en el caso de prueba anterior, se ha comprobado que el repositorio NDSL utiliza un conjunto fijo de campos diseñados para describir los registros almacenados. Estos campos no han sido publicados en ningún esquema descrito bajo un lenguaje formal (XSD, RDF u OWL), por ello en la tabla 6.8 se han identificado los campos para el estudio de evaluación presentado en este caso de prueba.

Con los campos identificados en la tabla 6.8 se puede definir el modelo de metadatos en OWL utilizando dos conceptualizaciones, "*LOR_NDSL*" y "*NDSL_Record*".

La primera de ellas, "*LOR_NDSL*", es un término basado en la conceptualización "*LearningObject_AsAnyThingDigital*" de McGreal. Representa un objeto de aprendizaje descrito bajo el modelo conceptual NDSL. En la tabla 6.9 se establecen las relaciones entre las propiedades OWL y los campos NDSL identificados.

Campo	**Descripción**
0. URL	Identificador y localizador del registro.
1. Title	Título.
2. Description	Descripción.
3. Collections	Repositorio de metadatos de origen.
4. Archives	Ficheros digitales (contenido).
5. Resource Type	Tipo de recurso.
6. Keywords	Palabras clave.
7. Publisher	Proveedor del recurso.
8. Intended Audience	Tipo de audiencia a la que va dirigido.
9. Language	Lenguaje principal utilizado en el contenido del recurso.

Tabla 6.8: Campos de un registro de metadatos del repositorio NDSL

La segunda conceptualización "*NDSL_Record*" representa un registro de metadatos conforme a los campos identificados en la tabla 6.10. El término "*NDSLAudience*" se refiere a cualquier colectivo de personas o individuos clasificados según una taxonomía interna no publicada del registro NDSL. Internamente en SLOR las instancias "*NDSLAudience*" pueden ser correlacionadas con colecciones de tipo OpenCyc o con cualquier otro tipo de término.

La propiedad "*has_audience*" tiene como propiedades derivadas "*dc:author*" y "*dc:publisher*" de Dublin Core, debido a que en los registros NDSL se incluyen los valores asociados a los autores, editores o proveedores del contenido del objeto de aprendizaje. El término "*oc:LivingLanguage*" es un término de la base de conocimiento OpenCyc que define cualquier tipo de lenguaje humano vigente en la actualidad.

Propiedad	**Rango**	**Correspondencia - NDSL**
(dp) Identifier	xs:string	0. ID
(dp) Title	xs:date	1. Title
(dp) oc:fileFormURI	xs:anyURI	4. Archives
(op) hasAssociated MetadataRecord	slor:LearningMetadataRecord	Registro de metadatos asociado al descriptor especificado bajo la definición de la clase "*LOR_NDSL*" (ver tabla 6.10)

Tabla 6.9: Correspondencia de las propiedades definidas para la conceptualización LOR_NDSL con las propiedades de un registro NDSL

Propiedad	**Rango**	**Correspondencia - MERLOT**
(op) description	slor:descriptiveProperty	2. Descripción
(op) resourceType	NDSLResourceType (i)	5. Resource Type
(op) keywords	slor:conceptualLink	6. Keywords
(op) publisher	NDSLPublisher	7. Publisher
(op) has_audience*	xs:string	8. Intended Audience
(op) language	oc:LivingLanguage	9. Language

Tabla 6.10: Correspondencia de las propiedades definidas para la conceptualización "*LOR_NDSL*" con las propiedades de un registro NDSL.

6.4.2. Cohexistencia de modelos, soporte de perfiles IEEE LOM (E_3A_2)

SLOR puede manejar múltiples modelos conceptuales como los definidos previamente, dando soporte al almacenamiento de registros diferentes. Existe una variante, la de definir restricciones sobre los esquemas definidos. Para demostrar esta característica hemos elegido los perfiles de aplicación IEEE LOM que dictaminan unas normas de uso específico del estandar LOM. Este tipo de restricciones pueden ser incluidas gracias al soporte semántico de SLOR, que permite la descripción de registros y propiedades con lógica de descripciones.

En esta sección se demuestra la inclusión del perfil "*UK LOM Core*" definido por CETIS[7] dentro de SLOR. Una vez definido el esquema pueden incluirse y procesarse registros conformes con el perfil.

Para ello se define un nuevo término "*LOR_LOMOWLCORE*" que define las restricciones de los objetos de aprendizaje conformes con el perfil LOM.

$$
\begin{aligned}
&LOR\_LOMCOREOWL \sqsubseteq LOR\_LOMOWL \\
&\quad hasAssociatedMetadataRecord\,\exists(LOMCOREOWL\_Record) \\
&\quad card(oc : fileFROMURI) = 1 \\
&\quad card(slor : title) \geq 1 \\
&\quad oc : usedIn\,\exists(oc : Learning)
\end{aligned}
$$

El registro de metadatos "*LOMCOREOWL_Record*" posee la siguiente definición en

[7] http://zope.cetis.ac.uk/profiles/uklomcore

lógica de descripciones:

$$LOMCOREOWL\_Record \sqsubseteq LOMOWL\_Record$$

$$lom : identifier\ \exists(lom : Identifier)$$

$$lom : language\ \exists(oc : HumanLanguage)$$

$$lom : description\ \exists(slor : descriptiveProperty)$$

$$lom : contributeLifeCycle\ \exists(lom : Contribute)$$

$$lom : metaMetadata\ \exists(lom : MetaMetaData\_Record)$$

$$lom : rightsDescription\ \exists(lom : LangString)$$

$$lom : copyrightAndOtherRestrictions\ \exists(lom : Vocabulary)$$

El uso de un razonador integrado en SLOR nos permite identificar aquellos registros de metadatos que son conformes con el perfil UK LOM Core, de tal forma que pueden realizarse operaciones de obtención de estos registros para uso en entornos distribuidos. Por ejemplo, es posible obtener aquellos registros conformes con el perfil solicitado por un agente software externo.

6.5. Resumen

En la tabla 6.11 se presenta de forma contrastada en la primera columna los objetivos planteados en el primer capítulo y en la segunda columna las actividades de evaluación que han verificado el cumplimiento del objetivo.

Objetivo	**Actividades de evaluación**
(O1) Estudiar las capacidades de representación de los repositorios de objetos de aprendizaje existentes en relación a su uso por parte de sistemas software.	Estudio del estado de la cuestión, particularmente punto 2.5.
(O2.1) Proponer una representación para el almacenamiento semántico de objetos de aprendizaje que dé cabida a objetos de aprendizaje concebidos bajo diferentes conceptualizaciones del término "learning object".	E1 / E3
(O2.2) Proponer una representación para el almacenamiento semántico de objetos de aprendizaje que cubra los estándares actuales de metadatos (particularmente IEEE LOM).	E1 / E2
(O2.3) Proponer una representación para el almacenamiento semántico de objetos de aprendizaje que introduzca semántica computacional en las descripciones de metadatos de los objetos de aprendizaje.	E3 / E4
(O3) Diseñar una arquitectura para la gestión flexible de objetos de aprendizaje que utilice la representación propuesta en el objetivo O2. Dicha arquitectura estará orientada a permitir funcionalidades extendidas con respecto a las que proporcionan los repositorios actuales.	Diseño presentado en el capítulo 5. Validación basada en la verificación funcional y aceptacion del prototipo.

Tabla 6.11: Tabla de verificación del cumplimiento de los objetivos

Capítulo 7

Conclusiones y trabajo futuro

El modo de dar una vez en el clavo es dar cien veces en la herradura.

Miguel de Unamuno

A lo largo del presente capítulo se exponen las conclusiones derivadas de la investigación, así como las líneas de trabajo futuro.

7.1. Conclusiones

La principal aportación del trabajo de investigación llevado a cabo es el diseño de un repositorio semántico que da cabida a las diferentes conceptualizaciones de los objetos de aprendizaje. Sobre el estudio de los problemas existentes en los repositorios de objetos de aprendizaje (presentado en el capítulo 3) pueden apreciarse, en el desarrollo de la solución de los problemas (descrita en los capítulos 4 y 5), las siguientes aportaciones relevantes:

- Estudio de los problemas existentes en los repositorios de objetos de aprendizaje. Se han analizado las carencias de los repositorios actuales y se han centrado los objetivos de esta investigación. La flexibilidad en el almacenamiento, desempeña un papel muy importante dada la heterogeneidad de los registros de metadatos almacenados. Una de las conclusiones más relevantes del estudio realizado es

el hecho de que no es posible realizar ningún tipo de inferencia sobre la información del repositorio por parte de agentes software externos, al no utilizarse ninguna especificación ni modelo que soporte las diferentes conceptualizaciones existentes.

- Representación semántica de los metadatos. El modelo formal utilizado en el repositorio semántico proporciona la unificación de las distintas conceptualizaciones existentes de los objetos de aprendizaje. Este modelo está basado en las especificaciones formales de la Web semántica, para garantizar la posibilidad de comunicación con agentes software externos, capaces de procesar y realizar inferencias sobre el conocimiento existente en el repositorio.

- Representación semántica de la metainformación. Se han definido mecanismos para la inclusión de expresiones semánticas de forma complementaria a la información existente en los registros de metadatos (descritas en los capítulos 4 y 5). Esta característica ha permitido la creación de un mecanismo de interpretación semántica, con lo que se han podido definir nuevos tipos de operaciones de búsqueda que ofrecen mejores resultados a los ya existentes. Así por ejemplo, se pueden recuperar con exactitud objetos de aprendizaje que incluyen contenidos específicamente encuadrados en una época histórica o en un determinado lugar geográfico, ya que existen ontologías que describen épocas y otras que describen lugares geográficos. Es posible, incluso, realizar inferencias más potentes basadas en una aproximación contextual, tales como buscar objetos que se encuentran dentro de otros, gracias al conocimiento general aportado por las ontologías de alto nivel [Len95].

- Comunicación e interoperabilidad. Tanto un sistema LMS externo que procese la información de empaquetado según la especificación IMS CP [Smy03] como un agente software que procese los campos de los metadatos descritos en LOM

[LTS02], pueden operar con un objeto de aprendizaje si este posee registros de metadatos conformes con las diferentes conceptualizaciones implementadas en SLOR.

- Arquitectura técnica. Se ha realizado el diseño completo de la arquitectura del repositorio, descrito en el capítulo 5, implementado un prototipo para la verificación funcional de los mecanismos semánticos propuestos en la investigación.

7.2. Líneas de trabajo futuras

Los resultados obtenidos en la elaboración de esta investigación promueven el desarrollo de una nueva generación de repositorios de objetos de aprendizaje, con altas prestaciones basadas en el intercambio y reutilización de los contenidos didácticos. El procesamiento autónomo de la metainformación por parte de los agentes software externos producirá nuevas herramientas de construcción didáctica, cuyas capacidades de búsqueda, integración y combinación de los objetos de aprendizaje serán beneficiadas gracias al modelo formal descrito. Parten como trabajo futuro las siguientes líneas de investigación:

- Progreso de diseño e implementación - SLOR v1.0: el trabajo no finaliza en un prototipo. Se desea extender las funcionalidades del mismo, dejando una versión web definitiva de acceso público en internet[1]. También se publicará una versión definitiva de la conversión IEEE LOM a OWL[2] de distribución libre. La evolución del repositorio se ve beneficiado por el proyecto europeo *Organic Edunet* [3], ya que parte de las ideas presentadas han sido estudiadas como futura contribución a parte del diseño del proyecto [MSE+08].

[1] http://slor.sourceforge.net
[2] http://slor.sourceforge.net/ontology/lom.owl
[3] http://www.organic-edunet.eu/

- Generación automática de interfaces: los trabajos realizados en esta línea de investigación por los creadores de Protégé o Reload, sugieren la posibilidad de utilizar los esquemas de metadatos como base para la generación de la interfaz del repositorio. Gracias a esta línea de trabajo se proporcionará un diseño mucho más extensible del repositorio, permitiendo así la inserción de nuevas conceptualizaciones ahorrando esfuerzo en programación.

- Comunicación distribuida: basándonos en los nuevos trabajos existentes sobre las arquitecturas orientadas a servicio (SOA) que hacen uso para su implantación de un *Enterprise Service Bus*, se ha definido una posible línea de desarrollo e investigación para la creación de un conector ESB para el repositorio semántico de objetos de aprendizaje compatible con IMS-DRI [IMS03b]. Derivado de esta línea de investigación, de forma colateral se realizará un estudio sobre un servicio ejecutor de consultas distribuidas, encargado de recibir las solicitudes de búsqueda de objetos de aprendizaje en un espacio de meta-información distribuido interconectado por un ESB. También en este escenario entra la interacción con las herramientas de autor.

- Extracción automática de la semántica tácita de la metainformación: la descripción de recursos planteada requiere un coste adicional de cara al creador de los metadatos. Por lo tanto, el diseño de un software inteligente capaz de reconocer los conceptos descriptivos utilizados, así como sus relaciones, permitirá el cumplimiento del modelo formal ahorrando parte de los costes adicionales de creación.

- Descripción de recursos: los mecanismos semánticos presentados ponen de relieve nuevas operaciones que permiten procesar los metadatos de los recursos, no solo en el campo de los objetos de aprendizaje sino también en otras áreas de

conocimiento. Un ejemplo se da en el ámbito de la comunicación distribuida, que utiliza estandares como UDDI para la descripción y localización de servicios; los mecanismos presentados en la presente tesis sugieren la posibilidad de incluir un mecanismo de catalogación más rico semánticamente que los aportados por NAICS[4] o UNSPSC[5] para proporcionar nuevos mecanismos de búsqueda y composición automática de aplicaciones. Otros ejemplos se dan en los recursos de arte, como fotografías, cuadros, textos literarios, música; cada uno de los recursos menciados se describe con un vocabulario específico, que como ya se ha visto en el desarrollo de esta investigación, pueden ser almacenados en un respositorio de forma flexible y estructurada utilizando ontologías específicas para cada recurso.

7.3. Publicaciones derivadas de la investigación

Durante el desarrollo de esta investigación se han realizado las siguientes publicaciones:

- [SnSSA05] *Resource descriptions for semantic element repositories: addressing flexibility*, publicado en la tercera conferencia internacional de tecnologías multimedia, información y comunicación en la educación (MICTE - 2005).
- [SSAnS05] *Semantic learning object repositories: flexibility and implementation issues*, publicado en la tercera conferencia internacional de tecnologías multimedia, información y comunicación en la educación (MICTE - 2005).
- [nSGSAS05] *A semantic lifecycle approach to learning object repositories. Presentado en Lisboa*, en el congreso internacional de eLearning y Telecomunicaciones (ELETE - 2005), publicado por el IEEE.

[4] http://www.census.gov/epcd/www/naics.html
[5] http://www.unspsc.org/

Estas ideas han ido desarrollándose a lo largo de estos años de investigación reflejando las siguientes publicaciones en revistas internacionales:

- [GS05a] *A brief reflection on the ontological definition of "learning objects´´*, publicado en la revista *Learning Objects and Learning Designs.*

- [SGSA06] *Problemas de Almacenamiento e inferencia sobre grandes conjuntos de metadatos en un repositorio semántico de objetos de aprendizaje.*, publicado en el III Simposio Plurisciplinar sobre objetos y diseños de aprendizaje, Oviedo (SPDECE-2006).

- [SGSA07] *Semantic Learning Object Repositories*, publicado en la revista *International Journal of Continuing Engineering Education and Life-Long Learning (IJCEELL).*

También se han realizado estudios sobre el modelo de procesamiento de los marcos de trabajo semánticos con grandes volúmenes de metadatos en la siguientes publicaciones en revistas y congresos internacionales:

- [SJJ06] *Semantic Web Servers: A new approach to query on big datasets of metadata*, publicado en la revista WSEAS Transactions on Computers (WSEAS-11/2006).

- [SGRJ06a] *A new approach to dynamic load balancing across multimedia servers*, publicado en la revista WSEAS Transactions on Computers (WSEAS - 11/2006).

- [SGM06] *Mejora de la calidad y servicio en los repositorios de objetos de aprendizaje multimedia*, publicado en el III Simposio Plurisciplinar sobre objetos y diseños de aprendizaje, Oviedo (SPDECE-2006).

- [SGSJ06] *An analysis about the RDF storage problems: Towards Semantic Web Servers*, publicado en el IV Simposio Internacional de Sistemas de Información e Ingeniería del Software en la Sociedad del Conocimiento. (SISOFT-2006).

- [SGRJ06b] *Open load multimedia server balancing*, publicado en el IV Simposio Internacional de Sistemas de Información e Ingeniería del Software en la Sociedad del Conocimiento. (SISOFT-2006).

De forma extendida con los resultados de las investigaciones realizadas se ha publicado el capítulo de libro -*Designing Flexible Learning Object Repositories: Balancing Flexibility and Delegation in Ontological Characterizations* [SASnS06], en el libro *Principles and Practices of the Effective Use of Learning Objects* editado por Informing Science Institute, 2006.

Para completar el desarrollo y el contraste de la ideas presentadas se realizó una estancia de investigación con el equipo del Dr. Miltiadis Lytras en Atenas (Grecia) desde el mes de Junio hasta finales del mes de Septiembre del año 2006. La estancia sirvió para aportar y madurar las ideas tratadas en la presente tesis.

En el marco del proyecto interno *Semantic Web technology-based standardized e-learning systems (ELSEM) UAHPI2005- 070* financiado por la Universidad de Alcalá, se participó como colaborador durante el año 2006 realizando un informé técnico sobre las características relevantes de implementación de un repositorio semántico de objetos de aprendizaje.

De forma colateral se han generado publicaciones relacionadas con las tecnologías de la Web semántica y el ámbito de los sistemas de e-learning, en concreto:

- [SG05] *Sistema multiagente inteligente para la planificación organizada del estudio de un alumno*, publicado en el III Simposio Internacional de Sistemas de Información e Ingeniería del Software en la Sociedad del Conocimiento(SISOFT - 2005).

- [GS05b] *Autenticidad basada en la Web semántica de confianza*, publicado en el IV Congreso Internacional de Auditoria y Seguridad de la Información(CIASI - 2005).

- [WJS05] *Modificación de la intención semántica de relatos utilizando ontologías y técnicas evolutivas*, publicado en el cuarto congreso español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados. (MAEB) - CEDI 05.

Respecto a las tecnologías relacionadas con el repositorio se han publicado los siguientes artículos: -

- [SSJ04] *Mejores Prácticas .NET*, publicado en la I Jornada de Investigación en Ingeniería Informática. Salamanca-Lisboa - Año 2004.

- [DSJS07] *Secure Mobile Sessions in Web Services with Serialized-Encrypted Objects: Regaining Client Confidence*, publicado en *International Conference on Parallel and Distributed Processing Techniques and Applications*,PDPTA 2007, Las Vegas, Nevada, USA, Año 2007.

Apéndice A

[I]Ontología del repositorio semántico de objetos de aprendizaje

El objetivo de este anexo es proporcionar el código de la ontología que aporta el modelo formal utilizado en el repositorio semántico.

