Generación de Estructuras de Conocimiento Conceptual mediante el análisis de relaciones taxonómicas de Ontologías de Dominio

Víctor Sanhueza Salazar , Jaime Luna Orrego

Departamento de Sistemas de Información

Universidad del Bío-Bío

Tesis presentada para el grado de

Ingeniero Civil en Informática

Marzo 6, 2013

No progresas mejorando lo que ya esta hecho, sino esforzándote por lograr lo que aún queda por hacer.

Gibran Jalil Gibran

Abstract

Hoy en día, el concepto de E-learning, se posiciona como una real alternativa que permite enfrentar barreras físicas de acceso al conocimiento, adaptándose a la flexibilidad temporal y a la interacción que por su naturaleza entrega a los estudiantes. No obstante el diseño de cursos en ambientes E-learning no es una actividad fácil y requiere el uso de teorías que apoyen esta labor. Las teorías de Diseño Instuccional proporcionan una guía que apoya el proceso de diseño de cursos y recursos de aprendizaje. Una de estas teorías, la Teoría de Elaboración, ayuda a la secuenciación de contenidos pero requiere la existencia de una Estructura de Conocimiento Conceptual (EGS) que ordena los contenidos. Este Proyecto de Título propone una estrategia e implementa una herramienta denominada OntoConcept que permite crear una EGS a partir del análisis de las relaciones ontológicas *is_a* y *part_of* de una ontología de dominio expresada en lenguaje OWL. Esta propuesta muestra el análisis, la implementación y la vinculación de la herramienta con otras aplicaciones de E-learning existentes.

Contenidos

CONTENIDOS

Lista de Figuras

LISTA DE FIGURAS

Lista de Tablas

1

Marco de la Investigación

1.1 Introducción

Cada vez es más frecuente la utilización de plataformas E-learning que dan soporte al proceso educativo dentro de un contexto formativo; fundamentadas en ser una opción de aprendizaje que va más allá de todas las barreras geográficas, adaptadas a la flexibilidad temporal, con autonomía e independencia , sin olvidar la posibilidad de personalización del aprendizaje y la accesibilidad de los contenidos a usuarios con discapacidades. El E-learning y el diseño de recursos de aprendizajes presentan una complejidad gravitante debido a que deben ser un apoyo instruccional y no tópicos sin cohesión alguna. El diseño, considera una serie de tópicos: contenidos de aprendizajes, recursos, actividades, servicios, roles, entre otros, usualmente estructurados con miras a producir objetos de aprendizajes reutilizables, bajo el alero del proceso sistemático, planificado y estructurado que se engloba en el concepto de **Diseño Instruccional (DI)** [Reigeluth, 1999](15). El Diseño Instruccional se manifiesta mediante un precepto de lenguaje natural, por lo que el modelado de este, por medio de lenguajes computables, aplacaria la carencia de una sintaxis formal que condiciona el actuar de cada instructor de acuerdo a como este percibe el DI, según lo precisa la literatura.

En virtud de lo anterior, existe la necesidad de modelar el DI apoyado en la implementación de un lenguaje con semántica computacional que de soporte al DI. Como consecuencia de esto, la creación de la especificación **IMS-LD (Learning Design)** por el IMS-Consortium para la descripción de recursos de aprendizajes, especifica un modelado de lenguaje para definir actividades instruccionales, con el objetivo de per-

mitir la reutilización e interoperabilidad. Existen aplicaciones que interpretan y que generan LD, como el caso de **ReCourse** [Beauvoir, 2009](16) y aquella en particular como **Terpsícore** [CVidal, 2011](26) que permite vincular algunas de las **teoría de Diseño Instruccional (TDI)** con un LD cualquiera.