El código está disponible en la siguiente dirección URL:

`http://slor.sourceforge.net/ontology/slor.owl`

A.1. Código OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://slor.sourceforge.net/ontology/slor.owl/##"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://slor.sourceforge.net/ontology/slor.owl/#">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="oc_Action">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >The collection of Events (q.v.) that are carried out by some "
       doer" (see doneBy). Instances of Action include any event in
       which one or more actors effect some change in the (tangible or
        intangible) state of the world, typically by an expenditure of
        effort or energy. Note that it is not required that any
       tangible object be moved, changed, produced, or destroyed for
       an action to occur; the effects of an action might be
       intangible (such as a change in a bank balance or the
       intimidation of a subordinate). </rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="oc_Learning">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The collection of all events, brief or extended, in which an
       agent is acquiring information or know-how.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="oc_Event"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="oc_ComputerFileCopy">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    > specialization of #$ComputerDataArtifact-Complete. Each instance
        of ComputerFileCopy is an information bearing thing that is
       identified as a unit by a unique name, and which is object-like
        in an important respect. Although some files may have other
       files as parts, every file is such that if some information
       were removed from it -- if it were truncated in a certain
       respect -- it would no longer be a file. Examples include
       individual image files, text files, sound files and executables
        (instances of ComputerProgramCopy) that are stored on some
       ComputerStorageDevice</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="LearningObject-AsAnything">
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="usedIn"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#oc_Learning"/>
      </owl:Restriction>
    </owl:equivalentClass>
```

```
33      <rdfs:subClassOf>
34        <owl:Class rdf:ID="LearningObject-Generic"/>
35      </rdfs:subClassOf>
36      <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#
           string"
37      >TODO: Add the possibility that also qualify as LO those things
           used in the events that are contained in actions</
           owl:versionInfo>
38      <owl:equivalentClass>
39        <owl:Restriction>
40          <owl:onProperty>
41            <owl:FunctionalProperty rdf:ID="hasAssociatedMetadataRecord"/>
42          </owl:onProperty>
43          <owl:someValuesFrom>
44            <owl:Class rdf:ID="LearningMetadataRecord"/>
45          </owl:someValuesFrom>
46        </owl:Restriction>
47      </owl:equivalentClass>
48    </owl:Class>
49    <owl:Class rdf:ID="LOM_Record">
50      <rdfs:subClassOf>
51        <owl:Class rdf:about="#LearningMetadataRecord"/>
52      </rdfs:subClassOf>
53    </owl:Class>
54    <owl:Class rdf:ID="SCORM_SCO_Manifest">
55      <rdfs:subClassOf>
56        <owl:Class rdf:about="#LearningMetadataRecord"/>
57      </rdfs:subClassOf>
58    </owl:Class>
59    <owl:Class rdf:ID="IMS_LearningDesign">
60      <rdfs:subClassOf>
61        <owl:Class rdf:ID="LearningObject-AsAnythingDigital"/>
62      </rdfs:subClassOf>
63      <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#
           string"
64      >TODO: Check if an IMS LD instance qualifies with this definition.
           </owl:versionInfo>
65    </owl:Class>
66    <owl:Class rdf:ID="LearningObject-AsAnythingWithEducationalPurpose">
67      <owl:equivalentClass>
```

```
68      <owl:Restriction>
69        <owl:onProperty>
70          <owl:FunctionalProperty rdf:about="#hasAssociatedMetadataRecord
                "/>
71        </owl:onProperty>
72        <owl:someValuesFrom>
73          <owl:Class rdf:about="#LearningMetadataRecord"/>
74        </owl:someValuesFrom>
75      </owl:Restriction>
76    </owl:equivalentClass>
77    <rdfs:subClassOf rdf:resource="#LearningObject-AsAnything"/>
78  </owl:Class>
79  <owl:Class rdf:ID="Description">
80    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
81    >Any descriptive</rdfs:comment>
82  </owl:Class>
83  <owl:Class rdf:about="#oc_Event">
84    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
85    >Each instance of Event is a dynamic situation in which the state
          of the world changes; each instance is something one would say
          "happens". </rdfs:comment>
86  </owl:Class>
87  <owl:Class rdf:ID="oc_IntelligentAgent">
88    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
89    >An agent is an IntelligentAgent if and only if it is capable of
          knowing and acting, and capable of employing its knowledge in
          its actions. An intelligent agent typically knowsAbout certain
          things, and its beliefs concerning those things influences its
          actions. As with agents generally, an intelligent agent might
          either be a single individual, such as a person, or a group
          consisting of two or more individual agents, such as a business
           or government organization.</rdfs:comment>
90  </owl:Class>
91  <owl:Class rdf:ID="LearningObject-LOM">
92    <owl:equivalentClass>
93      <owl:Class>
94        <owl:intersectionOf rdf:parseType="Collection">
95          <owl:Restriction>
96            <owl:someValuesFrom rdf:resource="#LOM_Record"/>
97            <owl:onProperty>
```

```
98              <owl:FunctionalProperty rdf:about="#
                   hasAssociatedMetadataRecord"/>
99            </owl:onProperty>
100         </owl:Restriction>
101         <owl:Class rdf:about="#LearningObject-AsAnythingDigital"/>
102       </owl:intersectionOf>
103     </owl:Class>
104   </owl:equivalentClass>
105 </owl:Class>
106 <owl:Class rdf:ID="oc_Statement">
107   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
108   >A Statement is the InformationBearingThing created in a
        StatingSomething. An #$Agent must create the Statement and it
        contains propositional information</rdfs:comment>
109 </owl:Class>
110   <owl:Class rdf:ID="descriptiveProperty">
111       <rdfs:subClassOf rdf:resource="http://www.cyc.com/2004/06/04/cyc
            /#Statement"/>
112 </owl:Class>
113
114 <owl:Class rdf:ID="simpleString">
115       <rdfs:subClassOf rdf:resource="#descriptiveProperty"/>
116       <owl:disjointWith rdf:resource="#OpencycLink"/>
117       <owl:disjointWith rdf:resource="#OwlLink"/>
118 </owl:Class>
119
120 <owl:DatatypeProperty rdf:ID="language">
121       <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            FunctionalProperty"/>
122       <rdfs:domain rdf:resource="#simpleString"/>
123       <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
            "/>
124       <rdfs:subPropertyOf rdf:resource="#slorCommonProperties"/>
125 </owl:DatatypeProperty>
126
127 <owl:Class rdf:about="#OpencycLink">
128       <rdfs:subClassOf rdf:resource="#descriptiveProperty"/>
129       <owl:disjointWith rdf:resource="#OwlLink"/>
130       <owl:disjointWith rdf:resource="#simpleString"/>
131 </owl:Class>
```

```
<owl:Class rdf:about="#OwlLink">
      <rdfs:subClassOf rdf:resource="#descriptiveProperty"/>
      <owl:disjointWith rdf:resource="#OpencycLink"/>
      <owl:disjointWith rdf:resource="#simpleString"/>
</owl:Class>
<owl:Class rdf:ID="SemanticExpression">
      <owl:equivalentClass>
          <owl:Class>
              <owl:unionOf rdf:parseType="Collection">
                  <owl:Class rdf:about="#conceptual_Link"/>
                  <owl:Class rdf:about="#SemanticProperty_Link"/>
              </owl:unionOf>
          </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#descriptiveProperty"/>
</owl:Class>
<owl:Class rdf:ID="SemanticProperty_Link">
      <owl:equivalentClass>
          <owl:Class>
              <owl:unionOf rdf:parseType="Collection">
                  <owl:Class rdf:about="#oc_SemanticLink"/>
                  <owl:Class rdf:about="#owl_SemanticLink"/>
              </owl:unionOf>
          </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#descriptiveProperty"/>
</owl:Class>
<owl:Class rdf:ID="oc_SemanticLink">
      <rdfs:subClassOf rdf:resource="#OpencycLink"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="arity">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
         FunctionalProperty"/>
      <rdfs:domain rdf:resource="#oc_SemanticLink"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="oc_Predicate">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
         FunctionalProperty"/>
      <rdfs:domain rdf:resource="#oc_SemanticLink"/>
```

```
170         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
171     </owl:DatatypeProperty>
172     <owl:Class rdf:ID="oc_ArgPredicate">
173         <rdfs:subClassOf rdf:resource="#OpencycLink"/>
174     </owl:Class>
175     <owl:DatatypeProperty rdf:ID="norderArg">
176         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
177         <rdfs:domain rdf:resource="#oc_ArgPredicate"/>
178         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
179     </owl:DatatypeProperty>
180     <owl:DatatypeProperty rdf:ID="termArg">
181         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
182         <rdfs:domain rdf:resource="#oc_ArgPredicate"/>
183         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
184     </owl:DatatypeProperty>
185     <owl:Class rdf:ID="conceptual_Link">
186         <owl:equivalentClass>
187             <owl:Class>
188                 <owl:unionOf rdf:parseType="Collection">
189                     <owl:Class rdf:about="#oc_ConceptLink"/>
190                     <owl:Class rdf:about="#owl_ConceptLink"/>
191                 </owl:unionOf>
192             </owl:Class>
193         </owl:equivalentClass>
194         <rdfs:subClassOf rdf:resource="#descriptiveProperty"/>
195     </owl:Class>
196     <owl:Class rdf:ID="owl_ConceptLink">
197         <rdfs:subClassOf rdf:resource="#OwlLink"/>
198         <rdfs:subClassOf>
199             <owl:Restriction>
200                 <owl:onProperty rdf:resource="#ref_Term"/>
201                 <owl:someValuesFrom rdf:resource="http://www.w3.org
                        /2002/07/owl#Thing"/>
202             </owl:Restriction>
203         </rdfs:subClassOf>
204     </owl:Class>
```

```
<owl:ObjectProperty rdf:ID="ref_Term">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
         FunctionalProperty"/>
      <rdfs:domain rdf:resource="#owl_ConceptLink"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="oc_ConceptLink">
      <rdfs:subClassOf rdf:resource="#OpencycLink"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="concept">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
         FunctionalProperty"/>
      <rdfs:domain rdf:resource="#oc_ConceptLink"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
         "/>
</owl:DatatypeProperty>  <owl:Class rdf:about="#
   LearningMetadataRecord">
      <rdfs:subClassOf rdf:resource="#oc_Statement"/>
</owl:Class>
<owl:Class rdf:ID="SCORM_SCO">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#LearningObject-AsAnythingDigital"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#
               hasAssociatedMetadataRecord"/>
          </owl:onProperty>
          <owl:someValuesFrom rdf:resource="#SCORM_SCO_Manifest"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#
     string"
  >TODO: Check that SCORM_SCO also qualifies as a LearningObject-
     AsAnythingWithEducationalPurpose</owl:versionInfo>
</owl:Class>
<owl:Class rdf:about="#LearningObject-AsAnythingDigital">
  <owl:equivalentClass>
```

```
238       <owl:Class>
239         <owl:intersectionOf rdf:parseType="Collection">
240           <owl:Class rdf:about="#LearningObject-AsAnything"/>
241           <owl:Class rdf:about="#oc_ComputerFileCopy"/>
242         </owl:intersectionOf>
243       </owl:Class>
244     </owl:equivalentClass>
245     <rdfs:subClassOf rdf:resource="#LearningObject-Generic"/>
246   </owl:Class>
247   <owl:Class rdf:ID="oc_Thing">
248     <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
249     >Thing is the "universal collection": the collection which, by
          definition, contains everything there is. </rdfs:comment>
250   </owl:Class>
251   <owl:ObjectProperty rdf:about="#usedIn">
252     <rdfs:domain rdf:resource="#LearningObject-AsAnythingDigital"/>
253   </owl:ObjectProperty>
254   <owl:DatatypeProperty rdf:ID="Coverage">
255     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
256     <rdfs:domain rdf:resource="#LOM_Record"/>
257   </owl:DatatypeProperty>
258   <owl:DatatypeProperty rdf:ID="Keywords">
259     <rdfs:domain rdf:resource="#LOM_Record"/>
260     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
261   </owl:DatatypeProperty>
262   <owl:DatatypeProperty rdf:ID="Descriptions">
263     <rdfs:domain rdf:resource="#LOM_Record"/>
264     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
265   </owl:DatatypeProperty>
266   <owl:DatatypeProperty rdf:ID="Language">
267     <rdfs:domain rdf:resource="#LOM_Record"/>
268     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
269   </owl:DatatypeProperty>
270   <owl:DatatypeProperty rdf:ID="Identifier">
271     <rdfs:domain rdf:resource="#LOM_Record"/>
272     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
273     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
          FunctionalProperty"/>
274   </owl:DatatypeProperty>
275   <owl:FunctionalProperty rdf:about="#hasAssociatedMetadataRecord">
```

```
276     <rdfs:domain rdf:resource="#LearningObject-AsAnythingDigital"/>
277     <rdfs:range rdf:resource="#LearningMetadataRecord"/>
278     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"
            />
279   </owl:FunctionalProperty>
280   <owl:FunctionalProperty rdf:ID="Title">
281     <rdfs:domain rdf:resource="#LOM_Record"/>
282     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            DatatypeProperty"/>
283     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
284   </owl:FunctionalProperty>
285 </rdf:RDF>
286
287 <!-- Created with Protege (with OWL Plugin 1.3, Build 225.4) http://
        protege.stanford.edu -->
```

Apéndice B

[II] Representación OWL de diferentes esquemas de registros de metadatos en SLOR

El código de la primera sección está disponible en la siguiente dirección Web:

`http://slor.sourceforge.net/ontology/lom.owl`

El código de las restantes secciones puede encontrarse en:

`http://slor.sourceforge.net/ontology/slor.owl`

B.1. Representación de un registro LOM definido con la ontología LOM-OWL

B.1.1. Ontología LOM-OWL incluida en el esquema base de SLOR

```
1  <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#Element
       "/>
2  <owl:Class rdf:about="http://slor.sourceforge.net/resources/ontologies
       /lom#LangString">
3     <owl:equivalentClass rdf:resource="#simpleString"/>
4     <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
          resources/ontologies/lom#Element"/>
```

```
    </owl:Class>
    <owl:Class rdf:about="http://slor.sourceforge.net/resources/ontologies
        /lom#DateTime">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="http://slor.sourceforge.net
                    /resources/ontologies/lom#dateTimeValue"/>
                <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#int">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Element"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/
        resources/ontologies/lom#dateTimeValue">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://slor.sourceforge.net/resources/
            ontologies/lom#DateTime"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
            dateTime"/>
        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Property"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/resources/
        ontologies/lom#dateTimeDescription">
        <rdfs:domain rdf:resource="http://slor.sourceforge.net/resources/
            ontologies/lom#DateTime"/>
        <rdfs:range rdf:resource="http://slor.sourceforge.net/resources/
            ontologies/lom#LangString"/>
        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Property"/>
    </owl:ObjectProperty>
    <owl:Class rdf:about="http://slor.sourceforge.net/resources/ontologies
        /lom#Duration">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="http://slor.sourceforge.net
                    /resources/ontologies/lom#durationValue"/>
```

```
30              <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#int">1</owl:cardinality>
31          </owl:Restriction>
32      </rdfs:subClassOf>
33      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Element"/>
34  </owl:Class>
35  <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/
        resources/ontologies/lom#durationValue">
36      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            FunctionalProperty"/>
37      <rdfs:domain rdf:resource="http://slor.sourceforge.net/resources/
            ontologies/lom#Duration"/>
38      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
            duration"/>
39      <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Property"/>
40  </owl:DatatypeProperty>
41  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/resources/
        ontologies/lom#durationDescription">
42      <rdfs:domain rdf:resource="http://slor.sourceforge.net/resources/
            ontologies/lom#Duration"/>
43      <rdfs:range rdf:resource="http://slor.sourceforge.net/resources/
            ontologies/lom#LangString"/>
44      <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Property"/>
45  </owl:ObjectProperty>
46  <owl:Class rdf:about="http://slor.sourceforge.net/resources/ontologies
        /lom#Vocabulary">
47      <owl:equivalentClass rdf:resource="#conceptual_Link"/>
48      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            resources/ontologies/lom#Element"/>
49  </owl:Class>
50
51  <owl:Class rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
        LOR_LOMOWL">
52      <owl:equivalentClass>
53          <owl:Restriction>
54              <owl:onProperty rdf:resource="http://slor.sourceforge.net
                    /ontology/slor.owl#hasAssociatedMetadataRecord"/>
```

```
55              <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                    .net/ontology/slor.owl#LOMOWL_Record"/>
56          </owl:Restriction>
57      </owl:equivalentClass>
58      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/slor.owl#LearningObject-AsAnythingDigital"/>
59  </owl:Class>
60  <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
        Identifier">
61      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#Element"/>
62  </owl:Class>
63  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#identifier">
64      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            FunctionalProperty"/>
65      <rdfs:domain>
66          <owl:Class>
67              <owl:unionOf rdf:parseType="Collection">
68                  <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/lom#MetaMetaData_Record"/>
69                  <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/slor.owl#LOMOWL_Record"/>
70              </owl:unionOf>
71          </owl:Class>
72      </rdfs:domain>
73      <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
            lom#Identifier"/>
74      <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#Property"/>
75  </owl:ObjectProperty>
76  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#language">
77      <rdfs:domain>
78          <owl:Class>
79              <owl:unionOf rdf:parseType="Collection">
80                  <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/lom#MetaMetaData_Record"/>
81                  <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/slor.owl#LOMOWL_Record"/>
```

```
82              </owl:unionOf>
83          </owl:Class>
84      </rdfs:domain>
85      <rdfs:range rdf:resource="http://www.cyc.com/2004/06/04/cyc/#
            HumanLanguage"/>
86      <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#GeneralPropertiesCategory"/>
87  </owl:ObjectProperty>
88  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#description">
89      <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
            slor.owl#LOMOWL_Record"/>
90      <rdfs:range>
91          <owl:Class>
92              <owl:unionOf rdf:parseType="Collection">
93                  <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/lom#LangString"/>
94                  <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/slor.owl#SemanticExpression"/>
95              </owl:unionOf>
96          </owl:Class>
97      </rdfs:range>
98      <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#GeneralPropertiesCategory"/>
99  </owl:ObjectProperty>
100 <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#keyword">
101     <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
            slor.owl#LOMOWL_Record"/>
102     <rdfs:range>
103         <owl:Class>
104             <owl:unionOf rdf:parseType="Collection">
105                 <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/slor.owl#conceptual_Link"/>
106                 <owl:Class rdf:about="http://slor.sourceforge.net/
                        ontology/lom#LangString"/>
107             </owl:unionOf>
108         </owl:Class>
109     </rdfs:range>
```

```
110         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#GeneralPropertiesCategory"/>
111     </owl:ObjectProperty>
112     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_author"/>
113     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_contentprovider"/>
114     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_editor"/>
115     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_educationalvalidator"/>
116     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_graphicaldesigner"/>
117     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_initiator"/>
118     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_instructionaldesigner"/>
119     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_publisher"/>
120     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_scriptwriter"/>
121     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_subjectmatterexpert"/>
122     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_technicalimplementer"/>
123     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_technicalvalidator"/>
124     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_terminator"/>
125     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_unknown"/>
126     <lom:RolesVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#_validator"/>
127     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#accessibility"/>
128     <lom:InteractivityTypeVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#active"/>
129     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#aggregationLevel">
```

```
130         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
               FunctionalProperty"/>
131         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               slor.owl#LOMOWL_Record"/>
132         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
               lom#Vocabulary"/>
133         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#GeneralPropertiesCategory"/>
134         <propertyContext rdf:resource="http://slor.sourceforge.net/
               ontology/lom#AggregationLevelVocabulary"/>
135     </owl:ObjectProperty>
136     <lom:DifficultyVocabularyItem rdf:about="http://slor.sourceforge.net
           /ontology/lom#_medium"/>
137     <lom:AggregationLevelVocabularyItem rdf:about="http://slor.
           sourceforge.net/ontology/lom#AggregationLevel1"/>
138     <lom:AggregationLevelVocabularyItem rdf:about="http://slor.
           sourceforge.net/ontology/lom#AggregationLevel2"/>
139     <lom:AggregationLevelVocabularyItem rdf:about="http://slor.
           sourceforge.net/ontology/lom#AggregationLevel3"/>
140     <lom:AggregationLevelVocabularyItem rdf:about="http://slor.
           sourceforge.net/ontology/lom#AggregationLevel4"/>
141     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           AggregationLevelVocabulary">
142         <owl:equivalentClass>
143             <owl:Restriction>
144                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                       /ontology/slor.owl#ref_Term"/>
145                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                       .net/ontology/lom#AggregationLevelVocabularyItem"/>
146             </owl:Restriction>
147         </owl:equivalentClass>
148         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#LOMv1.0"/>
149     </owl:Class>
150     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           AggregationLevelVocabularyItem">
151         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#VocabularyItem"/>
152     </owl:Class>
```

```
153     <lom:RequirementBrowserValue rdf:about="http://slor.sourceforge.net/
            ontology/lom#amaya"/>
154     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            Annotation">
155         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
156     </owl:Class>
157     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#annotation">
158         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
159         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Annotation"/>
160         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#annotationPropertiesCategory"/>
161     </owl:ObjectProperty>
162     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#annotationDate">
163         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
164         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Annotation"/>
165         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#DateTime"/>
166         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#annotationPropertiesCategory"/>
167     </owl:ObjectProperty>
168     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#annotationDescription">
169         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Annotation"/>
170         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
171         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#annotationPropertiesCategory"/>
172     </owl:ObjectProperty>
173     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#annotationEntity">
174         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
```