A partir de estos trabajos, se plasman los lineamientos para el desarrollo de la actividad de titulación, que pretende ser un aporte concreto al modelado de las TDI **particularmente sobre la Teoría de Elaboración** [Reigeluth, 1999](15).Pretendiendo darle sustento a la particularidad que presenta esta Teoría: **Un estructura de conocimiento conceptual (ECC)**. A raíz de esto, las principales aportaciones de esta tesis son la confección de una herramienta que permita generar ECC a partir del análisis de las relaciones taxonómicas de ontologías de dominio, permitiendo además, la adhesión de esta a Terpsícore, particularmente en función de dar soporte al modelado de la Teoría de Elaboración que manifiesta por medio de reglas e inferencias y por ende contribuir de manera concreta a dar un sustento pedagógico sólido al diseño de LD.

1.2 Descripción de la problemática

El anhelo continuo de incrementar la efectividad en la educación, la reducción de costos mediante la reutilización de recursos de aprendizajes, la interoperabilidad entre sistemas y la automatización de procesos de aprendizajes, tiene cimiento a partir de la utilización de notaciones formales en el diseño de cursos. A partir de esto, diversas propuestas han surgido para dar respuesta a la formalización la descripción de escenarios de aprendizajes, siendo la especificación IMS-LD como la solución más asentida.

El concepto de learning design, es una guía práctica de diseño de curso que permite plasmar la generación de secuencias instructivas, compuestas de roles, actividades, recursos de aprendizajes, que definen el escenario de aprendizaje, denominado también Uols (Unidades de Aprendizajes). Los LD, se describen por medio de metadatos estandarizados que caracterizan los contenidos en detalles. No obstante, la especificación IMS-LD propiamente tal, no ofrece un modelo pedagógico concreto, por lo que su utilización se define para un sinfín de escenarios educativos y a consecuencia de esto, se generan críticas por su neutralidad pedagógica, en base a la carencia de reglas, métodos o información de cómo se lleva a cabo el diseño. Como consecuencia de los trabajos sobre el modelado de TDI por medio de reglas e inferencias computables [CVidal, 2011](27);

la Teoría de Elaboración manifiesta una estructura de conocimiento conceptual sobre un dominio de aprendizaje, la que no posee un soporte computacional, por lo que el problema de investigación de esta tesis se enmarca dentro de este contexto y la necesidad de representar la (ECC) por medio de una semántica computable. En resumen, se requiere generar una estructura de conocimientos conceptual (ECC) a partir del análisis de relaciones taxonómicas de una ontología de dominio, que podría utilizarse por algún software para dar soporte al Diseño Instruccional a partir de la Teoría de Elaboración.

1.3 Hipótesis de partida

La hipótesis de partida se define de acuerdo a lo siguiente:

Es posible representar, una estructura de conocimiento conceptual que apoye la Teoría de Elaboración mediante un lenguaje con semántica computacional por medio del análisis de relaciones taxonómicas de una ontología de dominio.

Es decir, se suscita lo sustancial de esta tesis en relación a la carencia de una semántica computacional a la generación de una ECC que manifiesta la Teoría de Elaboración a partir de ontologías de dominio, de manera que pueda ser utilizada para dar soporte al proceso de Diseño Instruccional.

1.4 Objetivos

1.4.1 Objetivo General

El objetivo principal de esta tesis es desarrollar una herramienta, utilizando un lenguaje con semántica computacional que analice las relaciones ontológicas de una ontología de dominio, generando una estructura de conocimiento conceptual (ECC) a partir del marco de la Teoría de Elaboración de manera que sea posible dar soporte al Diseño Instruccional.

1.4.2 Objetivos Específicos

A partir de este objetivo principal se han definido los objetivos específicos de este trabajo:

1. Analizar la estructura y relaciones lógicas (*is_a*, *part_of*) de una ontología de dominio OWL.

2. Analizar la Teoría de Elaboración, dentro del contexto de teorías del Diseño Instruccional.

3. Estudiar el lenguaje de ontologías para la web OWL para su manejo sobre el lenguaje Java.

4. Construir una aplicación que genere una ECC a partir de una ontología de dominio utilizando un framework de ontología y RCP de Eclipse.

5. Integrar la herramienta a la aplicación Terpsícore, a través de un entorno gráfico generado por el framework GEF, permitiendo la interacción del usuario, para dar sustento a la Teoría de Elaboración por medio de la vinculación de un LD con una ontología de dominio tratada taxonómicamente como un ECC.

1.5 Metodología de Trabajo

A continuación se describen las principales etapas consideradas en la metodología de investigación consideradas para esta tesis.

- **Definición de la Problemática:** En esta primera fase, se define el origen de la problemática que aborda esta investigación, con énfasis en la necesidad de representar una ECC que de soporte a la Teoría de Elaboración dentro del marco de Diseño Instruccional.

- **Estado del Arte:** En esta fase, se realiza una conceptualización de tópicos respecto a la generación de un ECC. Además analiza de manera vasta, el concepto de ontologías, su estructura, su modelamiento por medios de lenguajes adecuados y las aplicaciones. Finalmente se realiza un análisis de las característica de las principales teorías de Diseño Instruccional, con énfasis en la Teoría de Elaboración.

- **Proposición de un mecanismo de generación de una ECC a partir de una ontología de dominio:** Se divide en dos actividades principales:

  - **Análisis de la Teoría de Elaboración y concepción del modelado de una ECC:** De acuerdo a la Teoría de Elaboración, se propone un mecanismo para generar un ECC, a partir de la estructuras taxonómicas que presentan ontologías de dominio.
  - **Diseño de una estrategia:** Por medios de lenguajes adecuados que den soportes a ontologías formales, se representa la ECC, por medio del tratamiento de descripciones ontológicas.

- **Generación de una herramienta:** En base al mecanismo sobre la ECC que da soporte a la Teoría de Elaboración y a los lenguajes de tratamientos de ontologías, analizados de manera exhaustiva en el estado del arte, se implementa una aplicación que permite dar fundamento a la generación de una ECC por medio del análisis taxonómico de una ontología de dominio.

- **Integración de la aplicación a la herramienta Terpsícore, como parte del contexto de soporte de las TDI:** En base a dar soporte a la TDI, se dispone la integración de la aplicación a la herramienta Terpsícore, fundamentado en representar la Teoría de Elaboración por medio de una semántica computacional, dentro de un entorno de Diseño Instruccional.

1.6 Metodología de Desarrollo de Aplicación

En base al análisis de tecnologías y conceptualización de entornos de aprendizajes y enseñanzas , se establece la metodología de desarrollo incremental, como la más apropiada para llevar a cabo los desafíos; fundamentada en ser una derivación del modelo cascada, que si bien incorpora las etapas de análisis, diseño, implementación y pruebas, lo hace bajo el principio del trabajo en cadena pipeline, que radica en la entrega de incrementos que representan un avance gradual de alguna funcionalidad de la aplicación. Los incrementos proporcionaran de forma progresiva nuevas funcionalidades operacionales a una aplicación. Es importante destacar que cada incremento se considera como un módulo incompleto del sistema, por lo tanto, debe ser evaluado, por quien disponga el rol de cliente, el cual determinará el cumpliendo de los requerimientos, o si deben ser modificados, ambas decisiones implican un nuevo incremento, los cuales finalizarán solo al completar los requisitos contemplados.

En resumen, de acuerdo al alto grado de incertidumbre que manifiesta la naturaleza en que se prevé desarrollar la aplicación, el grado de manejo teórico que manifiesta el entorno de aprendizaje-enseñanza, y a fin de disminuir los riesgos evaluados, la metodología de desarrollo incremental es la más pertinente, para el desarrollo de la aplicación.

1.7 Marco del Trabajo

El origen de esta investigación surge a partir del marco de los trabajos realizados por el Dr. Christian Vidal Castro (Universidad Del Bío-Bío), Dr. Miguel-Angel Sicilia (Universidad de Alcalá) y el Dr. Manuel Prieto (Universidad de Castilla-La Mancha) sobre el contexto del Diseño Instruccional [Castilla-La Mancha, 2011], y cuyo principal aporte involucra un modelado formal de las teorías del Diseño Instruccional (DI) que no existía en la literatura hasta ese entonces, la obtención de un catálogo de teorías del DI (33) y un prototipo de software (Terpsícore)(26) que analiza la conformidad de un Learning Design (LD) con una determinada Teoría del DI, dentro de la aplicación ReCourse que modela Learning Design.