```
175         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               lom#Annotation"/>
176         <rdfs:range>
177             <owl:Class>
178                 <owl:unionOf rdf:parseType="Collection">
179                     <owl:Class rdf:about="http://www.cyc.com/2004/06/04/
                           cyc/#Organization"/>
180                     <owl:Class rdf:about="http://slor.sourceforge.net/
                           ontology/slor.owl#vCard"/>
181                 </owl:unionOf>
182             </owl:Class>
183         </rdfs:range>
184         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#annotationPropertiesCategory"/>
185     </owl:ObjectProperty>
186     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
           lom#annotationPropertiesCategory">
187         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#Property"/>
188     </owl:ObjectProperty>
189     <lom:StructureVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#atomic"/>
190     <lom:IntendedEndUserRoleVocabularyItem rdf:about="http://slor.
           sourceforge.net/ontology/lom#author"/>
191     <lom:RequirementType rdf:about="http://slor.sourceforge.net/ontology/
           lom#browser"/>
192     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
           /lom#catalog">
193         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
               FunctionalProperty"/>
194         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               lom#Identifier"/>
195         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
               "/>
196         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#Property"/>
197     </owl:DatatypeProperty>
198     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           Classification">
```

```
199         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
200     </owl:Class>
201     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#classification">
202         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
203         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Classification"/>
204         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
205     </owl:ObjectProperty>
206     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#classificationPropertiesCategory">
207         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
208     </owl:ObjectProperty>
209     <lom:StructureVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#collection"/>
210     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#competency"/>
211     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#context">
212         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
213         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
214         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
215         <propertyContext rdf:resource="http://www.cyc.com/2004/06/04/cyc
                /#Microtheory"/>
216         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#ContextVocabulary"/>
217     </owl:ObjectProperty>
218     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            ContextVocabulary">
219         <owl:equivalentClass>
220             <owl:Restriction>
221                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
```

```
222             <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                    .net/ontology/lom#ContextVocabulary"/>
223         </owl:Restriction>
224     </owl:equivalentClass>
225     <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#LOMv1.0"/>
226 </owl:Class>
227 <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
        ContextVocabularyItem">
228     <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#VocabularyItem"/>
229 </owl:Class>
230 <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
        Contribute">
231     <rdfs:subClassOf>
232         <owl:Restriction>
233             <owl:onProperty rdf:resource="http://slor.sourceforge.net
                    /ontology/lom#entity"/>
234             <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#int">1</owl:minCardinality>
235         </owl:Restriction>
236     </rdfs:subClassOf>
237     <rdfs:subClassOf>
238         <owl:Restriction>
239             <owl:onProperty rdf:resource="http://slor.sourceforge.net
                    /ontology/lom#contributeRole"/>
240             <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#int">1</owl:cardinality>
241         </owl:Restriction>
242     </rdfs:subClassOf>
243     <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#Element"/>
244 </owl:Class>
245 <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#contributeDate">
246     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            FunctionalProperty"/>
247     <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
            lom#Contribute"/>
```

```
248         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#DateTime"/>
249         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#contributeProperty"/>
250     </owl:ObjectProperty>
251     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#contributeLifeCycle">
252         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
253         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Contribute"/>
254         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LifeCyclePropertiesCategory"/>
255     </owl:ObjectProperty>
256     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#contributeMetaData">
257         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
258         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#MetaMetaData_Record"/>
259         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Contribute"/>
260         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#metametadataPropertiesCategory"/>
261     </owl:ObjectProperty>
262     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#contributeProperty">
263         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
264     </owl:ObjectProperty>
265     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#contributeRole">
266         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
267         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Contribute"/>
268         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
269         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#contributeProperty"/>
```

```
270         <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"
271             >lom:Metadata_Record===lom:MetaRolVocabulary
272 LOMOWL_Record===lom:RolesVocabulary</rdfs:comment>
273         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#MetaRolVocabulary"/>
274         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RolesVocabulary"/>
275     </owl:ObjectProperty>
276     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#copyrightAndOtherRestrictions">
277         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
278         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
279         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
280         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#rightsPropertiesCategory"/>
281         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#YesNoVocabulary"/>
282     </owl:ObjectProperty>
283     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#cost">
284         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
285         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
286         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
287         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#rightsPropertiesCategory"/>
288         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#YesNoVocabulary"/>
289     </owl:ObjectProperty>
290     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#coverage">
291         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
292         <rdfs:range>
```

```
293             <owl:Class>
294                 <owl:unionOf rdf:parseType="Collection">
295                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#LangString"/>
296                     <owl:Class rdf:about="http://www.cyc.com/2004/06/04/
                            cyc/#AdministrativeUnit"/>
297                     <owl:Class rdf:about="http://www.cyc.com/2004/06/04/
                            cyc/#GeographicalRegion"/>
298                     <owl:Class rdf:about="http://www.cyc.com/2004/06/04/
                            cyc/#TimeInterval"/>
299                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#OwlLink"/>
300                 </owl:unionOf>
301             </owl:Class>
302         </rdfs:range>
303         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#GeneralPropertiesCategory"/>
304     </owl:ObjectProperty>
305     <lom:MetaRolVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#creator"/>
306     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            DateTime">
307         <rdfs:subClassOf>
308             <owl:Restriction>
309                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/lom#dateTimeValue"/>
310                 <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">1</owl:cardinality>
311             </owl:Restriction>
312         </rdfs:subClassOf>
313         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
314     </owl:Class>
315     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#dateTimeDescription">
316         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#DateTime"/>
317         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
```

```
318         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
319     </owl:ObjectProperty>
320     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#dateTimeValue">
321         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
322         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#DateTime"/>
323         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                dateTime"/>
324         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
325     </owl:DatatypeProperty>
326
327     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#descriptionAgeRange">
328         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#SemanticAgeRange"/>
329         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
330         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
331     </owl:ObjectProperty>
332     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#diagram"/>
333     <lom:DifficultyVocabularyItem rdf:about="http://slor.sourceforge.net
            /ontology/lom#difficult"/>
334     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            DifficultVocabulary">
335         <owl:equivalentClass>
336             <owl:Restriction>
337                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
338                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#DifficultyVocabularyItem"/>
339             </owl:Restriction>
340         </owl:equivalentClass>
341         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
```

```
342     </owl:Class>
343     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#difficulty">
344         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
345         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
346         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
347         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
348     </owl:ObjectProperty>
349     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            DifficultyVocabularyItem">
350         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
351     </owl:Class>
352     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#discipline"/>
353     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#dominantLearningResourceType">
354         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
355         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
356         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
357         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
358         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LearningResourceVocabulary"/>
359     </owl:ObjectProperty>
360     <lom:StatusVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#draft"/>
361     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            Duration">
362         <rdfs:subClassOf>
363             <owl:Restriction>
364                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/lom#durationValue"/>
```

```
365             <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#int">1</owl:cardinality>
366         </owl:Restriction>
367     </rdfs:subClassOf>
368     <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#Element"/>
369 </owl:Class>
370 <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#duration">
371     <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
            slor.owl#LOMOWL_Record"/>
372     <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
            lom#Duration"/>
373     <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#technicalPropertiesCategory"/>
374 </owl:ObjectProperty>
375 <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        lom#durationDescription">
376     <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
            lom#Duration"/>
377     <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
            lom#LangString"/>
378     <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#Property"/>
379 </owl:ObjectProperty>
380 <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
        /lom#durationValue">
381     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            FunctionalProperty"/>
382     <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
            lom#Duration"/>
383     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
            duration"/>
384     <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
            ontology/lom#Property"/>
385 </owl:DatatypeProperty>
386 <lom:DifficultyVocabularyItem rdf:about="http://slor.sourceforge.net
        /ontology/lom#easy"/>
387 <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
        ontology/lom#educational_level"/>
```

```
388     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#educational_objective"/>
389     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#educationalDescription">
390         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
391         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
392         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
393     </owl:ObjectProperty>
394     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#EducationalPropertiesCategory">
395         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
396     </owl:ObjectProperty>
397     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#entity">
398         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Contribute"/>
399         <rdfs:range>
400             <owl:Class>
401                 <owl:unionOf rdf:parseType="Collection">
402                     <owl:Class rdf:about="http://www.cyc.com/2004/06/04/
                            cyc/#Organization"/>
403                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#vCard"/>
404                 </owl:unionOf>
405             </owl:Class>
406         </rdfs:range>
407         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#contributeProperty"/>
408     </owl:ObjectProperty>
409     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#entry">
410         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
411         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Identifier"/>
```

```
412         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
413         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
414     </owl:DatatypeProperty>
415     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#exam"/>
416     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#exercise"/>
417     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#experiment"/>
418     <lom:InteractivityTypeVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#expositive"/>
419     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#figure"/>
420     <lom:StatusVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#final"/>
421     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#format">
422         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
423         <rdfs:range>
424             <owl:Class>
425                 <owl:unionOf rdf:parseType="Collection">
426                     <owl:Class rdf:about="http://www.iana.org/assignments/
                            media-types/#MimeType"/>
427                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#Non-Digital"/>
428                 </owl:unionOf>
429             </owl:Class>
430         </rdfs:range>
431         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalPropertiesCategory"/>
432     </owl:ObjectProperty>
433     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#GeneralPropertiesCategory">
434         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
435     </owl:ObjectProperty>
```

```
436     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#graph"/>
437     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#hasformat"/>
438     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#haspart"/>
439     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#hasTechnicalRequirement">
440         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#OrComposite"/>
441         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#TechnicalRequirement"/>
442         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
443     </owl:ObjectProperty>
444     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#hasversion"/>
445     <lom:StructureVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#hierarchical"/>
446     <lom:LevelVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#high"/>
447     <lom:ContextVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#higher_education"/>
448     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#idea"/>
449     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#index"/>
450     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#installationRemarks">
451         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
452         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
453         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalPropertiesCategory"/>
454     </owl:ObjectProperty>
455     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#intendedEndUserRole">
456         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
```

```
457         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
458         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
459         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
460         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#IntendedEndUserRoleVocabulary"/>
461     </owl:ObjectProperty>
462     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            IntendedEndUserRoleVocabulary">
463         <owl:equivalentClass>
464             <owl:Restriction>
465                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
466                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#IntendedEndUserRoleVocabularyItem"/>
467             </owl:Restriction>
468         </owl:equivalentClass>
469         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
470     </owl:Class>
471     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            IntendedEndUserRoleVocabularyItem">
472         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
473     </owl:Class>
474     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            InteractivityLevelVocabulary">
475         <owl:equivalentClass>
476             <owl:Restriction>
477                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
478                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#LevelVocabularyItem"/>
479             </owl:Restriction>
480         </owl:equivalentClass>
481         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
482     </owl:Class>
```

```
483     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#interactivityType">
484         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
485         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
486         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
487         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
488         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#InterativityTypeVocabulary"/>
489     </owl:ObjectProperty>
490     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            InteractivityTypeVocabularyItem">
491         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
492     </owl:Class>
493     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#interativityLevel">
494         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
495         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
496         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
497         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#InterativityTypeVocabulary"/>
498     </owl:ObjectProperty>
499     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            InterativityTypeVocabulary">
500         <owl:equivalentClass>
501             <owl:Restriction>
502                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
503                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#InteractivityTypeVocabularyItem"/>
504             </owl:Restriction>
505         </owl:equivalentClass>
```

```
506         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
507     </owl:Class>
508     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#isbasedon"/>
509     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#isbasicfor"/>
510     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#isformatof"/>
511     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#ispartof"/>
512     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#isreferencedby"/>
513     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#isrequiredby"/>
514     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#isversionof"/>
515     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#kindOfRelation">
516         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Relation"/>
517         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
518         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#relationPropertiesCategory"/>
519         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RelationVocabulary"/>
520     </owl:ObjectProperty>
521     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            LangString">
522         <owl:equivalentClass rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#simpleString"/>
523         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
524     </owl:Class>
525
526     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /slor.owl#Identifier">
527         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
```

```
528         <rdfs:domain>
529             <owl:Class>
530                 <owl:unionOf rdf:parseType="Collection">
531                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#LearningObject-Generic"/>
532                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#TaxonID"/>
533                 </owl:unionOf>
534             </owl:Class>
535         </rdfs:domain>
536         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
537         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#slorCommonProperties"/>
538     </owl:DatatypeProperty>
539
540
541     <lom:IntendedEndUserRoleVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#learner"/>
542     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#learningResourceType">
543         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
544         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
545         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
546         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LearningResourceVocabulary"/>
547     </owl:ObjectProperty>
548     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            LearningResourceVocabulary">
549         <owl:equivalentClass>
550             <owl:Restriction>
551                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
552                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#LearningResourceVocabularyItem"/>
553             </owl:Restriction>
554         </owl:equivalentClass>
```

```
555         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
556     </owl:Class>
557     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            LearningResourceVocabularyItem">
558         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
559     </owl:Class>
560     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#lecture"/>
561     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            LevelVocabularyItem">
562         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
563     </owl:Class>
564     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#LifeCyclePropertiesCategory">
565         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
566     </owl:ObjectProperty>
567     <lom:StructureVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#linear"/>
568     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#location">
569         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
570         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
571         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI
                "/>
572         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalPropertiesCategory"/>
573         <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"
574             >The same property that fileFromURI (oc:ComputerFileCopy)</
                    rdfs:comment>
575     </owl:DatatypeProperty>
576     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#LOMv1.0
            ">
577         <owl:equivalentClass>
```

```
578             <owl:Class>
579                 <owl:intersectionOf rdf:parseType="Collection">
580                     <owl:Restriction>
581                         <owl:onProperty rdf:resource="http://slor.
                                sourceforge.net/ontology/slor.owl#
                                ontologySchemaLink"/>
582                         <owl:hasValue rdf:datatype="http://www.w3.org
                                /2001/XMLSchema#string">LOMv1.0</owl:hasValue>
583                     </owl:Restriction>
584                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#owl_ConceptLink"/>
585                 </owl:intersectionOf>
586             </owl:Class>
587         </owl:equivalentClass>
588         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Vocabulary"/>
589         <owl:disjointWith rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#oc_ConceptLink"/>
590     </owl:Class>
591     <lom:RolesVocabulary rdf:about="http://slor.sourceforge.net/ontology/
            lom#LOMv1.0-Creator">
592         <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
                creator"/>
593     </lom:RolesVocabulary>
594     <lom:IntendedEndUserRoleVocabulary rdf:about="http://slor.
            sourceforge.net/ontology/lom#LOMv1.0-Learner">
595         <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
                learner"/>
596     </lom:IntendedEndUserRoleVocabulary>
597     <lom:RequirementValuesVocabulary rdf:about="http://slor.sourceforge.
            net/ontology/lom#LOMv1.0-MS_Windows">
598         <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
                ms-windows"/>
599     </lom:RequirementValuesVocabulary>
600     <lom:RequirementTypesVocabulary rdf:about="http://slor.sourceforge.
            net/ontology/lom#LOMv1.0-Operating_System">
601         <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
                operating_system"/>
602     </lom:RequirementTypesVocabulary>
```

```
603     <lom:InteractivityLevelVocabulary rdf:about="http://slor.sourceforge
            .net/ontology/lom#LOMv1.0-very_low">
604         <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
                very_low"/>
605     </lom:InteractivityLevelVocabulary>
606     <lom:RolesVocabulary rdf:about="http://slor.sourceforge.net/ontology/
            lom#LOMv1.0_ContentProvider">
607         <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
                _contentprovider"/>
608     </lom:RolesVocabulary>
609     <lom:LevelVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#low"/>
610     <lom:RequirementOperatingSystemValue rdf:about="http://slor.
            sourceforge.net/ontology/lom#macos"/>
611     <lom:IntendedEndUserRoleVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#manager"/>
612     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#maxAgeRange">
613         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
614         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#SemanticAgeRange"/>
615         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
616         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
617     </owl:DatatypeProperty>
618     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#maxVersion">
619         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
620         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#TechnicalRequirement"/>
621         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
622         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalrequirementProperty"/>
623     </owl:DatatypeProperty>
624     <lom:LevelVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#medium"/>
```

```
625     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            MetadataSchema">
626         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
627     </owl:Class>
628     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#metaMetadata">
629         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
630         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
631         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#MetaMetaData_Record"/>
632         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#metametadataPropertiesCategory"/>
633     </owl:ObjectProperty>
634     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            MetaMetaData_Record">
635         <rdfs:subClassOf>
636             <owl:Restriction>
637                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/lom#language"/>
638                 <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">1</owl:maxCardinality>
639             </owl:Restriction>
640         </rdfs:subClassOf>
641         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
642     </owl:Class>
643     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#metametadataPropertiesCategory">
644         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
645     </owl:ObjectProperty>
646     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            MetaRolVocabulary">
647         <owl:equivalentClass>
648             <owl:Restriction>
649                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
```

```
650                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#MetaRolVocabularyItem"/>
651             </owl:Restriction>
652         </owl:equivalentClass>
653         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
654     </owl:Class>
655     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            MetaRolVocabularyItem">
656         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
657     </owl:Class>
658     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#metaSchema">
659         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#MetaMetaData_Record"/>
660         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#MetadataSchema"/>
661         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#metametadataPropertiesCategory"/>
662     </owl:ObjectProperty>
663     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#minAgeRange">
664         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
665         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#SemanticAgeRange"/>
666         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
667         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
668     </owl:DatatypeProperty>
669     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#minVersion">
670         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
671         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#TechnicalRequirement"/>
672         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
```

```
673         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalrequirementProperty"/>
674     </owl:DatatypeProperty>
675     <lom:InteractivityTypeVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#mixed"/>
676     <lom:RequirementBrowserValue rdf:about="http://slor.sourceforge.net/
            ontology/lom#ms-internet_explorer"/>
677     <lom:RequirementOperatingSystemValue rdf:about="http://slor.
            sourceforge.net/ontology/lom#ms-windows"/>
678     <lom:RequirementOperatingSystemValue rdf:about="http://slor.
            sourceforge.net/ontology/lom#multi-os"/>
679     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#narrative_text"/>
680     <lom:RequirementBrowserValue rdf:about="http://slor.sourceforge.net/
            ontology/lom#netscape_communicator"/>
681     <lom:StructureVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#networked"/>
682     <lom:YesNoVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#no"/>
683     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#Non-
            Digital">
684         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
685     </owl:Class>
686     <lom:RequirementOperatingSystemValue rdf:about="http://slor.
            sourceforge.net/ontology/lom#none"/>
687     <lom:RequirementBrowserValue rdf:about="http://slor.sourceforge.net/
            ontology/lom#opera"/>
688     <lom:RequirementType rdf:about="http://slor.sourceforge.net/ontology/
            lom#operating_system"/>
689     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            OrComposite">
690         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
691     </owl:Class>
692     <lom:ContextVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#other"/>
693     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#otherPlatformRequirements">
```

```
694         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               slor.owl#LOMOWL_Record"/>
695         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
               lom#LangString"/>
696         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#technicalPropertiesCategory"/>
697     </owl:ObjectProperty>
698     <lom:RequirementOperatingSystemValue rdf:about="http://slor.
           sourceforge.net/ontology/lom#pc-dos"/>
699     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#prerequisite"/>
700     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
           sourceforge.net/ontology/lom#problem_statement"/>
701     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
           lom#Property"/>
702     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
           lom#purpose">
703         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
               FunctionalProperty"/>
704         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               lom#Classification"/>
705         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
               lom#Vocabulary"/>
706         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#classificationPropertiesCategory"/>
707         <propertyContext rdf:resource="http://slor.sourceforge.net/
               ontology/lom#PurposeVocabulary"/>
708     </owl:ObjectProperty>
709     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           PurposeVocabulary">
710         <owl:equivalentClass>
711             <owl:Restriction>
712                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                       /ontology/slor.owl#ref_Term"/>
713                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                       .net/ontology/lom#PurposeVocabularyItem"/>
714             </owl:Restriction>
715         </owl:equivalentClass>
716         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#LOMv1.0"/>
```

```
717     </owl:Class>
718     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            PurposeVocabularyItem">
719         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
720     </owl:Class>
721     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#questionnarie"/>
722     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#references"/>
723     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#refTaxon">
724         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Taxon"/>
725         <rdfs:range>
726             <owl:Class>
727                 <owl:unionOf rdf:parseType="Collection">
728                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#conceptual_Link"/>
729                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#TaxonID"/>
730                 </owl:unionOf>
731             </owl:Class>
732         </rdfs:range>
733         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
734     </owl:ObjectProperty>
735     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#relatedLO">
736         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Relation"/>
737         <rdfs:range>
738             <owl:Class>
739                 <owl:unionOf rdf:parseType="Collection">
740                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#LearningObject-AsAnything"/>
741                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#Identifier"/>
742                 </owl:unionOf>
743             </owl:Class>
```