En virtud de lo anterior, la presente investigación se enmarca dentro del trabajo previo, aplicado a dar soporte a la Teoría de Elaboración, dentro del contexto de las teorías de Diseño Instruccional, permitiendo el análisis de relaciones taxonómicas de ontologías de dominios, generando una ECC, propia de la Teoría de Elaboración a partir de un tópico ontológico. Adicionalmente, proporciona a Terpsícore una optimización en base a un prototipo de software que permite dar sustento a la Teoría Elaborativa, por medio de la vinculación de los tópicos considerados en un de LD con tópicos de aprendizaje proporcionados por una ECC.

1.8 Organización del Trabajo

- **Capítulo 1: Conceptualización del problema de investigación:** En este capítulo se presentan los objetivos e hipótesis de investigación, así como el marco de trabajo y método utilizado para el desarrollo de la tesis. Se considera también el contexto de la problemática relacionada con la carencia de un sustento semántico a la Teoría de Elaboración. Finalmente se considera también la metodología para

el desarrollo de la aplicación que de sustento a la hipótesis de partida con respecto a la problemática suscitada.

- **Capítulo 2: Estado del Arte:** Este capítulo tiene su base en el análisis exhaustivo de los diversos tópicos que se ven inmiscuidos en la presente investigación; En primera instancia, se analizan las ontologías de dominio y sus tipos, las principales taxonomías de estas, los lenguajes utilizados para su modelación, las propiedades de las relaciones *is_a* y *part_of*, la aplicabilidad ontológica y herramientas existentes. Por otro lado, se contextualizan temas como: El E-learning, standares y la Especificación IMS-LD. Además, de manera más vasta se analizan la representación del Diseño Instruccional y sus teorías, en particular la Teoría Elaborativa y su Estructura de Conocimiento Conceptual (ECC) que precisa.

- **Capítulo 3: Modelado para la representación de una Estructura de Conocimiento Conceptual:** Este capítulo contempla el modelado general que marca las directrices para la representación de una ECC automatizada, mediante el tratamiento taxonómico de relaciones ontológicas. En primera instancia se contempla el proceso improvisado actual (TGS) para representarla, sugiriendo su automatización a partir del estudio de la confección de relaciones de tipo y de partes.

- **Capítulo 4: Implementación de la solución:** Esta sección, proyecta la implementación de la representación a partir del actual proceso (TGS) por medio de mecanismos automatizados. En primera instancia, se realiza una propuesta de solución para cada relación, plasmando además los desafíos enfrentados durante el desarrollo. Luego, se realiza la implementación propiamente tal, cuyo resultado en una aplicación denominada *OntoConcept* que permite representar una ECC e ilustrarla gráficamente a través de su extensión *OntoConcept_Graph*. En virtud de lo anterior, una segunda etapa es el proceso de vinculación que sostiene dicha herramienta dentro del entorno de la aplicación Terpsícore, y cuyo nexo permite representar en su totalidad la Teoría Elaborativa.

- **Capítulo 5: Trabajos Futuros:** Este capítulo presenta las proyecciones venideras que sostendrá la presente investigación. Esta se sostiene en dos direcciones: La evaluación de la aplicación y su vinculación con Terpsícore en los

próximos meses y las contribuciones plasmadas en el actual desarrollo de un artículo científico como apoyo al proyecto: "Uso de ontología formales en el Diseño Instruccional"del Departamento de Desarrollo e Innovación de la Universidad del Bío-Bío.

- **Capítulo 6: Conclusiones:** En este capítulo se describen las aportaciones de este trabajo, tanto prácticas como teóricas, además de sus conclusiones y el cumplimiento cabal con los objetivos planteados a comienzos de esta investigación.

La figura 1.1, muestra en forma gráfica la estructura de la presente memoria.

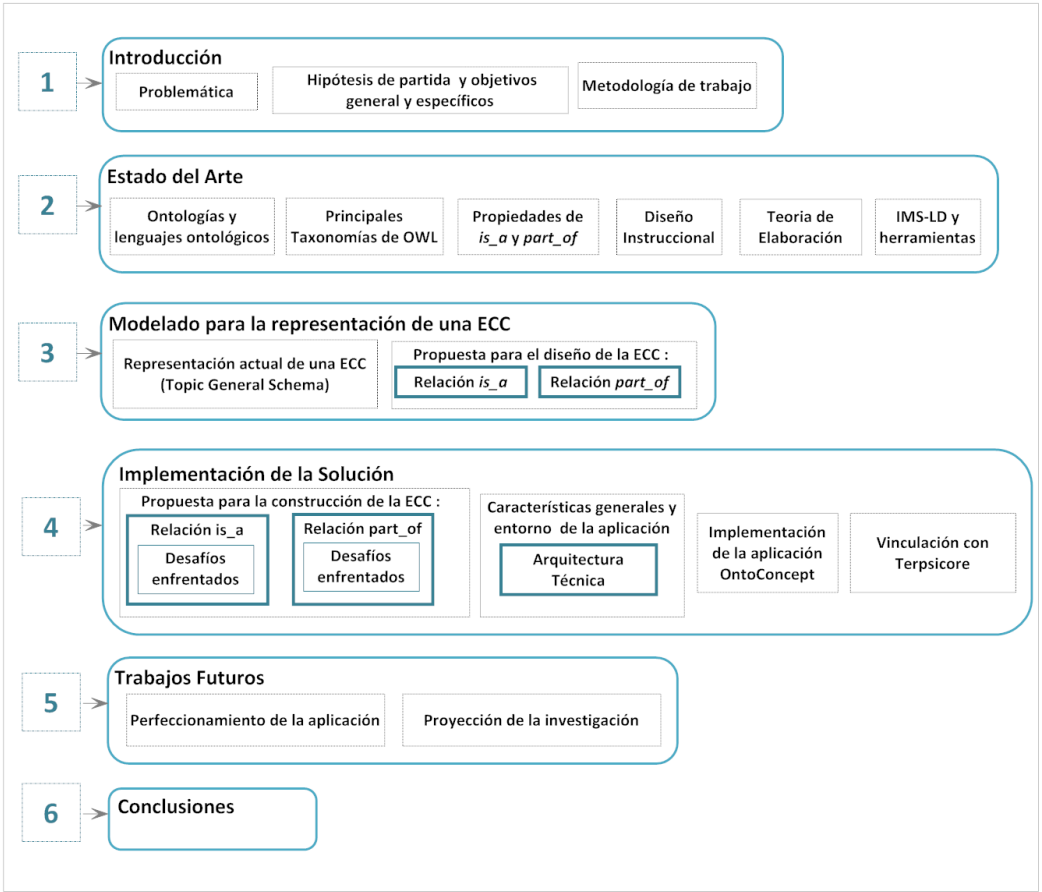

Figura 1.1: Esquema general - Esquema general de la investigación

2

Estado del Arte

2.1 Introducción

En este capítulo se presentan los temas más relevantes para la finalidad de esta investigación, en primera instancia, se analiza la importancia de las ontologías dentro del área de representación de conocimiento y su aplicabilidad dentro del E-learning. Además se realiza un análisis profundo de la composición y características que presentan en su estructura, cimentando los lineamientos que permiten el tratamiento de sus relaciones taxonómicas por medio de lenguajes ontológicos. A continuación, se profundiza en el Diseño Instruccional y sus teorías asociadas, particularmente sobre la Teoría Elaborativa y sus características, con especial énfasis en la Estructura de Conocimiento Conceptual que precisa. Además, en virtud de brindar un marco de aplicabilidad pedagógica, se analizan tópicos asociados al área, contemplando la proliferación del E-learning, las tecnologías al servicio del aprendizaje y sus principales conceptos asociados; Recursos de Aprendizajes, Objetos de Aprendizajes y Metadatos, así como la especificación IMS-LD cuyo uso está sujeto a la evaluación de una Estructura de Conocimiento Conceptual automatizada que vislumbra esta investigación; dentro del entorno de la herramienta IMS-LD, ReCourse, y particularmente dentro del marco de la herramienta adosada a este, Terpsícore, la cual analiza la conformidad de un Learning Design, elaborado en ReCourse respecto a diversas Teorías de Diseño Instruccional. Sin embargo, actualmente no existe un proceso automatizado para representar una ECC, que dicta la Teoría Elaborativa, por lo cual, durante las siguientes secciones se presenta un marco teórico, de los principales tópicos que conducen a modelarla.