```
744         </rdfs:range>
745         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#relationPropertiesCategory"/>
746     </owl:ObjectProperty>
747     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#relatedLODescription">
748         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Relation"/>
749         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
750         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#relationPropertiesCategory"/>
751     </owl:ObjectProperty>
752     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            Relation">
753         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
754     </owl:Class>
755     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#relationLO">
756         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
757         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Relation"/>
758         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#relationPropertiesCategory"/>
759     </owl:ObjectProperty>
760     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#relationPropertiesCategory">
761         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
762     </owl:ObjectProperty>
763     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RelationVocabulary">
764         <owl:equivalentClass>
765             <owl:Restriction>
766                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
767                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#RelationVocabularyItem"/>
```

```
768             </owl:Restriction>
769         </owl:equivalentClass>
770         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
771     </owl:Class>
772     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RelationVocabularyItem">
773         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
774     </owl:Class>
775     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#requirement">
776         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
777         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#OrComposite"/>
778         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalPropertiesCategory"/>
779     </owl:ObjectProperty>
780     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementBrowserValue">
781         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementType"/>
782     </owl:Class>
783     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#requirementName">
784         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#TechnicalRequirement"/>
785         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
786         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalrequirementProperty"/>
787         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementValuesVocabulary"/>
788     </owl:ObjectProperty>
789     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementOperatingSystemValue">
790         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementType"/>
791     </owl:Class>
```

```
792     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementType">
793         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementVocabularyItem"/>
794     </owl:Class>
795     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#requirementType">
796         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#TechnicalRequirement"/>
797         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
798         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalrequirementProperty"/>
799         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementTypesVocabulary"/>
800     </owl:ObjectProperty>
801     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementTypesVocabulary">
802         <owl:equivalentClass>
803             <owl:Class>
804                 <owl:unionOf rdf:parseType="Collection">
805                     <owl:Restriction>
806                         <owl:onProperty rdf:resource="http://slor.
                                sourceforge.net/ontology/slor.owl#ref_Term"/>
807                         <owl:hasValue rdf:resource="http://slor.
                                sourceforge.net/ontology/lom#browser"/>
808                     </owl:Restriction>
809                     <owl:Restriction>
810                         <owl:onProperty rdf:resource="http://slor.
                                sourceforge.net/ontology/slor.owl#ref_Term"/>
811                         <owl:hasValue rdf:resource="http://slor.
                                sourceforge.net/ontology/lom#operating_system"/>
812                     </owl:Restriction>
813                 </owl:unionOf>
814             </owl:Class>
815         </owl:equivalentClass>
816         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementVocabulary"/>
817     </owl:Class>
```

```
818     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementValuesVocabulary">
819         <owl:equivalentClass>
820             <owl:Restriction>
821                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
822                 <owl:someValuesFrom>
823                     <owl:Class>
824                         <owl:unionOf rdf:parseType="Collection">
825                             <owl:Class rdf:about="http://slor.sourceforge.
                                    net/ontology/lom#RequirementBrowserValue"/>
826                             <owl:Class rdf:about="http://slor.sourceforge.
                                    net/ontology/lom#
                                    RequirementOperatingSystemValue"/>
827                         </owl:unionOf>
828                     </owl:Class>
829                 </owl:someValuesFrom>
830             </owl:Restriction>
831         </owl:equivalentClass>
832         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#RequirementVocabulary"/>
833     </owl:Class>
834     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementVocabulary">
835         <owl:equivalentClass>
836             <owl:Restriction>
837                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
838                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#RequirementVocabularyItem"/>
839             </owl:Restriction>
840         </owl:equivalentClass>
841         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
842     </owl:Class>
843     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RequirementVocabularyItem">
844         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
845     </owl:Class>
```

```
846     <lom:RelationVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#requires"/>
847     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#restrictions"/>
848     <lom:StatusVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#revised"/>
849     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#rightsDescription">
850         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
851         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
852         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#rightsPropertiesCategory"/>
853     </owl:ObjectProperty>
854     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#rightsPropertiesCategory">
855         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
856     </owl:ObjectProperty>
857     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RolesVocabulary">
858         <owl:equivalentClass>
859             <owl:Restriction>
860                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
861                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#RolesVocabularyItem"/>
862             </owl:Restriction>
863         </owl:equivalentClass>
864         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
865     </owl:Class>
866     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            RolesVocabularyItem">
867         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
868     </owl:Class>
869     <lom:ContextVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#school"/>
```

```
870     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#security_level"/>
871     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#self_assessment"/>
872     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            SemanticAgeRange">
873         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
874     </owl:Class>
875     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#semanticDensity">
876         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
877         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
878         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
879         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
880         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#SemanticDensityVocabulary"/>
881     </owl:ObjectProperty>
882     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            SemanticDensityVocabulary">
883         <owl:equivalentClass>
884             <owl:Restriction>
885                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
886                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#LevelVocabularyItem"/>
887             </owl:Restriction>
888         </owl:equivalentClass>
889         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
890     </owl:Class>
891     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#simulation"/>
892     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /lom#size">
```

```
893         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
894         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
895         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/
                >
896         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalPropertiesCategory"/>
897     </owl:DatatypeProperty>
898     <lom:PurposeVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#skill_level"/>
899     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#slide"/>
900     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#status">
901         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
902         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
903         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
904         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LifeCyclePropertiesCategory"/>
905         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#StatusVocabulary"/>
906     </owl:ObjectProperty>
907     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            StatusVocabulary">
908         <owl:equivalentClass>
909             <owl:Restriction>
910                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
911                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#StatusVocabularyItem"/>
912             </owl:Restriction>
913         </owl:equivalentClass>
914         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
915     </owl:Class>
```

```
916     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            StatusVocabularyItem">
917         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
918     </owl:Class>
919     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#structure">
920         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
921         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
922         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Vocabulary"/>
923         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#GeneralPropertiesCategory"/>
924         <propertyContext rdf:resource="http://slor.sourceforge.net/
                ontology/lom#StructureVocabulary"/>
925     </owl:ObjectProperty>
926     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            StructureVocabulary">
927         <owl:equivalentClass>
928             <owl:Restriction>
929                 <owl:onProperty rdf:resource="http://slor.sourceforge.net
                        /ontology/slor.owl#ref_Term"/>
930                 <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                        .net/ontology/lom#StructureVocabularyItem"/>
931             </owl:Restriction>
932         </owl:equivalentClass>
933         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0"/>
934     </owl:Class>
935     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            StructureVocabularyItem">
936         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#VocabularyItem"/>
937     </owl:Class>
938     <lom:LearningResourceVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#table"/>
939     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#Taxon"
            >
```

```
940         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
941     </owl:Class>
942     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#taxon">
943         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Classification"/>
944         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Taxon"/>
945         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
946     </owl:ObjectProperty>
947     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#taxonDescription">
948         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Taxon"/>
949         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
950         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
951     </owl:ObjectProperty>
952     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#taxonEntry">
953         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
954         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#TaxonID"/>
955         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#LangString"/>
956         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
957     </owl:ObjectProperty>
958     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#TaxonID
            ">
959         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
960     </owl:Class>
961     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#taxonKeyword">
```

```
962         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Taxon"/>
963         <rdfs:range>
964             <owl:Class>
965                 <owl:unionOf rdf:parseType="Collection">
966                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#conceptual_Link"/>
967                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#LangString"/>
968                 </owl:unionOf>
969             </owl:Class>
970         </rdfs:range>
971         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
972     </owl:ObjectProperty>
973     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#taxonSource">
974         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
975         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Taxon"/>
976         <rdfs:range>
977             <owl:Class>
978                 <owl:unionOf rdf:parseType="Collection">
979                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/slor.owl#ClassificationSchema"/>
980                     <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#LangString"/>
981                 </owl:unionOf>
982             </owl:Class>
983         </rdfs:range>
984         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#classificationPropertiesCategory"/>
985     </owl:ObjectProperty>
986     <lom:IntendedEndUserRoleVocabularyItem rdf:about="http://slor.
            sourceforge.net/ontology/lom#teacher"/>
987     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#technicalPropertiesCategory">
988         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Property"/>
```

```
989     </owl:ObjectProperty>
990     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
            TechnicalRequirement">
991         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#Element"/>
992     </owl:Class>
993     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#technicalrequirementProperty">
994         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#technicalPropertiesCategory"/>
995     </owl:ObjectProperty>
996     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#typicalAgeRange">
997         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
998         <rdfs:range>
999             <owl:Class>
1000                <owl:unionOf rdf:parseType="Collection">
1001                    <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#LangString"/>
1002                    <owl:Class rdf:about="http://slor.sourceforge.net/
                            ontology/lom#SemanticAgeRange"/>
1003                </owl:unionOf>
1004            </owl:Class>
1005        </rdfs:range>
1006        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
1007    </owl:ObjectProperty>
1008    <lom:ContextVocabularyItem rdf:about="http://slor.sourceforge.net/
            ontology/lom#training"/>
1009    <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            lom#typicalLearningTime">
1010        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
1011        <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMOWL_Record"/>
1012        <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                lom#Duration"/>
1013        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/lom#EducationalPropertiesCategory"/>
```

```
1014    </owl:ObjectProperty>
1015    <lom:StatusVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#unavaiable"/>
1016    <lom:RequirementOperatingSystemValue rdf:about="http://slor.
           sourceforge.net/ontology/lom#unix"/>
1017    <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
           lom#userLanguage">
1018        <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               slor.owl#LOMOWL_Record"/>
1019        <rdfs:range rdf:resource="http://www.cyc.com/2004/06/04/cyc/#
               HumanLanguage"/>
1020        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#EducationalPropertiesCategory"/>
1021    </owl:ObjectProperty>
1022    <lom:MetaRolVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#validator"/>
1023    <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
           lom#version">
1024        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
               FunctionalProperty"/>
1025        <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
               slor.owl#LOMOWL_Record"/>
1026        <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
               lom#LangString"/>
1027        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#LifeCyclePropertiesCategory"/>
1028    </owl:ObjectProperty>
1029    <lom:DifficultyVocabularyItem rdf:about="http://slor.sourceforge.net
           /ontology/lom#very_difficult"/>
1030    <lom:DifficultyVocabularyItem rdf:about="http://slor.sourceforge.net
           /ontology/lom#very_easy"/>
1031    <lom:LevelVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#very_high"/>
1032    <lom:LevelVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#very_low"/>
1033    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           Vocabulary">
1034        <owl:equivalentClass rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#conceptual_Link"/>
```

```
1035        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#Element"/>
1036    </owl:Class>
1037    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           VocabularyItem">
1038        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#Element"/>
1039    </owl:Class>
1040    <lom:YesNoVocabularyItem rdf:about="http://slor.sourceforge.net/
           ontology/lom#yes"/>
1041    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           YesNoVocabulary">
1042        <owl:equivalentClass>
1043            <owl:Restriction>
1044                <owl:onProperty rdf:resource="http://slor.sourceforge.net
                       /ontology/slor.owl#ref_Term"/>
1045                <owl:someValuesFrom rdf:resource="http://slor.sourceforge
                       .net/ontology/lom#YesNoVocabularyItem"/>
1046            </owl:Restriction>
1047        </owl:equivalentClass>
1048        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#LOMv1.0"/>
1049    </owl:Class>
1050    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/lom#
           YesNoVocabularyItem">
1051        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/lom#VocabularyItem"/>
1052    </owl:Class>
1053    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
           LOM_Record">
1054        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#LearningMetadataRecord"/>
1055    </owl:Class>
1056    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
           LOMOWL_Record">
1057        <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#LearningMetadataRecord"/>
1058    </owl:Class>
1059    <lom:MetadataSchema rdf:about="http://slor.sourceforge.net/ontology/
           slor.owl#LOMv1.0"/>
```

```
1060    <lom:AggregationLevelVocabulary rdf:about="http://slor.sourceforge.
           net/ontology/slor.owl#LOMv1.0-Aggregation_Level_2">
1061        <ontologySchemaLink rdf:datatype="http://www.w3.org/2001/
               XMLSchema#string">LOMv1.0</ontologySchemaLink>
1062        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               AggregationLevel2"/>
1063    </lom:AggregationLevelVocabulary>
1064    <lom:AggregationLevelVocabulary rdf:about="http://slor.sourceforge.
           net/ontology/slor.owl#LOMv1.0-AggregationLevel2">
1065        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               AggregationLevel2"/>
1066    </lom:AggregationLevelVocabulary>
1067    <lom:ContextVocabulary rdf:about="http://slor.sourceforge.net/
           ontology/slor.owl#LOMv1.0-Context">
1068        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               higher_education"/>
1069    </lom:ContextVocabulary>
1070    <lom:DifficultVocabulary rdf:about="http://slor.sourceforge.net/
           ontology/slor.owl#LOMv1.0-Difficult">
1071        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               difficult"/>
1072    </lom:DifficultVocabulary>
1073    <lom:PurposeVocabulary rdf:about="http://slor.sourceforge.net/
           ontology/slor.owl#LOMv1.0-Discipline">
1074        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               discipline"/>
1075    </lom:PurposeVocabulary>
1076    <lom:StatusVocabulary rdf:about="http://slor.sourceforge.net/ontology
           /slor.owl#LOMv1.0-Draft">
1077        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               draft"/>
1078    </lom:StatusVocabulary>
1079    <lom:InterativityTypeVocabulary rdf:about="http://slor.sourceforge.
           net/ontology/slor.owl#LOMv1.0-Expositive">
1080        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               expositive"/>
1081    </lom:InterativityTypeVocabulary>
1082    <lom:SemanticDensityVocabulary rdf:about="http://slor.sourceforge.
           net/ontology/slor.owl#LOMv1.0-High">
```

```
1083        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               high"/>
1084    </lom:SemanticDensityVocabulary>
1085    <lom:RelationVocabulary rdf:about="http://slor.sourceforge.net/
           ontology/slor.owl#LOMv1.0-IsBasedOn">
1086        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               isbasedon"/>
1087    </lom:RelationVocabulary>
1088    <lom:StructureVocabulary rdf:about="http://slor.sourceforge.net/
           ontology/slor.owl#LOMv1.0-linear">
1089        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               linear"/>
1090    </lom:StructureVocabulary>
1091    <lom:LearningResourceVocabulary rdf:about="http://slor.sourceforge.
           net/ontology/slor.owl#LOMv1.0-Narrative_Text">
1092        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               narrative_text"/>
1093    </lom:LearningResourceVocabulary>
1094    <lom:YesNoVocabulary rdf:about="http://slor.sourceforge.net/ontology/
           slor.owl#LOMv1.0-Yes">
1095        <ref_Term rdf:resource="http://slor.sourceforge.net/ontology/lom#
               yes"/>
1096    </lom:YesNoVocabulary>
1097    <lom:Identifier rdf:about="http://slor.sourceforge.net/ontology/slor.
           owl#LOMWG">
1098        <lom:entry rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1099            >The joint ARIADNE-IMS submission to the LOM WG.</lom:entry
                >
1100        <lom:catalog rdf:datatype="http://www.w3.org/2001/XMLSchema#
               string"
1101            >LOM WG12&amp;apos;s set of base documents.</lom:catalog>
1102    </lom:Identifier>
```

B.1.2. Representación de una instancia LOM-OWL

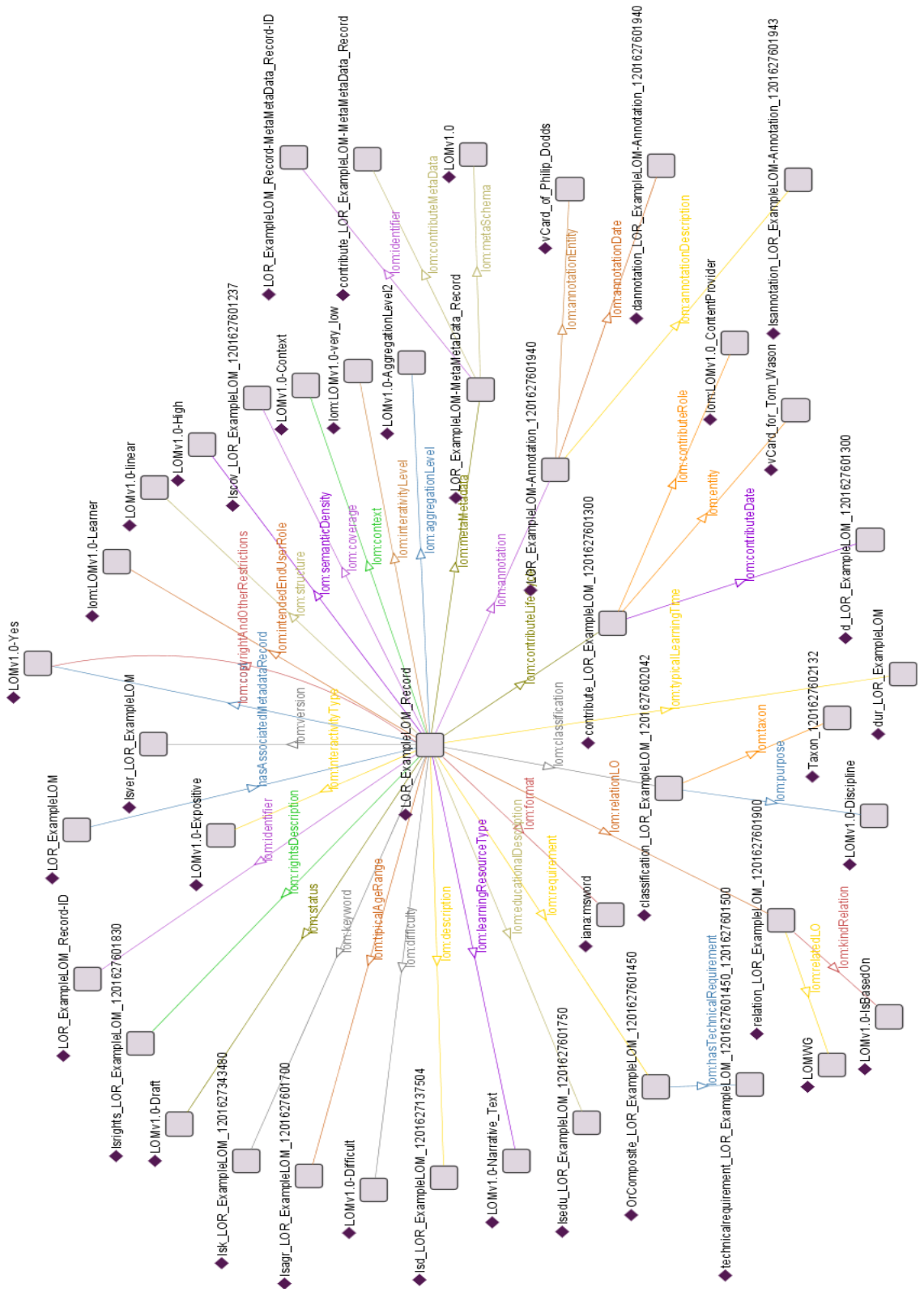

Figura B.1: Instancia de la clase "*LOMOWL_Record*"

```
<LOR_LOMOWL rdf:about="http://slor.sourceforge.net/ontology/slor.owl#LOR_ExampleLOM">
    <fileFromURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf</fileFromURI>
    <Identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >LOM_standart_draft01</Identifier>
    <Title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Draft Standard for Learning Object Metadata</Title>
    <usedIn rdf:resource="http://www.cyc.com/2004/06/04/cyc/#Learning" />
    <hasAssociatedMetadataRecord rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#LOR_ExampleLOM_Record"/>
</LOR_LOMOWL>

<LOMOWL_Record rdf:about="http://slor.sourceforge.net/ontology/slor.owl#LOR_ExampleLOM_Record">
    <lom:size rdf:datatype="http://www.w3.org/2001/XMLSchema#long">210000</lom:size>
    <lom:context rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#LOMv1.0-Context"/>
    <lom:keyword rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#lsk_LOR_ExampleLOM_1201627343480"/>
    <lom:interativityLevel rdf:resource="http://slor.sourceforge.net/ontology/lom#LOMv1.0-very_low"/>
    <lom:contributeLifeCycle rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#contribute_LOR_ExampleLOM_1201627601300"/>
    <lom:learningResourceType rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#LOMv1.0-Narrative_Text"/>
    <lom:version rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#lsver_LOR_ExampleLOM"/>
    <lom:coverage rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#lscov_LOR_ExampleLOM_1201627601237"/>
    <lom:format rdf:resource="http://www.iana.org/assignments/media-types/#msword"/>
    <lom:typicalAgeRange rdf:resource="http://slor.sourceforge.net/ontology/slor.owl#lsagr_LOR_ExampleLOM_1201627601700"/>
```