2.2 Ontologías

El término ontología tiene sus orígenes en la filosofía aristotélica, y es una rama de la metafísica conocida como *metaphysica generalis* o metafísica general que alude a la investigación filosófica de la existencia. A diferencia de la *metaphysica specialis* que centra su estudio en la divinidad de Dios, el alma y el universo, la ontología busca dar respuesta a la pregunta ¿Qué tipos de cosas hay?, conduciendo al estudio de las categorías generales o conceptualización para todas las cosas que existen con propósitos de clasificación y jerarquización.

Actualmente este sentido filosófico de una ontología, es sucedido hacia un enfoque centrado en la representación de conceptos acerca de un dominio; precisando a una ontología como la especificación explicita de una conceptualización [Gruber, 1993](1), definición que fue ampliada posteriormente a la especificación formal de una conceptualización compartida dentro de un dominio de interés [Borst, 1997](2); la que plantea un lenguaje formal de manera que pueda ser procesada por máquinas y así permitir compartir, reutilizar y analizar el conocimiento. No obstante, la definición de ontología más acertada, desde el ámbito de las ciencias de la computación fue la presentada por [Sowa, 1999](3), el cual relacionó ontologías, lenguajes formales y programas informáticos como la representación del conocimiento bajo las teorías y técnicas de los siguientes tres campos:

- Lógica, que provee la estructura formal y las reglas de inferencia. Sin esta una representación del conocimiento seria vaga, sin ningún criterio para determinar si las declaraciones son redundantes o contradictorias.

- Ontología, que define los tipos de cosas que existen en el dominio de la aplicación, Sin esta los términos y los símbolos no estarían bien definidos, generando confusión.

- Computacional, que admite aplicaciones que distinguen la representación del conocimiento de la filosofía pura. Sin modelos computables, la lógica y la ontología no se podrían implementar computacionalmente.

En virtud de lo anterior, la representación del conocimiento es la aplicación de la lógica y la ontología a la tarea de construir modelos computables para un dominio.

2.2.1 Tipos de ontologías

Muchos autores han catalogado las ontologías dependiendo de su uso [Van Heijst, Schereiber y Wieringa, 1996](4), nivel de generalidad [Guarino, 1998](13), y nivel de especificación de las relaciones entre los términos [Gómez-Pérez, Fernández-López & Corcho, 2003](14), pero se distingue dentro de la línea de investigación de este estudio la clasificación realizada por [Devedziz, 2006](5), considerando a un ontología educacional a cualquiera que pueda ser utilizada en la enseñanza a través de tecnologías.

Devedziz propuso la siguiente categorización de las ontologías:

- **Ontología de dominio:** Representa los conceptos esenciales, relaciones y teorías de los diferentes dominios de interés.
- **Ontología de tareas:** Los conceptos y relaciones que se incluyen en este tipo de ontología pertenecen a los tipos de problemas, estructuras, partes, actividades y pasos a seguir en el proceso de solución de problemas.
- **Ontología de apoyo a la estrategia de la enseñanza:** Permite modelar experiencias en la enseñanza, especificando el conocimiento y los principios de las diferentes acciones pedagógicas y comportamientos.
- **Ontología de modelo de aprendizaje:** Se utiliza para construir modelos y en sistemas adaptativos de apoyo al aprendizaje.
- **Ontología de interfaz:** Especifica el comportamiento adaptativo y las técnicas en el nivel de interfaz de usuario.
- **Ontología de comunicación:** Se utiliza en el intercambio de mensajes entre diferentes plataformas, repositorios y servicios educativos. Define la semántica en que se basaran los mensajes, por ejemplo, el vocabulario de términos que se utilizaran en la comunicación.
- **Ontología de servicios educacionales:** Se relaciona con la ontología de comunicación, está basada en OWL-S y proporciona medios para crear descripciones de los servicios educacionales, procesables por ordenadores.