```
23        <lom:aggregationLevel rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#LOMv1.0-AggregationLevel2"/>
24        <lom:language rdf:resource="http://slor.sourceforge.net/ontology/
              slor.owl#English"/>
25        <lom:intendedEndUserRole rdf:resource="http://slor.sourceforge.
              net/ontology/lom#LOMv1.0-Learner"/>
26        <lom:rightsDescription rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#lsrights_LOR_ExampleLOM_1201627601830"/>
27        <lom:educationalDescription rdf:resource="http://slor.
              sourceforge.net/ontology/slor.owl#
              lsedu_LOR_ExampleLOM_1201627601750"/>
28        <lom:requirement rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#OrComposite_LOR_ExampleLOM_1201627601450"/>
29        <lom:typicalLearningTime rdf:resource="http://slor.sourceforge.
              net/ontology/slor.owl#dur_LOR_ExampleLOM"/>
30        <lom:description rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#lsd_LOR_ExampleLOM_1201627137504"/>
31        <lom:relationLO rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#relation_LOR_ExampleLOM_1201627601900"/>
32        <lom:status rdf:resource="http://slor.sourceforge.net/ontology/
              slor.owl#LOMv1.0-Draft"/>
33        <lom:metaMetadata rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#LOR_ExampleLOM-MetaMetaData_Record"/>
34        <lom:cost rdf:resource="http://slor.sourceforge.net/ontology/slor
              .owl#LOMv1.0-Yes"/>
35        <lom:annotation rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#LOR_ExampleLOM-Annotation_1201627601940"/>
36        <lom:copyrightAndOtherRestrictions rdf:resource="http://slor.
              sourceforge.net/ontology/slor.owl#LOMv1.0-Yes"/>
37        <lom:interactivityType rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#LOMv1.0-Expositive"/>
38        <lom:identifier rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#LOR_ExampleLOM_Record-ID"/>
39        <lom:userLanguage rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#English"/>
40        <lom:structure rdf:resource="http://slor.sourceforge.net/ontology
              /slor.owl#LOMv1.0-linear"/>
41        <lom:semanticDensity rdf:resource="http://slor.sourceforge.net/
              ontology/slor.owl#LOMv1.0-High"/>
```

```
42          <lom:difficulty rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#LOMv1.0-Difficult"/>
43          <lom:classification rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#classification_LOR_ExampleLOM_1201627602042"/>
44      </LOMOWL_Record>
45
46      <lom:Identifier rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#LOR_ExampleLOM_Record-ID">
47          <lom:entry rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >P1484.12.1</lom:entry>
48          <lom:catalog rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string">IEEE</lom:catalog>
49      </lom:Identifier>
50
51       <lom:MetaMetaData_Record rdf:about="http://slor.sourceforge.net/
            ontology/slor.owl#LOR_ExampleLOM-MetaMetaData_Record">
52          <lom:contributeMetaData rdf:resource="http://slor.sourceforge.
                net/ontology/slor.owl#contribute_LOR_ExampleLOM-
                MetaMetaData_Record"/>
53          <lom:metaSchema rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#LOMv1.0"/>
54          <lom:identifier rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#LOR_ExampleLOM_Record-MetaMetaData_Record-ID"/
                >
55      </lom:MetaMetaData_Record>
56
57      <lom:Identifier rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#LOR_ExampleLOM_Record-MetaMetaData_Record-ID">
58          <lom:entry rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >2000121601</lom:entry>
59          <lom:catalog rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"
60              >Erik Duval&amp;apos;s set of metadata records.</lom:catalog>
61      </lom:Identifier>
62
63      <lom:Contribute rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#contribute_LOR_ExampleLOM-MetaMetaData_Record">
64          <lom:contributeRole rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0-Creator"/>
```

```
65          <lom:contributeDate rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#d_LOR_ExampleLOM_Record-
                MetaMetaData_Record_1201627601400"/>
66          <lom:entity rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#vCard_for_Erik_Duval"/>
67      </lom:Contribute>
68
69      <lom:DateTime rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#d_LOR_ExampleLOM_Record-MetaMetaData_Record_1201627601400">
70          <lom:dateTimeValue rdf:datatype="http://www.w3.org/2001/XMLSchema
                #dateTime"
71              >2008-01-30T00:00:00</lom:dateTimeValue>
72          <lom:dateTimeDescription rdf:resource="http://slor.sourceforge.
                net/ontology/slor.owl#lsdatemetadatacontrib_"/>
73      </lom:DateTime>
74
75      <lom:Contribute rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#contribute_LOR_ExampleLOM_1201627601300">
76          <lom:contributeRole rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0_ContentProvider"/>
77          <lom:contributeDate rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#d_LOR_ExampleLOM_1201627601300"/>
78          <lom:entity rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#vCard_for_Tom_Wason"/>
79      </lom:Contribute>
80
81      <lom:DateTime rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#d_LOR_ExampleLOM_1201627601300">
82          <lom:dateTimeValue rdf:datatype="http://www.w3.org/2001/XMLSchema
                #dateTime"
83              >2008-01-01T00:00:00</lom:dateTimeValue>
84          <lom:dateTimeDescription rdf:resource="http://slor.sourceforge.
                net/ontology/slor.owl#lsdatecontrib_LOR_ExampleLOM_1201627601350
                "/>
85      </lom:DateTime>
86
87      <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsver_LOR_ExampleLOM">
88          <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
```

```
89          <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1.0
               </value>
90      </lom:LangString>
91
92      <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
           owl#lsrights_LOR_ExampleLOM_1201627601830">
93          <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
               en</language>
94          <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
95              >Copyright &#169; 2000 by the Institute of Electrical and
                   Electronics Engineers, Inc. 3 Park Avenue New York, NY
                   10016-5997, USA All rights reserved. This is an
                   unapproved draft of a proposed IEEE Standard, subject to
                    change. Permission is hereby granted for IEEE Standards
                    Committee participants to reproduce this document for
                   purposes of IEEE standardization activities. If this
                   document is to be submitted to ISO or IEC, notification
                   shall be given to the IEEE Copyright Administrator.
                   Permission is also granted for member bodies and
                   technical committees of ISO and IEC to reproduce this
                   document for purposes of developing a national position.
                    Other entities seeking permission to reproduce this
                   document for standardization or other activities, or to
                   reproduce portions of this document for these or other
                   uses, must contact the IEEE Standards Department for the
                    appropriate license. Use of information contained in
                   this unapproved draft is at your own risk. IEEE
                   Standards Department Copyright and Permissions 445 Hoes
                   Lane, P.O. Box 1331 Piscataway, NJ 08855-1331, USA</
                   value>
96      </lom:LangString>
97
98
99      <lom:Relation rdf:about="http://slor.sourceforge.net/ontology/slor.
           owl#relation_LOR_ExampleLOM_1201627601900">
100         <lom:kindOfRelation rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#LOMv1.0-IsBasedOn"/>
101         <lom:relatedLO rdf:resource="http://slor.sourceforge.net/ontology
               /slor.owl#LOMWG"/>
102     </lom:Relation>
```

```
103
104     <lom:OrComposite rdf:about="http://slor.sourceforge.net/ontology/slor
            .owl#OrComposite_LOR_ExampleLOM_1201627601450">
105         <lom:hasTechnicalRequirement rdf:resource="http://slor.
                sourceforge.net/ontology/slor.owl#
                technicalrequirement_LOR_ExampleLOM_1201627601450_1201627601500"
                />
106     </lom:OrComposite>
107
108     <lom:TechnicalRequirement rdf:about="http://slor.sourceforge.net/
            ontology/slor.owl#
            technicalrequirement_LOR_ExampleLOM_1201627601450_1201627601500">
109         <lom:requirementName rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0-MS_Windows"/>
110         <lom:requirementType rdf:resource="http://slor.sourceforge.net/
                ontology/lom#LOMv1.0-Operating_System"/>
111     </lom:TechnicalRequirement>
112
113     <lom:Annotation
114         rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
                LOR_ExampleLOM-Annotation_1201627601940">
115         <lom:annotationDescription rdf:resource="http://slor.sourceforge
                .net/ontology/slor.owl#lsannotation_LOR_ExampleLOM-
                Annotation_1201627601943"/>
116         <lom:annotationEntity rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#vCard_of_Philip_Dodds"/>
117         <lom:annotationDate rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#dannotation_LOR_ExampleLOM-
                Annotation_1201627601940"/>
118     </lom:Annotation>
119
120     <lom:Classification rdf:about="http://slor.sourceforge.net/ontology/
            slor.owl#classification_LOR_ExampleLOM_1201627602042">
121         <lom:purpose rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#LOMv1.0-Discipline"/>
122         <lom:taxon rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#Taxon_1201627602132"/>
123     </lom:Classification>
124
```

```
125     <lom:Taxon rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
            Taxon_1201627602132">
126         <lom:taxonSource rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#source_Taxon_1201627602132"/>
127         <lom:refTaxon rdf:resource="http://slor.sourceforge.net/ontology/
                slor.owl#id_Taxon_1201627602132"/>
128     </lom:Taxon>
129
130     <lom:DateTime rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#dannotation_LOR_ExampleLOM-Annotation_1201627601940">
131         <lom:dateTimeValue rdf:datatype="http://www.w3.org/2001/XMLSchema
                #dateTime"
132             >2000-12-17T00:00:00</lom:dateTimeValue>
133         <lom:dateTimeDescription rdf:resource="http://slor.sourceforge.
                net/ontology/slor.owl#lsdannotation_LOR_ExampleLOM-
                Annotation_1201627601940_1201627601945"/>
134     </lom:DateTime>
135
136     <lom:TaxonID rdf:about="http://slor.sourceforge.net/ontology/slor.owl
            #id_Taxon_1201627602132">
137         <Identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string
                ">en</Identifier>
138         <lom:taxonEntry rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#lstaxonid_Taxon_1201627602132-
                Information_Science"/>
139     </lom:TaxonID>
140
141     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#source_Taxon_1201627602132">
142         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
143         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
144             >A great taxonomic source.</value>
145     </lom:LangString>
146
147     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsagr_LOR_ExampleLOM_1201627601700">
148         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">18-
                </value>
149     </lom:LangString>
```

```
150
151     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsannotation_LOR_ExampleLOM-Annotation_1201627601943">
152         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
153             >I have read this with great attention and extreme interest
                    . I think this is great!</value>
154     </lom:LangString>
155
156     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lscov_LOR_ExampleLOM_1201627601237">
157         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
158         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                contemporary</value>
159     </lom:LangString>
160
161     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsd_LOR_ExampleLOM_1201627137504">
162         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
163         <value xml:lang="en"
164             >Metadata is information about an object, be it physical or
                    digital. As the number of objects grows exponentially
                and our needs for learning expand equally dramatically,
                the lack of information or metadata about objects places
                 a critical and fundamental constraint on our ability to
                 discover, manage and use objects. This standard
                addresses this problem by defining a structure for
                interoperable descriptions of learning objects.</value>
165     </lom:LangString>
166
167     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsdannotation_LOR_ExampleLOM-
            Annotation_1201627601940_1201627601945">
168         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                20001217</value>
169     </lom:LangString>
170
171     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsdatecontrib_LOR_ExampleLOM_1201627601350">
```

```
172         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
173         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                1998</value>
174     </lom:LangString>
175
176     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsdatemetadatacontrib_">
177         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
178         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                2000-12-16</value>
179     </lom:LangString>
180
181     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsedu_LOR_ExampleLOM_1201627601750">
182         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
183         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
184             >Comments on how this resource is to be used.</value>
185     </lom:LangString>
186
187     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lsk_LOR_ExampleLOM_1201627343480">
188         <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
189         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                metadata</value>
190     </lom:LangString>
191
192     <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.
            owl#lstaxonid_Taxon_1201627602132-Information_Science">
193         <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
194             >Information Science</value>
195     </lom:LangString>
```

B.1.3. Ejemplo - Expresiones semánticas en el campo descripción

```
<LOMOWL_Record rdf:ID="LOR_ExampleLOM_Record">
   ...
   <lom:description rdf:resource="#ocsld_LOR_ExampleLOM_1201627137558"/>
   <lom:description rdf:resource="#ocsld_LOR_ExampleLOM_1201627137560"/>
   <lom:description rdf:resource="#ocsld_LOR_ExampleLOM_1201627137565"/>
   <lom:description rdf:resource="#ocsld_LOR_ExampleLOM_1201627137556"/>
   <lom:description rdf:resource="#owlsld_LOR_ExampleLOM_1201627137593"/>
   ...
<\LOMOWL_Record>

    <owl_SemanticLink rdf:about="http://slor.sourceforge.net/ontology/
       slor.owl#owlsld_LOR_ExampleLOM_1201627137593">
    <ontologySchemaLink rdf:datatype="http://www.w3.org/2001/XMLSchema#
       string">
       http://slor.sourceforge.net/ontology/edutest.owl
    </ontologySchemaLink>
    <property_value rdf:resource="http://slor.sourceforge.net/ontology/
       edutest.owl#usesin"/>
    <property rdf:resource="http://slor.sourceforge.net/ontology/edutest.
       owl#Catalogue_Learning_Objects"/>
    </owl_SemanticLink>

    <oc_ArgPredicate rdf:about="http://slor.sourceforge.net/ontology/slor
       .owl#LORID_arg1">
       <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1<
          /norderArg>
       <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          LOR_ExampleLOM</termArg>
    </oc_ArgPredicate>

    <oc_ArgPredicate rdf:about="http://slor.sourceforget.net/ontology/
       slor.owl#ocsld_LOR_ExampleLOM_1201627137556_arg2">
       <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2<
          /norderArg>
       <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >SpecificationDocument</termArg>
    </oc_ArgPredicate>

```

```
30    <oc_SemanticLink rdf:about="http://slor.sourceforge.net/ontology/slor
          .owl#ocsld_LOR_ExampleLOM_1201627137558">
31        <arity rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</
              arity>
32        <oc_Predicate rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string">isa</oc_Predicate>
33    </oc_SemanticLink>
34
35    <oc_ArgPredicate rdf:about="http://slor.sourceforge.net/ontology/slor
          .owl#ocsld_LOR_ExampleLOM_1201627137558_arg1">
36        <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1<
              /norderArg>
37        <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
38            >LOM-Record-Structure</termArg>
39    </oc_ArgPredicate>
40
41    <oc_ArgPredicate rdf:about="http://slor.sourceforge.net/ontology/slor
          .owl#ocsld_LOR_ExampleLOM_1201627137558_arg2">
42        <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2<
              /norderArg>
43        <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              LogicalSchema</termArg>
44    </oc_ArgPredicate>
45
46    <oc_SemanticLink rdf:about="http://slor.sourceforge.net/ontology/slor
          .owl#ocsld_LOR_ExampleLOM_1201627137560">
47        <arity rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</
              arity>
48        <oc_Predicate rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string">thingSpecified</oc_Predicate>
49        <oc_Args rdf:resource="http://slor.sourceforge.net/ontology/slor.
              owl#ocsld_LOR_ExampleLOM_1201627137560_arg1"/>
50    </oc_SemanticLink>
51
52    <oc_ArgPredicate rdf:about="http://slor.sourceforge.net/ontology/slor
          .owl#ocsld_LOR_ExampleLOM_1201627137560_arg1">
53        <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1<
              /norderArg>
54        <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
55            >LOM-Record-Structure</termArg>
```

```
56      </oc_ArgPredicate>
57      <oc_SemanticLink rdf:about="http://slor.sourceforge.net/ontology/slor
            .owl#ocsld_LOR_ExampleLOM_1201627137565">
58          <arity rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</
                arity>
59          <oc_Predicate rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string">usedIn</oc_Predicate>
60          <oc_Args rdf:resource="http://slor.sourceforge.net/ontology/slor.
                owl#ocsld_LOR_ExampleLOM_1201627137565_arg1"/>
61      </oc_SemanticLink>
62      <oc_ArgPredicate rdf:about="http://slor.sourceforge.net/ontology/slor
            .owl#ocsld_LOR_ExampleLOM_1201627137565_arg1">
63          <norderArg rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1<
                /norderArg>
64          <termArg rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
65              >MakingSomethingAbstractAvailable</termArg>
66      </oc_ArgPredicate>
```

B.2. Representación de un registro MERLOT

B.2.1. Esquema del registro de metadatos

```
1       <owl:Class rdf:ID="LOR_Merlot">
2          <owl:equivalentClass>
3              <owl:Class>
4                  <owl:intersectionOf rdf:parseType="Collection">
5                      <owl:Restriction>
6                          <owl:onProperty rdf:resource="#
                               hasAssociatedMetadataRecord"/>
7                          <owl:someValuesFrom rdf:resource="#MERLOT_Record"
                               />
8                      </owl:Restriction>
9                      <owl:Class rdf:about="#LearningObject-
                           AsAnythingDigital"/>
10                 </owl:intersectionOf>
11             </owl:Class>
12         </owl:equivalentClass>
13      </owl:Class>
14
```

```
15    <owl:Class rdf:ID="MERLOT_Record">
16        <rdfs:subClassOf rdf:resource="#LearningMetadataRecord"/>
17    </owl:Class>
18
19    <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
          merlot#category">
20        <rdfs:domain rdf:resource="#MERLOT_Record"/>
21        <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
              merlot#MerlotCategory"/>
22        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
              ontology/merlot#merlotMetadataRecordProperty"/>
23    </owl:ObjectProperty>
24    <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
          /merlot#CopyRight">
25        <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
              merlot#Merlot_InformationDescriptor"/>
26        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
              "/>
27        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
              ontology/merlot#merlotMetadataRecordProperty"/>
28    </owl:DatatypeProperty>
29    <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
          /merlot#costInvolved">
30        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
              FunctionalProperty"/>
31        <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
              merlot#Merlot_InformationDescriptor"/>
32        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
              boolean"/>
33        <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
              ontology/merlot#merlotMetadataRecordProperty"/>
34    </owl:DatatypeProperty>
35    <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
          /merlot#creativeCommons">
36        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
              FunctionalProperty"/>
37        <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
              merlot#Merlot_InformationDescriptor"/>
38        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
              boolean"/>
```

```
39          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
40      </owl:DatatypeProperty>
41      <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#date_Added">
42          <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
43          <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_MetaMetaDataDescriptor"/>
44          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                dateTime"/>
45          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
46      </owl:DatatypeProperty>
47      <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#date_Modified">
48          <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
49          <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_MetaMetaDataDescriptor"/>
50          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                dateTime"/>
51          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
52      </owl:DatatypeProperty>
53      <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Education">
54          <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
55      </owl:Class>
56      <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#education_center">
57          <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_LOCreator"/>
58          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
59      </owl:DatatypeProperty>
60      <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#email">
```

```
61          <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_LOCreator"/>
62          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI
                "/>
63          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
64      </owl:DatatypeProperty>
65      <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#format">
66          <rdfs:domain rdf:resource="#MERLOT_Record"/>
67          <rdfs:range rdf:resource="#TechnicalFormat"/>
68          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
69      </owl:ObjectProperty>
70      <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Humanities">
71          <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
72      </owl:Class>
73      <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#information">
74          <rdfs:domain rdf:resource="#MERLOT_Record"/>
75          <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_InformationDescriptor"/>
76          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
77      </owl:ObjectProperty>
78      <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#language">
79          <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_InformationDescriptor"/>
80          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
81          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
82      </owl:DatatypeProperty>
83      <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#lorauthor">
84          <rdfs:domain rdf:resource="#MERLOT_Record"/>
```

```
85          <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_LOCreator"/>
86          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
87      </owl:ObjectProperty>
88      <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#LORauthorName">
89          <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_LOCreator"/>
90          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
91          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
92      </owl:DatatypeProperty>
93      <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#material_Type">
94          <rdfs:domain rdf:resource="#MERLOT_Record"/>
95          <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#MerlotMaterialType"/>
96          <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
97      </owl:ObjectProperty>
98      <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Mathematics">
99          <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#Mathematics_and_Statistics"/>
100     </owl:Class>
101     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Mathematics_and_Statistics">
102         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
103     </owl:Class>
104     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Merlot_InformationDescriptor">
105         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotOntologyElement"/>
106     </owl:Class>
107     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Merlot_LOCreator">
```

```
108         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotOntologyElement"/>
109     </owl:Class>
110     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Merlot_MetaMetaDataDescriptor">
111         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotOntologyElement"/>
112     </owl:Class>
113     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            MerlotCategory">
114         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotType"/>
115     </owl:Class>
116     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            MerlotMaterialType">
117         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotType"/>
118     </owl:Class>
119     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#merlotMetadataRecordProperty"/>
120     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            MerlotOntologyElement"/>
121     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            MerlotType">
122         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotOntologyElement"/>
123     </owl:Class>
124     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#metametadata">
125         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
126         <rdfs:domain rdf:resource="#MERLOT_Record"/>
127         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_MetaMetaDataDescriptor"/>
128         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
129     </owl:ObjectProperty>
130     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#primaryAudience">
```

```
131         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_InformationDescriptor"/>
132         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
133         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
134     </owl:DatatypeProperty>
135     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Review">
136         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotOntologyElement"/>
137     </owl:Class>
138     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#review">
139         <rdfs:domain rdf:resource="#MERLOT_Record"/>
140         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Review"/>
141         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
142     </owl:ObjectProperty>
143     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#sourceCode">
144         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
145         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_InformationDescriptor"/>
146         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                boolean"/>
147         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
148     </owl:DatatypeProperty>
149     <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            merlot#submitter">
150         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                FunctionalProperty"/>
151         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_MetaMetaDataDescriptor"/>
152         <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#User"/>
```

```
153         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
154     </owl:ObjectProperty>
155     <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
            /merlot#technicalRequirements">
156         <rdfs:domain rdf:resource="http://slor.sourceforge.net/ontology/
                merlot#Merlot_InformationDescriptor"/>
157         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
                "/>
158         <rdfs:subPropertyOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#merlotMetadataRecordProperty"/>
159     </owl:DatatypeProperty>
160     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#User
            ">
161         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotOntologyElement"/>
162     </owl:Class>
163
164     <!-- Taxonomy example -->
165     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#Arts
            ">
166         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
167     </owl:Class>
168     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Business">
169         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
170     </owl:Class>
171     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            General_and_Liberal_Arts_Math">
172         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#Mathematics"/>
173     </owl:Class>
174     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            History_of_Mathematics">
175         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#General_and_Liberal_Arts_Math"/>
176     </owl:Class>
```

```
177     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Science_and_Technology">
178         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
179     </owl:Class>
180     <owl:Class rdf:about="http://slor.sourceforge.net/ontology/merlot#
            Social_Sciences">
181         <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#MerlotCategory"/>
182     </owl:Class>
```

B.2.2. Representación de un individual

```
1     <LOR_Merlot rdf:ID="LOR_Merlot_89142">
2         <Title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
3             >MacTutor History of Mathematics Archive</Title>
4         <fileFromURI rdf:datatype="http://www.w3.org/2001/XMLSchema#
                anyURI"
5             >http://www-history.mcs.st-and.ac.uk/history/</fileFromURI>
6         <Identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string
                ">89142</Identifier>
7         <usedIn rdf:resource="http://www.cyc.com/2004/06/04/cyc/#
                MathemathicsLearningActivity"/>
8         <hasAssociatedMetadataRecord rdf:resource="#MERLOT_Record_89142"
                />
9     </LOR_Merlot>
10
11    <MERLOT_Record rdf:ID="MERLOT_Record_89142">
12        <merlot:review rdf:resource="#Review_01"/>
13        <merlot:format rdf:resource="#HTML_Text"/>
14        <merlot:lorauthor rdf:resource="#John_OConnor"/>
15        <merlot:lorauthor rdf:resource="#Edmund_Robertson"/>
16        <merlot:information rdf:resource="#
                Merlot_InformationDescriptor_89142"/>
17        <merlot:material_Type rdf:resource="#Reference_Material"/>
18        <merlot:category rdf:resource="http://slor.sourceforge.net/
                ontology/merlot#History_of_Mathematics"/>
19        <merlot:metametadata rdf:resource="#
                Merlot_MetaMetaDataDescriptor_89142"/>
```

```
20      </MERLOT_Record>
21
22      <merlot:User rdf:ID="James_Rutledge"/>
23
24      <merlot:Merlot_LOCreator rdf:ID="John_OConnor">
25          <merlot:email rdf:datatype="http://www.w3.org/2001/XMLSchema#
                anyURI"
26              >joc@st-andrews.ac.uk</merlot:email>
27          <merlot:LORauthorName rdf:datatype="http://www.w3.org/2001/
                XMLSchema#string"
28              >Dr. John O&amp;apos;Connor</merlot:LORauthorName>
29      </merlot:Merlot_LOCreator>
30
31      <merlot:Merlot_LOCreator rdf:ID="Edmund_Robertson">
32          <merlot:email rdf:datatype="http://www.w3.org/2001/XMLSchema#
                anyURI"
33              >efr@st-andrews.ac.uk</merlot:email>
34          <merlot:LORauthorName rdf:datatype="http://www.w3.org/2001/
                XMLSchema#string"
35              >Prof. Edmund Robertson</merlot:LORauthorName>
36      </merlot:Merlot_LOCreator>
37
38      <merlot:Merlot_MetaMetaDataDescriptor rdf:ID="
            Merlot_MetaMetaDataDescriptor_89142">
39          <merlot:date_Added rdf:datatype="http://www.w3.org/2001/XMLSchema
                #dateTime"
40              >2001-04-19T00:00:00</merlot:date_Added>
41          <merlot:date_Modified rdf:datatype="http://www.w3.org/2001/
                XMLSchema#dateTime"
42              >2007-12-10T00:00:00</merlot:date_Modified>
43          <merlot:submitter rdf:resource="#James_Rutledge"/>
44      </merlot:Merlot_MetaMetaDataDescriptor>
45
46      <merlot:Merlot_InformationDescriptor rdf:ID="
            Merlot_InformationDescriptor_89142">
47          <merlot:technicalRequirements rdf:datatype="http://www.w3.org
                /2001/XMLSchema#string"
```

```
48          > Web browser for most content; Java-enabled browser for
                limited number of interactive Java applets concerning
                famous mathematical curves.</
                merlot:technicalRequirements>
49      <merlot:primaryAudience rdf:datatype="http://www.w3.org/2001/
            XMLSchema#string">High School</merlot:primaryAudience>
50      <merlot:primaryAudience rdf:datatype="http://www.w3.org/2001/
            XMLSchema#string">College General</merlot:primaryAudience>
51      <merlot:CopyRight rdf:datatype="http://www.w3.org/2001/XMLSchema#
            string">yes</merlot:CopyRight>
52      <merlot:language rdf:datatype="http://www.w3.org/2001/XMLSchema#
            string">English</merlot:language>
53      <merlot:sourceCode rdf:datatype="http://www.w3.org/2001/XMLSchema
            #boolean">false</merlot:sourceCode>
54      <merlot:costInvolved rdf:datatype="http://www.w3.org/2001/
            XMLSchema#boolean">false</merlot:costInvolved>
55  </merlot:Merlot_InformationDescriptor>
56
57  <merlot:Review rdf:ID="Review_01">
58      <!-- review metadata fields
59      -->
60  </merlot:Review>
```

B.3. Representación registro de metadatos NDSL

B.3.1. Esquema del registro de metadatos

```
1   <owl:Class rdf:ID="LOR_NDSL">
2       <owl:equivalentClass>
3           <owl:Class>
4               <owl:intersectionOf rdf:parseType="Collection">
5                   <owl:Restriction>
6                       <owl:onProperty rdf:resource="#
                            hasAssociatedMetadataRecord"/>
7                       <owl:someValuesFrom rdf:resource="#NDSL_Record"/>
8                   </owl:Restriction>
9                   <owl:Class rdf:about="#LearningObject-
                        AsAnythingDigital"/>
10              </owl:intersectionOf>
```

```
11          </owl:Class>
12      </owl:equivalentClass>
13  </owl:Class>
14
15  <owl:Class rdf:ID="NDSL_Record">
16      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/ndsl#NDSLOntologyElement"/>
17      <rdfs:subClassOf rdf:resource="#LearningMetadataRecord"/>
18  </owl:Class>
19
20  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        ndsl#description">
21      <rdfs:domain rdf:resource="#NDSL_Record"/>
22      <rdfs:range rdf:resource="#descriptiveProperty"/>
23  </owl:ObjectProperty>
24  <owl:DatatypeProperty rdf:about="http://slor.sourceforge.net/ontology
        /ndsl#has_Audience">
25      <rdfs:domain rdf:resource="#NDSL_Record"/>
26      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string
            "/>
27  </owl:DatatypeProperty>
28  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        ndsl#keyword">
29      <rdfs:domain rdf:resource="#NDSL_Record"/>
30      <rdfs:range rdf:resource="#conceptual_Link"/>
31  </owl:ObjectProperty>
32  <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
        ndsl#language">
33      <rdfs:domain rdf:resource="#NDSL_Record"/>
34      <rdfs:range rdf:resource="http://www.cyc.com/2004/06/04/cyc/#
            LivingLanguage"/>
35  </owl:ObjectProperty>
36  <owl:Class rdf:about="http://slor.sourceforge.net/ontology/ndsl#
        NDSLOntologyElement"/>
37  <owl:Class rdf:about="http://slor.sourceforge.net/ontology/ndsl#
        Publisher">
38      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/ndsl#NDSLOntologyElement"/>
39  </owl:Class>
```

```
40      <owl:Class rdf:about="http://slor.sourceforge.net/ontology/ndsl#
            ResourceType">
41          <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
                ontology/ndsl#NDSLOntologyElement"/>
42      </owl:Class>
43      <owl:ObjectProperty rdf:about="http://slor.sourceforge.net/ontology/
            ndsl#resourceType">
44          <rdfs:domain rdf:resource="#NDSL_Record"/>
45          <rdfs:range rdf:resource="http://slor.sourceforge.net/ontology/
                ndsl#ResourceType"/>
46      </owl:ObjectProperty>
```

B.3.2. Representación de un individual

```
1   <LOR_NDSL rdf:ID="LOR_NDSL_01">
2           <Title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
3               >How The Heart Works</Title>
4           <fileFromURI rdf:datatype="http://www.w3.org/2001/XMLSchema#
                anyURI"
5               >http://www.nhlbi.nih.gov/health/dci/Diseases/hhw/
                    hhw_whatis.html</fileFromURI>
6           <Identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string
                "
7               >http://www.nhlbi.nih.gov/health/dci/Diseases/hhw/
                    hhw_whatis.html</Identifier>
8           <hasAssociatedMetadataRecord rdf:resource="#NDSL_Record_01"/>
9           <usedIn rdf:resource="http://www.cyc.com/2004/06/04/cyc/#Learning"
                />
10  </LOR_NDSL>
11
12  <NDSL_Record rdf:ID="NDSL_Record_01">
13          <ndsl:has_Audience rdf:datatype="http://www.w3.org/2001/XMLSchema
                #string">Lung</ndsl:has_Audience>
14          <ndsl:has_Audience rdf:datatype="http://www.w3.org/2001/XMLSchema
                #string">Blood Institute</ndsl:has_Audience>
15          <ndsl:has_Audience rdf:datatype="http://www.w3.org/2001/XMLSchema
                #string">General Public</ndsl:has_Audience>
16          <ndsl:keyword rdf:resource="http://www.cyc.com/2004/06/04/cyc/#
                Heart"/>
```

```
        <ndsl:keyword rdf:resource="http://www.cyc.com/2004/06/04/cyc/#
            Anatomy-FieldOfStudy"/>
        <ndsl:resourceType rdf:resource="#Science_Materials"/>
        <ndsl:resourceType rdf:resource="#Reading_Materials"/>
        <ndsl:resourceType rdf:resource="#Still_Image"/>
        <ndsl:resourceType rdf:resource="#Charts_and_Diagrams"/>
        <ndsl:description rdf:resource="#ssd_LOR_NDSL_01_1200660986015"/>
        <ndsl:language rdf:resource="#English"/>
</NDSL_Record>
```

B.4. Representación registro de metadatos LOM-COREOWL

B.4.1. Esquema del registro de metadatos

```
    <owl:Class rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
        LOR_LOMCOREOWL">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="http://slor.
                            sourceforge.net/ontology/slor.owl#
                            hasAssociatedMetadataRecord"/>
                        <owl:someValuesFrom rdf:resource="http://slor.
                            sourceforge.net/ontology/slor.owl#
                            LOMCOREOWL_Record"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="http://www.cyc.com
                            /2004/06/04/cyc/#fileFromURI"/>
                        <owl:cardinality rdf:datatype="http://www.w3.org
                            /2001/XMLSchema#int">1</owl:cardinality>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="http://slor.
                            sourceforge.net/ontology/slor.owl#Title"/>
```

```
15                      <owl:cardinality rdf:datatype="http://www.w3.org
                           /2001/XMLSchema#int">1</owl:cardinality>
16                  </owl:Restriction>
17              </owl:intersectionOf>
18          </owl:Class>
19      </owl:equivalentClass>
20      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
           ontology/slor.owl#LOR_LOMOWL"/>
21  </owl:Class>
22
23   <owl:Class rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
        LOMCOREOWL_Record">
24      <owl:equivalentClass>
25          <owl:Class>
26              <owl:intersectionOf rdf:parseType="Collection">
27                  <owl:Restriction>
28                      <owl:onProperty rdf:resource="http://slor.
                           sourceforge.net/ontology/lom#contributeLifeCycle
                           "/>
29                      <owl:someValuesFrom rdf:resource="http://slor.
                           sourceforge.net/ontology/lom#Contribute"/>
30                  </owl:Restriction>
31                  <owl:Restriction>
32                      <owl:onProperty rdf:resource="http://slor.
                           sourceforge.net/ontology/lom#
                           copyrightAndOtherRestrictions"/>
33                      <owl:someValuesFrom rdf:resource="http://slor.
                           sourceforge.net/ontology/lom#Vocabulary"/>
34                  </owl:Restriction>
35                  <owl:Restriction>
36                      <owl:onProperty rdf:resource="http://slor.
                           sourceforge.net/ontology/lom#description"/>
37                      <owl:someValuesFrom rdf:resource="http://slor.
                           sourceforge.net/ontology/slor.owl#
                           descriptiveProperty"/>
38                  </owl:Restriction>
39                  <owl:Restriction>
40                      <owl:onProperty rdf:resource="http://slor.
                           sourceforge.net/ontology/lom#identifier"/>
```

```
41                      <owl:someValuesFrom rdf:resource="http://slor.
                            sourceforge.net/ontology/lom#Identifier"/>
42                  </owl:Restriction>
43                  <owl:Restriction>
44                      <owl:onProperty rdf:resource="http://slor.
                            sourceforge.net/ontology/lom#language"/>
45                      <owl:someValuesFrom rdf:resource="http://www.cyc.
                            com/2004/06/04/cyc/#HumanLanguage"/>
46                  </owl:Restriction>
47                  <owl:Restriction>
48                      <owl:onProperty rdf:resource="http://slor.
                            sourceforge.net/ontology/lom#metaMetadata"/>
49                      <owl:someValuesFrom rdf:resource="http://slor.
                            sourceforge.net/ontology/lom#MetaMetaData_Record
                            "/>
50                  </owl:Restriction>
51                  <owl:Restriction>
52                      <owl:onProperty rdf:resource="http://slor.
                            sourceforge.net/ontology/lom#rightsDescription"/
                            >
53                      <owl:someValuesFrom rdf:resource="http://slor.
                            sourceforge.net/ontology/lom#LangString"/>
54                  </owl:Restriction>
55              </owl:intersectionOf>
56          </owl:Class>
57      </owl:equivalentClass>
58      <rdfs:subClassOf rdf:resource="http://slor.sourceforge.net/
            ontology/slor.owl#LOMOWL_Record"/>
59  </owl:Class>
```

B.4.2. Representación de una instancia

```
1  <LOR_LOMCOREOWL rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
       LOR_LOMCOREOWL_DNAFromTheBeginning">
2      <Identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string
           "
3          >LOR_DNAFromTheBeginning</Identifier>
4      <oc:fileFromURI rdf:datatype="http://www.w3.org/2001/XMLSchema#
           anyURI"
```

```
5           >http://www.dnaftb.org/dnaftb/</oc:fileFromURI>
6       <Title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
7           >DNA from the Beginning</Title>
8       <usedIn rdf:resource="http://www.cyc.com/2004/06/04/cyc/#Learning"
            />
9       <hasAssociatedMetadataRecord rdf:resource="http://slor.
            sourceforge.net/ontology/slor.owl#LOR_DNAFromTheBeginning"/>
10  </LOR_LOMCOREOWL>
11
12   <LOMCOREOWL_Record rdf:about="http://slor.sourceforge.net/ontology/slor.
        owl#LOR_DNAFromTheBeginning">
13      <lom:identifier rdf:resource="http://slor.sourceforge.net/
            ontology/slor.owl#LOR_DNAFromTheBeginning-ID"/>
14       <lom:description rdf:resource="http://slor.sourceforge.net/
             ontology/slor.owl#lsd_LOR_DNAFromTheBeginning_1201629133586"/>
15      <lom:metaMetadata rdf:resource="http://slor.sourceforge.net/
            ontology/slor.owl#LOR_DNAFromTheBeginning-MetaMetaData_Record"/>
16      <lom:contributeLifeCycle rdf:resource="http://slor.sourceforge.
            net/ontology/slor.owl#
            contribute_DNAFromTheBeginning_1201629701405"/>
17      <lom:rightsDescription rdf:resource="http://slor.sourceforge.net/
            ontology/slor.owl#lsrights_LOR_DNAFromTheBeginning_1201627501943
            "/>
18      <lom:language rdf:resource="http://slor.sourceforge.net/ontology/
            slor.owl#English"/>
19      <lom:copyrightAndOtherRestrictions rdf:resource="http://slor.
            sourceforge.net/ontology/slor.owl#LOMv1.0-Yes"/>
20  </LOMCOREOWL_Record>
21
22  <lom:Identifier rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
        LOR_DNAFromTheBeginning-ID">
23      <lom:catalog rdf:datatype="http://www.w3.org/2001/XMLSchema#
            string"
24          >UAHDNAUniversity</lom:catalog>
25      <lom:entry rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
26          >MetadataRecord-LOR_DNAFromTheBeginning</lom:entry>
27  </lom:Identifier>
28
29  <lom:MetaMetaData_Record rdf:about="http://slor.sourceforge.net/ontology
        /slor.owl#LOR_DNAFromTheBeginning-MetaMetaData_Record">
```

```
30          <lom:contributeMetaData rdf:resource="http://slor.sourceforge.
               net/ontology/slor.owl#contribute_DNAFromTheBeginning-
               MetaMetaData_Record"/>
31          <lom:metaSchema rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#LOMv1.0"/>
32          <lom:identifier rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#LOR_DNAFromTheBeginning-MetaMetaData_Record-ID
               "/>
33  </lom:MetaMetaData_Record>
34
35  <lom:Identifier rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
       LOR_DNAFromTheBeginning-MetaMetaData_Record-ID">
36          <lom:catalog rdf:datatype="http://www.w3.org/2001/XMLSchema#
               string"
37              >UAHDNAUniversity</lom:catalog>
38          <lom:entry rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
               >ID-DNA012003101</lom:entry>
39  </lom:Identifier>
40
41  <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
       lsd_LOR_DNAFromTheBeginning_1201629133586">
42          <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
               en</language>
43          <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
44              >DNA from the Beginning is an animated tutorial on DNA,
                   genes and heredity. The science behind each concept is
                   explained using animations, an image gallery, video
                   interviews, problems, biographies, and links. There are
                   three sections, Classical Genetics, Molecules of
                   Genetics and Organization of Genetic Material. Key
                   features are the clear explanations of classical
                   experiments and the excellent photographs of researchers
                    and their labs.</value>
45  </lom:LangString>
46
47  <lom:Contribute rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
       contribute_DNAFromTheBeginning_1201629701405">
48          <lom:contributeRole rdf:resource="http://slor.sourceforge.net/
               ontology/slor.owl#LOMv1.0_ContentProvider"/>
```