2.2.2 Principios del XML, RDF y OWL sobre Ontologías de Dominio

La necesidad de un lenguaje ontológico, surge a partir de la carencia de un vocabulario semántico comprensible por máquinas y que puedan ser utilizados en la web, a través de metadatos, que permiten describir y compartir contenidos en esta, con fines de interoperabilidad y reúso de contenidos. Ante esto la W3C (28) principal impulsora de la Web Semántica, introdujo la llamada arquitectura de tres capas, que permiten mediante la inclusión de estándares de representación y metalenguajes, interpretar una base de conocimiento de semántica computacional que permita a los datos ser reutilizados y compartidos. Para lograr este razonamiento lógico sobre datos mediante técnicas propias de la **Inteligencia Artificial**, era necesario un lenguaje de programación lógico e interpretativo capaz deducir y validar nueva información, el lenguaje **OWL**, construido sobre la base de las siguientes capas:

El modelo de tres capas consta de los siguientes niveles, representados en la figura 2.1:

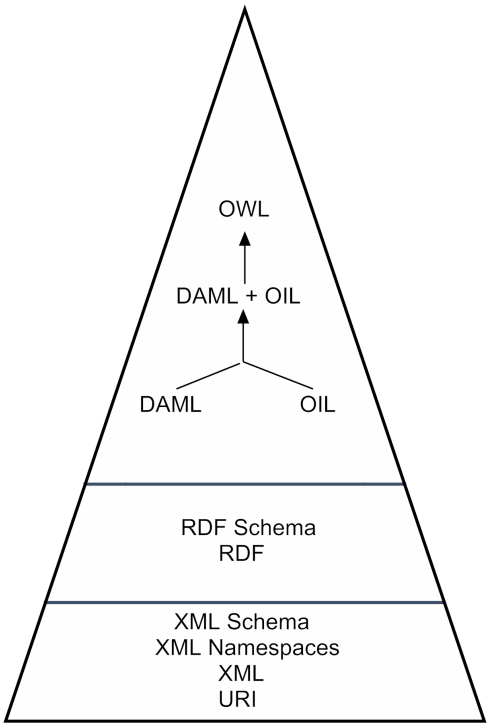

Figura 2.1: Modelo de tres capas - Jerarquía de los lenguajes antecesores de OWL.(6)

1. **Primer Nivel:** Permite el intercambio de datos. Este nivel contiene los siguientes elementos:

   - **Unicode:** Es un alfabeto y un medio para codificar texto mediante la unión de caracteres. El Unicode permite disponer información en la web semántica, en diferentes idiomas.
   - **URI**: Son considerados identificadores de objetos y enlaces de acceso a cualquier contenido Web. Una URI está compuesta de una secuencia de caracteres que identifica a un recurso abstracto o físico. Es decir cualquier objeto puede identificarse a través de una URI.
     Cabe destacar que existe disimilitud entre URIs y URLs, las URIs **identifican** mientras que las URLs **localizan**; sin embargo las localizaciones también son identificaciones, por lo que cada URL es también una URI, pero hay URIs que no son URLs. Por ejemplo, *Juan Felipe* es un nombre por lo tanto la identificación de una persona, es decir, una URI pero no URL ya que no entrega información sobre la localización o como ponerse en contacto, mientras que *Avenida Collao 1202, Concepción* es una posición, una identificación de la ubicación física, por lo que es una URI y a la vez una URL.
   - **XML + Namespace(NS) + XML Schema:** Son tecnologías que permiten el intercambio de datos en la web. XML ofrece un formato común para el intercambio de documentos, NS se define como una URI que permite asociar un conjunto de nombres, dentro de la estructura de datos que están definidos, por ejemplo, los namespace en XML se representan mediante la palabra xmlns, mientras que en RDF mediante rdf. XML Schema es un lenguaje para describir la estructura y restringir el contenido de un documento XML.