```
49          <lom:contributeDate rdf:resource="http://slor.sourceforge.net/
                ontology/slor.owl#d_DNAFromTheBeginning_1201629701405"/>
50          <lom:entity rdf:resource="http://www.cshl.edu#cshl"/>
51  </lom:Contribute>
52
53  <lom:LangString rdf:about="http://slor.sourceforge.net/ontology/slor.owl#
        lsrights_LOR_DNAFromTheBeginning_1201627501943">
54          <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                en</language>
55          <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
56              >Unless stated otherwise, all material appearing on this
                    web site is: Copyright 2002, Cold Spring Harbor
                    Laboratory. The images and content in DNA from the
                    Beginning are for educational use only. Material may be
                    used in reports, research, and other noncommercial
                    projects provided that proper attribution with the
                    copyright notice accompany the material. An acceptable
                    copyright notice is as follows: Copyright, Cold Spring
                    Harbor Laboratory. All rights reserved. Material in DNA
                    from the Beginning may not be used in any form by
                    organizations or commercial concerns, except with
                    express permission. All material appearing on this web
                    site and not generated by the Dolan DNA Learning Center
                    is accompanied by its own disclaimer and copyright
                    notice. Permission for reuse of such material must be
                    obtained from the source.
57          </value>
58  </lom:LangString>
```

Apéndice C

[III] Ejemplo - Jerarquía OpenCyc

```
*TransportationDevice-Vehicle
    *AirTransportationVehicle [UniversalVocabularyMt]
        *Airplane
        [ProductGMt -> (*ManufacturedGoods)]
        [UniversalVocabularyMt -> (*AirTransportationVehicle, *
            FixedWingAircraft, *FuelPoweredDevice)]
          *(ArtifactTypeForUserTypeFn Airplane (
              SubcollectionOfWithRelationFromTypeFn Person passengers
              TransportationEvent)) [BaseKB, ProductGMt] <-
          *(SubcollectionOfUsedForEventTypeFn Airplane
              Surveillance-InformationGatheringActivity) [BaseKB]
          *(SubcollectionOfWithRelationFromFn (
              SubcollectionOfUsedForEventTypeFn Airplane
              Surveillance-InformationGatheringActivity)
              possessiveRelation Israel) [BaseKB] <-
          *(SubcollectionOfWithRelationFromFn Airplane owns
              UnitedNationsOrganization) [BaseKB] <-
          *(SubcollectionOfWithRelationFromFn Airplane possesses (
              GovernmentFn Israel)) [BaseKB] <-
          *(SubcollectionOfWithRelationFromFn Airplane
              possessiveRelation Israel) [BaseKB] <-
              *Airbus [CycNounLearnerMt] <-
              *Airliner [UniversalVocabularyMt] <-
              *Biplane [UniversalVocabularyMt] <-
              *CargoAirplane [UniversalVocabularyMt]
                  *TankerAircraft [UniversalVocabularyMt] <-
              *CessnaAirplane [UniversalVocabularyMt] <-
```

```
18              *FloatPlane [UniversalVocabularyMt] <-
19              *JetPropelledAircraft [UniversalVocabularyMt]
20                  *B-1-Bomber [UniversalVocabularyMt] <-
21                  *B-52-Bomber [UniversalVocabularyMt] <-
22                  *EurofighterTyphoon-FighterPlane [
                        UniversalVocabularyMt] <-
23                  *F-18-Fighter [UniversalVocabularyMt] <-
24                  *F-4D-Fighter [UniversalVocabularyMt] <-
25                  *F-4E-Fighter [UniversalVocabularyMt] <-
26                  *F-5E-Fighter [UniversalVocabularyMt] <-
27                  *FighterPlane-A10 [UniversalVocabularyMt] <-
28                  *FighterPlane-F14 [UniversalVocabularyMt] <-
29                  *FighterPlane-F15 [UniversalVocabularyMt] <-
30                  *FighterPlane-F16 [UniversalVocabularyMt] ...
31                  *MiG-23-Fighter [UniversalVocabularyMt] <-
32                  *MiG-29-Fighter [CSIS-CreepingProliferationMt] <-
33                  *Mirage-F1-Fighter [UniversalVocabularyMt] <-
34                  *Mirage2000EMFighterPlane [UniversalVocabularyMt] <-
35                  *Mirage5E2FighterPlane [UniversalVocabularyMt] <-
36                  *Su-20-Fighter [CSIS-CreepingProliferationMt] <-
37                  *SuperEtendard-Fighter [Iran-IraqTankerWarMt] <-
38                  *TurbojetPropelledAircraft [UniversalVocabularyMt]
                        ...
39                  *WildWeasel [UniversalVocabularyMt] <-
40              *KE-3TankerPlane [UniversalVocabularyMt] <-
41              *MilitaryAirplane [UniversalVocabularyMt]
42                  *Warplane [UniversalVocabularyMt] ...
43              *Monoplane [UniversalVocabularyMt]
44                  *JetPropelledAircraft [UniversalVocabularyMt] ...
45                  *P51Mustang [UniversalVocabularyMt] <-
46                  *Ultralight-Airplane [UniversalVocabularyMt] <-
47              *PassengerAirplane [UniversalVocabularyMt]
48                  *Boeing707 [UniversalVocabularyMt] <-
49                  *Boeing727 [UniversalVocabularyMt] <-
50                  *Boeing737 [UniversalVocabularyMt] <-
51                  *Boeing747 [UniversalVocabularyMt] <-
52                  *Boeing757 [UniversalVocabularyMt] <-
53                  *Boeing767 [UniversalVocabularyMt] <-
54                  *CommercialPassengerAirplane [UniversalVocabularyMt]
                        <-
```

```
55                    *Concorde [UniversalVocabularyMt] <-
56                 *PrivatePlane [UniversalVocabularyMt] <-
57                 *PiperAirplane [UniversalVocabularyMt] <-
58                 *PropellerDrivenAirplane [UniversalVocabularyMt]
59                    *B-29-Bomber [UniversalVocabularyMt] <-
60                    *P51Mustang [UniversalVocabularyMt] <-
61                    *Ultralight-Airplane [UniversalVocabularyMt] <-
62                 *ReconnaissancePlane [UniversalVocabularyMt]
63                    *E-3-AWACS [UniversalVocabularyMt] <-
64                    *TornadoIDS [UniversalVocabularyMt] <-
65                 *SpyPlane [UniversalVocabularyMt] <-
66                 *Warplane [UniversalVocabularyMt]
67                    *(SubcollectionOfWithRelationFromFn Warplane
                         possesses (GovernmentFn Israel)) [BaseKB] <-
68                    *(SubcollectionOfWithRelationFromFn Warplane
                         possessiveRelation Israel) [BaseKB] <-
69                       *BomberPlane [UniversalVocabularyMt] ...
70                       *FighterPlane [UniversalVocabularyMt] ...
```

Apéndice D

[IV]Tratamiento de OWL con JENA

D.1. Cargar un modelo a partir de un fichero OWL

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.iterator.*;
import com.hp.hpl.jena.ontology.*;

OntModel m= ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM,null
    );
m.read( "file:/pro1/onto.owl" );
```

D.2. Establecer un modelo persistente

```
try {
    Class.forName("com.mysql.jdbc.Driver"); // carga del driver
}catch (ClassNotFoundException ex) {//....}

    String DB_URL = "jdbc:mysql://localhost/test"; // URL BASE DE
       DATOS
    String DB_USER = "root";                    // USUARIO DB
    String DB_PASSWD = "";                      // PWD BD
    String DB = "MySQL";                        // TIPO BD

IDBConnection conn= new DBConnection(DB_URL, DB_USER,DB_PASSWD, DB);
```

```
Model model;
try {
    ModelMaker c = ModelFactory.createModelRDBMaker(conn);
    if(!conn.containsModel("ontologiaX"))
    {
    // Creación inicial si no existe el modelo en la base de datos.
        c=ModelFactory.createModelRDBMaker(conn);
        model= c.createModel("ontologiaX");
        //Carga del Modelo - OWL FILE
        model.read("http://localhost:8080/PrototypeSLOR/onto-flexible.
           owl");
    }else {
        c.openModel("ontologiaX");
    }
    model = ModelRDB.open(conn, "slorModel");
    m = ModelFactory.createOntologyModel(OntModelSpec.
       OWL_DL_MEM_RDFS_INF, model);
} catch (RDFRDBException ex1) {//....}
```

En un sistema gestor de base de datos, el modelo de la ontología se guarda en varias tablas, mostradas en la figura D.1

D.3. Modificar un modelo

D.3.1. CLASES

```
String ontoNS = "http://www.xfront.com/owl/ontologies/animales/#";
//Crear una clase
OntClass humano = m.createClass( ontoNS + "Humano");

//Recuperar una clase
humano = m.getOntClass( ontoNS + "Humano" );

//Recuperar todas las clases derivadas de una dada.
    OntClass c=m.getOntClass(ontoNS + "Humano");
    if (c!=null){
         for (ExtendedIterator i = c.listSubClasses(false); i.hasNext
            (); ) {
            OntClass subc = (OntClass) i.next();
```

```
13              System.out.println("CLASS:" + subc.getLocalName());
14
15          }
16      }
```

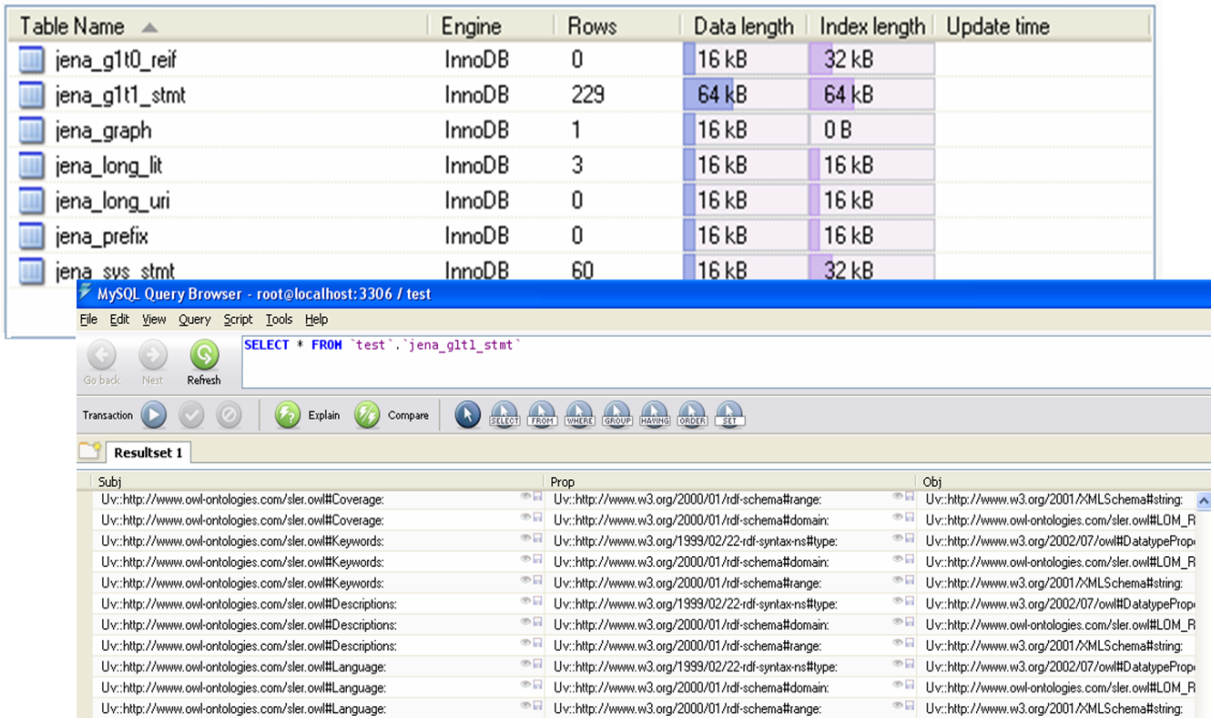

Figura D.1: Modelo de Ontología serializado

D.3.2. PROPIEDADES

```
1  DatatypeProperty pnombre = m.createDatatypeProperty(ontoNS+"Nombre" );
2  ObjectProperty pusadoEn = m.createObjectProperty(ontoNS+"usadoEn" );
3  pnombre = m.getOntProperty(ontoNS+"Nombre" ).asDatatypeProperty();
4  pusadoEn = m.getOntProperty(ontoNS+"usadoEn" ).asObjectProperty();
```

D.3.3. INDIVIDUALES

```
humano = m.getOntClass( ontoNS + "Humano" );
Individual inst = m.createIndividual( ontoNS + "pepe", humano );

inst.addProperty(Schema.Nombre,"HOLA");
inst.addProperty(Schema.Descriptions,"(en) D1");
//...
```

D.4. CONSULTAR EL MODELO

D.4.1. Obtener los individuales de una clase dada

```
      OntClass c = m.getOntClass(ontoNS+ \Humano");
       for (ExtendedIterator i = c.listInstances(); i.hasNext(); ) {
           Individual ind = (Individual) i.next();
           if (ind != null) {
               System.out.println("INDIV:" + ind.getLocalName());
               listIndividuals.add(ind);
           }
       }
```

D.4.2. Acceso al modelo – Motor RDQL

```
//...
import com.hp.hpl.jena.reasoner.*;
import com.hp.hpl.jena.rdql.QueryExecution;
import com.hp.hpl.jena.rdql.*;

             // OntModel m=...

               Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
               reasoner = reasoner.bindSchema(m);
               infmodel = ModelFactory.createInfModel(reasoner, m);

\end{lstlisting}

\subsection{Consulta RDQL}
```

```
\begin{lstlisting}[frame=tlrb]{}
String queryString =
    "SELECT * " +
      "WHERE (?x, <rdfs:subClassOf>, ?y), (?x, <rdf:type>, ?z) " +
       "AND ?y EQ <ex:LearningObject-Generic> && ?z EQ <owl:Class> "
          +
       "USING " +
        "      ex   FOR <" + Schema.NS + "> ," +
        "      rdf  FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
           ," +
        "      rdfs FOR <http://www.w3.org/2000/01/rdf-schema#> , " +
        "      owl  FOR  <http://www.w3.org/2002/07/owl#>";
```

D.4.3. Obtener los resultados

```
try {
          Query query = new Query(queryString);
          query.setSource(infmodel);
          QueryExecution qe = new QueryEngine(query);
          QueryResults results = qe.exec();
          for (Iterator iter = results; iter.hasNext(); ) {
              ResultBinding res = (ResultBinding) iter.next();
              String x = res.get("x").toString();
              String y = res.get("y").toString();
              String z = res.get("z").toString();

              x = x.substring(x.indexOf('#')+1) ;
              y = y.substring(y.indexOf('#') + 1);

              System.out.println(x + " subClassOf " + y + " type " +
                 z);
              listClasses.add(x);

          }
          results.close();
} catch (Exception err) {
          err.printStackTrace();
}
```

Apéndice E

[V]Diagrama de interfaces y clases abstractas de Protege

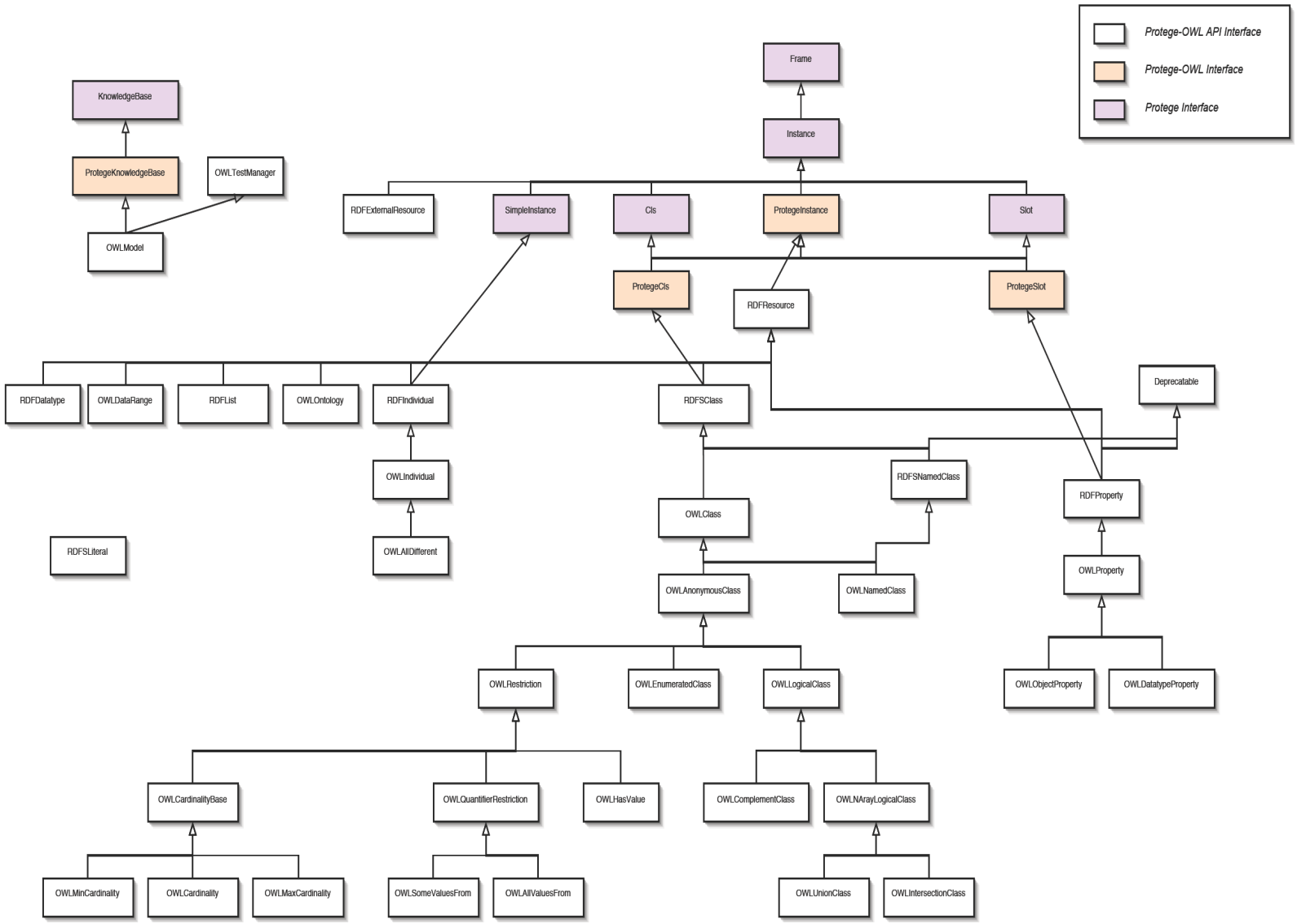

Figura E.1: Diagrama de interfaces y clases abstractas de Protégé

Bibliografía

[ADL03] ADL. *Sharable Courseware Object Reference Model (SCORM) Version 1.3, Application profile working draft 1.0.* Advanced Distributed Learning (ADL), March 2003.

[AFC+02] Luis E. Anido, Manuel J. Fernández, Manuel Caeiro, Juan M. Santos, Judith S. Rodríguez, y Martín Llamas. Educational metadata and brokerage for learning resources. *Computers and Education*, 38(4):351–374, May 2002.

[Alo05] Salvador Sánchez Alonso. *Diseño y uso de objetos didácticos basado en contratos.* PhD thesis, Universidad de Politécnica de Madrid, 2005.

[Arc04] Valerie Archambeau. *Draft Standard for Learning Technology — Reusable Competency Definitions.* IEEE Learning Technology Standards Committee, May 2004.

[Bar96] Douglas K. Barry. *The Object Database Handbook: How to Select, Implement, and Use Object-Oriented Databases.* John Wiley and Sons, 1th edition, 1996.

[BC01] Tom Boyle y John Cook. Towards a pedagogically sound basis for learning object portability and re-use. In G. Kennedy, M. Keppell, C. McNaught, y T. Petrovic, editors, *Meeting at the Crossroads. Proceedings of the 18th Annual Conference of the Australasian Society for Computers in Learning*

in Tertiary Education, pages 101–109, Melbourne: Biomedical Multimedia Unit, 2001. University of Melbourne.

[BCM+02] F. Baader, D. Calvanese, D.L. McGuinness, D.Ñardi, y P.F. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[BH07] Don Box y Anders Hejlsberg. Linq: .net language-integrated query. Technical report, Microsoft, 2007.

[BKPP07] Harold Boley, Michael Kifer, Paula-Lavinia Pătrânjan, y Axel Polleres. Rule Interchange on the Web. In *Proceedings of Summer School Reasoning Web 2007, Dresden, Germany (3rd–7th September 2007)*, volume 4634 of *LNCS*. REWERSE, 2007.

[BKvH01] J. Broekstra, A. Kampman, y F. van Harmelen. Sesame: An architecture for storing and querying rdf data and schema information, 2001.

[BKvH02] J. Broekstra, A. Kampman, y F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema, 2002.

[BL98] Tim Berners-Lee. *Notation 3*. RDF Interest Group and Semantic Web Interest Groups, 1998.

[BLFIM98] Tim Berners-Lee, R. Fielding, U. C. Irvine, y L. Masinter. Uniform Resource Identifiers (URI): Generic syntax. RFC 2396. ftp://ftp.isi.edu/in-notes/rfc2396.txt, August 1998.

[BLHL01] Tim Berners-Lee, James Hendler, y Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[BP98] S. Brin y L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[Caf02] Ralph Cafolla. Project Merlot: Bringing peer review to web-based educational resources. In *Proceedings of the USA Society for Information Technology and Teacher Education International Conference*, volume 2002, pages 614–618. AACE, 2002.

[Cam04] Lorna M. Campbell. *UK Learning Object Metadata Core, Draft 0.2*. Centre for Educational Technology Interoperability Standards (CETIS), May 2004.

[CGRU96] C. Chekuri, M. Goldwasser, Prabhakar Raghavan, y E. Upfal. Web search using automatic classification. In *Proceedings of WWW-96, 6th International Conference on the World Wide Web*, San Jose, US, 1996.

[CO02] Adam Cooper y Claude Ostyn. *IMS Reusable Definition of Competency or Educational Objective, version 1.0*. IMS Global Learning Consortium, Inc., October 2002.

[Cor05] R. Corazzon. Ontology. a resource guide for philosophers, July 2005.

[DCM00] DCMI. *Dublin Core Qualifiers*. Dublin Core Metadata Initiative (DMCI), June 2000.

[DCM03] DCMI. *Dublin Core Metadata Element Set, version 1.1*. Dublin Core Metadata Initiative (DMCI), November 2003.

[DHSW02] Erik Duval, Wayne Hodgins, Stuart Sutton, y Stuart L. Wiebel. Metadata principals and practicalities. *D-Lib Magazine*, 8(4), April 2002.

[Dow04] Steven Downes. Resource profiles. *Journal of Interactive Media in Education*, 2004(5), May 2004.

[DSJS07] Zapico D., Oscar Sanjuán, Luis Joyanes, y Jesús Soto. Secure mobile sessions in web services with serialized-encrypted objects: Regaining client

confidence. In Hamid R. Arabnia, editor, *PDPTA*, pages 710–718. CSREA Press, 2007.

[Duv04] Erik Duval. Learning technology standardization: making sense of it all. *International Journal on Computer Science and Information Systems*, 1(1):33–43, february 2004.

[EdN02] EdNA. *EdNA Metadata Standard, version 1.1*. Educational Network Australia (EdNA) Online, September 2002.

[Far01] Frank Farance. *Draft Standard for Learning Technology — Public and Private Information (PAPI) for Learners*. IEEE Learning Technology Standards Committee, November 2001.

[FFR02] S. Fisher, N. Friesen, y A. Roberts. *CanCore Element Set, version 1.1*. Athabasca University, January 2002.

[FHH04] Richard Fikes, Patrick Hayes, y Ian Horrocks. OWL-QL – a language for deductive query answering on the semantic web. Technical report, KSL, 2004.

[FKP96] N. Freed, J. Klensin, y J. Postel. *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. IANA, 1996.

[Fri01] Norm Friesen. What are educational objects? *Interactive Learning Environments*, 9(3), December 2001.

[FS87] Charles L Forgy y Susan J Shepard. Rete: a fast match algorithm. *AI Expert*, 2(1):34–40, 1987.

[FT01] Frank Farance y Joshua Tonkel. *Draft Standard for Learning Technology — Learning Technology Systems Architecture (LTSA)*. IEEE Learning Technology Standards Committee, November 2001.

[Gar05] Jesse J. Garrett. Ajax: a new approach to web application, 2005. http://www.adaptivepath.com/publications/essays/archives/000385.php.

[GG95] Nicola Guarino y Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, 1995.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, y John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[Gre03] Jane Greenberg. *Encyclopedia of Library and Information Science*, chapter Metadata and the World-Wide-Web, pages 1876–1888. Marcel Dekker Inc., New York, second edition, 2003.

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 6(2):199–221, 1993.

[GS05a] Elena Garcia y Jesús Soto. A brief reflection on the ontological definition of learning objects. pages 25–28, 2005.

[GS05b] Elisa Garcia y Jesús Soto. Autenticidad basada en la web semántica de confianza. In *Actas de CIASI 2005 - IV Congreso Internacional de Auditoria y Seguridad de la Información*, volume 1, pages 147–157. CIASI, 2005.

[HKS06] Ian Horrocks, Oliver Kutz, y Ulrike Sattler. The even more irresistible sroiq. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 57–67. 10th International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, June 2006.

[HMW] Volker Haarslev, Ralf Möller, y Michael Wessel. Querying the semantic web with racer + nrql.

[HRJ02] Cheryl J. Hamel y David Ryan-Jones. Designing instruction with learning objects. *International Journal of Educational Technology*, 3(1), November 2002.

[Hyd01] Jack Hyde. *CMI Guidelines for Interoperability.* AICC, April 2001.

[IMS03a] IMS Global Learning Consortium, Inc. *IMS IMS Digital Repositories Interoperability - Core Functions Best Practice Guide*, Januray 2003.

[IMS03b] IMS Global Learning Consortium, Inc. *IMS IMS Digital Repositories Interoperability - Core Functions Information Mode, version 1.0*, Januray 2003.

[IPT05] IPTC Standars. *ÍPTC Core'Schema for XMP*, March 2005.

[Jac04] Alex Jackl. *IMS Resource List Interoperability Information Model, version 1.0.* IMS Global Learning Consortium, Inc., May 2004.

[JBR99] Ivar Jacobson, Grady Booch, y James Rumbaugh. *The Unified Software Development Process.* Addison Wesley, 1999.

[Kee06] Ruth Keeling. The bologna process and the lisbon research agenda: the european commissions expanding role in higher education discourse. *European Journal of Education*, 41(2):203–223, June 2006.

[KGdRB91] Kiczales, Gregor, Jim des Rivieres, y Daniel G. Bobrow. *The Art of the Metaobject Protocol.* MIT Press., 1991.

[KOA03a] Rob Koper, Bill Olivier, y Thor Anderson. *IMS Learning design best practice and implementation guide, version 1.0.* IMS Global Learning Consortium, Inc., January 2003.

[KOA03b] Rob Koper, Bill Olivier, y Thor Anderson. *IMS Learning Design Information Model, version 1.0*. IMS Global Learning Consortium, Inc., January 2003.

[Kop01] Rob Koper. Modeling units of study from a pedagogical perspective: the pedagogical meta-model behind EML. Technical report, Open University of the Nederlands, June 2001.

[Lay04] Steve Lay. *IMS Question and Test Interoperability: Item Information Model, version 2.0*. IMS Global Learning Consortium, Inc., June 2004.

[Len91] Douglas B. Lenat. Cyc: Priming the knowledge sharing pump. In *Proc. of the International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, pages 151–152, Tokyo, Japan, 1991.

[Len95] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[Lew04] Scott Lewis. *Draft Standard for Learning Technology — Data Model for Content Object Communication*. IEEE Learning Technology Standards Committee, July 2004.

[Lon00] Warren Longmire. A primer on learning objects. *ASTD Learning Circuits*, March, March 2000.

[LOR05] LORI. *Learning Object Repositories Interoperability Framework (LORI) Version 1.0 Beta*. CEN/ISSS and PROLEARN and Other Contributors, February 2005.

[LTS02] LTSC. *IEEE Standard for Learning Object Metadata, 1484.12.1-2002*. IEEE Learning Technology Standards Committee, 2002.

[LVNW03] Carl Lagoze, Herbert Van de Sompel, Michael Nelson, y Simeon Warner. *The Open Archives Initiative Protocol for Metadata Harvesting Protocol, version 2.0*. The Open Archives Initiative (OAI), February 2003.

[Lyn97] Clifford A. Lynch. The Z39.50 information retrieval standard. Part I: A strategic view of its past, present and future. *D-Lib Magazine*, April, April 1997.

[M.05] Weiss M. *Text Mining: Predictive Methods For Analyzing Unstructured Information*. Springer, 2005.

[Mag] Roger Magoulas. Programming language trends. OReally.

[Mar02] Margaret Martinez. Designing learning objects to personalize learning. In David A. Wiley, editor, *The instructional use of learning objects*, pages 151–171. Agency for Instructional Technology and Association for Educational Communications and Technology, Bloomington, Indiana, 2002.

[McG04] Rory McGreal. Learning objects: A practical definition. *International Journal of Instructional Technology and Distance Learning*, 1(9):21–32, September 2004.

[Mci07] I.C. Mcilwaine. *The Universal Decimal Classification: a guide to its use*. The Hague : UDC Consortium, 1th edition, 2007.

[Mic06] Microsoft. *COM + technology*. Microsoft, Januray 2006.

[Min86] Marvin Minsky. *The society of mind*. Simon and Schuster, 1986.

[Min88] Marvin Minsky. Why people think computers can't. pages 224–236, 1988.

[Min06] Marvin Minsky. *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. Simon & Schuster, November 2006.

[Mor02] Steve Morrison. Get on with IT. Technical report, Post-16 E-learning Strategy Task Force, UK Department for Education and Skills, July 2002.

[MR01] Rory McGreal y Toni Roberts. A primer on metadata for learning objects: fostering an interoperable environment. *e-Learning Magazine*, 2(10):26–29, October 2001.

[MSE+08] Nikos Manouselis, Jesús Soto, Hannes Ebner, Matthias Palmér, y Ambjorn Naeve. A semantic infrastructure to support a federation of agricultural learning repositories. In *ICALT Learning technologies in the Information society*. ICALT, 2008.

[MW03] Jon Mason y Nigel Ward. The le@rning federation metadata application profile. *IEEE Learning Technology Newsletter*, 5(1):7–8, January 2003.

[MYO01] Sosteric M., Shi Y., y Wenker O. The upcoming revolution in the scholarly communication system. *The Journal of Electronic Publishing*, 7(13):259—-272, 2001.

[ND02] Filip Neven y Erik Duval. Reusable learning objects: a survey of LOM-based repositories. In *Proceedings of the Tenth ACM International Conference on Multimedia*, pages 291–294. ACM, 2002.

[Net06] SDN Sun Developer Network. *EJB Specification, version 3.0*. Sun MicroSystems, May 2006.

[NM01] N. Fridman Noy y D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems LaboratoryTechnical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[nSGSAS05] Miguel Ángel Sicilia, Elena García, Salvador Sánchez-Alonso, y Jesús Soto. A semantic lifecycle approach to learning object repositories. In *Proceedings of ELETE 2005 - eLearning on Telecommunications*, 2005.

[OAS02] OASIS. *UDDI Version 2.04 API Specification*, July 2002.

[OMG01] OMG. *CORBA Specification, versión 3.0*. Object Management Group, March 2001.

[Pae93] A. Paepcke. *Object-Oriented Programming: the CLOS Perspective*. MIT Press., 1993.

[Pol03] Pithamber R. Polsani. Use and abuse of reusable learning objects. *Journal of Digital Information*, 3(4), February 2003. Article No. 164.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, y Ian Horrocks. OWL Web Ontology Language - Semantics and Abstract Syntax. W3c recommentation, W3C, February 2004.

[RBY99] B. RibeiroÑeto R. Baeza Yates. *Modern Information Retrieval*. New York: Adison-Wesley, 1999.

[RM03] Daniel R. Rehak y Robin Mason. *Reusing online resources: a sustainable approach to e-learning*, chapter 3: Keeping the learning in learning objects, pages 20–35. RoutledgeFalmer, London, U.K., first edition, 2003.

[Rob02] Robby Robson. Metadata, schmetadata: Why do I have to know about this? *e-Learning Magazine*, 3(5):48–50, May 2002.

[RTJ05] Dirk Riehle, Michel Tilman, y Ralph Johnson. *Pattern Languages of Program Design*. Addison-Wesley, 2005.

[RvRK+02] Adrian Rawlings, Peter van Rosmalen, Rob Koper, Miguel Rodríguez-Artacho, y Paul Lefrere. Survey of educational modelling languages

(EMLs), version 1. Technical report, CEN/ISSS Learning Technologies Workshop, September 2002.

[SAS05] Salvador Sánchez-Alonso y Miguel Angel Sicilia. Normative specifications of learning objects and learning processes: Towards higher levels of automation in standardized e-learning. *International Journal of Instructional Technology and Distance Learning*, 2(3):3–11, March 2005.

[SASnS06] Salvador Sánchez-Alonso, Jesús Soto, y Miguel Ángel Sicilia. *Learning Objects: Standards, Metadata, Repositories, and LCMS (Paperback)*, chapter Designing Flexible Learning Object Repositories: Balancing Flexibility and Delegation in Ontological Characterizations, pages 221–255. Informing Science Institute, California, USA, first edition, 2006.

[SeR04] The SeRQL query language (revision 3.0). *Re-SQL*, Januray 2004.

[SG03] Miguel-Angel Sicilia y Elena García. On the concepts of usability and reusability of learning objects. *International Review of Research in Open and Distance Learning*, 4(2), October 2003.

[SG05] Jesús Soto y Elisa Garcia. Sistema multiagente inteligente para la planificación organizada del estudio de un alumno. *III Simposio Internacional de Sistemas de Información e Ingeniería del Software en la Sociedad del Conocimiento*, 1(1):51–34, Ago 2005.

[SGM06] Jesús Soto, Gloria Garcia, y Víctor Martín. Mejora de la calidad y servicio en los repositorios de objetos de aprendizaje multimedia. In *Actas del III Simposio Plurisciplinar sobre objetos y diseños de aprendizaje, Oviedo, Spain.* SPDECE, 2006.

[SGRJ06a] Jesús Soto, Gloria Garcia, Ismael Rodriguez, y Luis Joyanes. A new approach to dynamic load balancing across multimedia servers. *WSEAS Transactions on Computers*, 5(11):2758–2764, 2006.

[SGRJ06b] Jesús Soto, Gloria Garcia, Ismael Rodriguez, y Luis Joyanes. Open load multimedia server balancing. 2006.

[SGSA06] Jesús Soto, Elisa Garcia, y Salvador Sánchez-Alonso. - problemas de almacenamiento e inferencia sobre grandes conjuntos de metadatos en un repositorio semántico de objetos de aprendizaje. In *Actas del III Simposio Plurisciplinar sobre objetos y diseños de aprendizaje, Oviedo, Spain.* SPDECE, 2006.

[SGSA07] Jesús Soto, Elisa Garcia, y Salvador Sánchez-Alónso. Semantic learning object repositories. 17(6):432–446, 2007.

[SGSJ06] Jesús Soto, Elisa Garcia, Oscar SanJuan, y Luis Joyanes. An analysis about the rdf storage problems: Towards semantic web servers. 2006.

[Sin00] Harvi Singh. Achieving interoperability in e-learning. *ASTD Learning Circuits*, March, March 2000.

[SJJ06] Jesús Soto, Oscar San Juan, y Luís Joyanes. Semantic web servers: A new approach to query on big datasets of metadata. *WSEAS Transactions on Computers*, 5(11):2658–2662, 2006.

[SL02] R. SL. Mapping ontologies into cyc, 2002.

[Sla01] Jenny Slater. *FAILTE guidelines, version 1.2*. Facilitating Access to Information on Learning Technology for Engineers (FAILTE), December 2001.

[Smy03] Colin Smythe. *IMS Content packaging information model, version 1.1.3 final specification*. IMS Global Learning Consortium, Inc., June 2003.

[SnSSA05] Jesús Soto, Miguel Ángel Sicilia, y Salvador Sánchez-Alonso. Resource descriptions for semantic element repositories: addressing flexibility. In *Proceedings of m-ICTE 2005 - Third International Conference on Multimedia and Information and Communication Technologies in Education*, 2005.

[SQI05] SQI. *Simple Query Interface (SQI) Version 1.0 Beta*. CEN/ISSS and PROLEARN and Other Contributors, April 2005.

[SRT05] A. Seth, C. Ramakrishnan, y C. Thomas. Semantics for the semantic web:the implicit, the formal and the powerful. *International Journal on Semantic Web and Information Systems*, 1(1):1–18, 2005.

[SS02] Steven Shaw y Sarah Sniderman. Reusable learning objects: Critique and future directions. In *Proceedings of ELearn 2002 - World Conference on E-Learning in Corporate, Government, Healthcare and Higher Education*, volume 2002, pages 2154–2157. AACE, 2002.

[SS05] Miguel-Angel Sicilia y Salvador Sánchez. Uso de objetos y diseños para el aprendizaje. REDAOPA, 2005.

[SSAnS05] Jesús Soto, Salvador Sánchez-Alonso, y Miguel Ángel Sicilia. Semantic learning object repositories: flexibility and implementation issues. In *Proceedings of m-ICTE 2005 - Third International Conference on Multimedia and Information and Communication Technologies in Education*, 2005.

[SSJ04] Jesús Soto, Oscar SanJuan, y Luis Joyanes. Mejores prácticas .net. In *Actas de la I Jornada de Investigación en Ingeniería Informática, Salamanca-Lisboa*. JIII, 2004.

[Sta05] Standard of Japan Electronics and Information Technology Industries Association. *Exchangeable image file format for digital still cameras*, March 2005.

[Sta07] Stanford Center for Biomedical Informatics Research. *Protégé: JDBC back-end design rationale*, 2007.

[STR01] Colin Smythe, Frank Tansey, y Robby Robson. *IMS Learner Information Package Information Model Specification, version 1.0*. IMS Global Learning Consortium, Inc., March 2001.

[Swi] Terrance Swift. Efficiently implementing slg resolution:.

[TLF03] TLF. *Metadata application profile, version 1.3*. The Le@rning Federation initiative, June 2003.

[VM05] J.P. Vidal y Meire. Bridging the gap between learning management systems and learning object repositories: exploiting learning context information. In IEEE, editor, *eLearning on Telecommunications Workshop*, page 478–483. ELETE, 2005.

[W.03] Berry W. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer, 2003.

[W3C99] W3C. *XML Path Language Specification, version 1.0*, November 1999.

[W3C01] W3C. *Web Services Description Language*, March 2001.

[W3C03] W3C. *SOAP Specifications*, June 2003.

[W3C04a] W3C. *SWRL A Semantic Web Rule LanguageCombining OWL and RuleML*, May 2004.

[W3C04b] W3C World Wide Web Consortium. *RDF/XML Syntax Specification*, February 2004.

[W3C07] W3C World Wide Web Consortium. *SPARQL Query Language for RDF*, June 2007.

[WG01] C. Welty y N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, 39(1):51–34, May 2001.

[Wil99] David A. Wiley. The post-lego learning object, November 1999.

[Wil02] David A. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor and a taxonomy. In David A. Wiley, editor, *The instructional use of learning objects*, pages 3–24. Agency for Instructional Technology and Association for Educational Communications and Technology, Bloomington, Indiana, 2002.

[Wil06] K. Wilkinson. Jena property table design, 2006.

[Win95] Dave Winer. *XML-RPC Specification.* UserLand Software, June 1995.

[WJ94] M. Wooldridge y N. R. Jennings. Agent theories, architectures, and languages: A survey. In M. J. Wooldridge y N. R. Jennings, editors, *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, pages 1–39. Springer, Berlin,, 1994.

[WJS05] León Weicki, Oscar San Juan, y Jesús Soto. Modificación de la intención semántica de relatos utilizando ontologías y técnicas evolutivas. In *Actas de CEDI-MAEB 2005 - Congreso Nacional de Metaheurísticas y algoritmos evolutivos.* CEDI, 2005.

[WMB+05] M. Witbrock, C. Matuszek, A. Brusseau, R.C. Kahlert, C.B. Fraser, y D. B. Lenat. Knowledge begets knowledge: Steps towards assisted knowledge acquisition in cyc. In *Papers of AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors (KCVC)*, pages 99–105, Stanford, USA, 2005.

[WSKR03] K. Wilkinson, C. Sayers, H. Kuno, y D. Reynolds. Efficient rdf storage and retrieval in jena, 2003.