   Cabe destacar que XML, como base ofrece flexibilidad, extensibilidad, interoperabilidad y claridad, pero por sí solo, no permite representar, validar ni modelar, ni menos inferir una semántica sobre los datos.

2. **Segundo nivel:** Permite la creación de sentencias o declaraciones y definir relaciones semánticas entre distintas URIs asociándoles un conjunto de propiedades y valores.
   El **Framework RDF** es un modelo base para poder describir recursos. Consiste principalmente de una sentencia conformada por una tripleta, compuesta de un sujeto, predicado y objeto, en donde, el sujeto lo relacionamos con un recurso, al predicado con una propiedad, y al objeto al valor de la propiedad, por ejemplo:

   <sujeto, predicado, objeto>=<alumno_1, nombre, "Juan Felipe">

   RDF fue diseñado para su uso en la web, por lo tanto, cada sentencia describe a un recurso que a su vez puede ser referenciado por una URI. Una sentencia llevada al primer nivel, se representa mediante la etiqueta <Description>en donde el atributo *about* describe al sujeto y *resource* indica los recursos que pueden ser nombrados por una URI. Cada elemento dentro de la descripción se considera propiedades de ese sujeto, por ejemplo:

```
1 <rdf:Description rdf:about=   #alumno_1 >
2   <rdf:nombre rdf:resource=   #Juan Felipe />
3 </rdf:Description>
```

   Para poder representar afirmaciones en RDF podemos usar:

   - **Declaración explicita:** mediante el uso de la expresión lógica (x, P, y), en donde, el predicado binario P relaciona el objeto x con el objeto y.
   - **Grafos RDF:** representa a la tripleta <sujeto, predicado, objeto>como un grafo en el que se conectan los dos nodos correspondientes a los elementos sujeto y objeto a través de un arco que representa al predicado.2.2

   Por otro lado, RDF solo permite modelar recursos y sus relaciones, lo que limita la expresividad al momento de describir propiedades y clases de un recurso, ya que no posee un vocabulario semántico necesario para poder realizar afirmaciones sobre los recursos, ante esto surge una extensión de RDF llamada **RDF Schema**

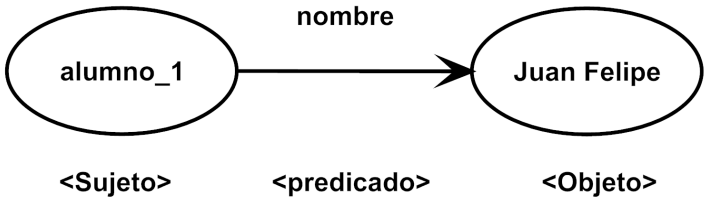

Figura 2.2: Grafo RDF - Representación de un Grafo RDF

(RDFs)[I], con el propósito de mejorar la semántica añadiendo un nuevo vocabulario, que permite expresar el significado, características y relaciones de un conjunto de propiedades, e incluir restricciones y herencia de propiedades.

3. **Tercer Nivel:**

   - **DAML + OIL:** Es un nombre compuesto de las siglas DML (DARPA Agent Markup Language) y OIL (Ontology Inference Layer), este surge de las limitaciones de RDF Schema incorporando la lógica descriptiva que son formalismos basados en la lógica para la representación del conocimiento y permitiendo a la maquinas realizar inferencias sobre el conocimiento y obtener razonamiento lógico.

   En virtud de las capas mencionadas, su integración y aportes de nuevas etiquetas que componen una base de conocimiento con una semántica comprendidas por máquinas, se modela el lenguaje OWL, con mayor poder expresivo, permitiendo la confección de ontologías robustas así como fomentar el reúso e interoperabilidad de estas.

[I]Los constructores aportados por RDF Schema, se encuentran detallados en :`http://www.w3.org/TR/2002/WD-rdf-schema-20021112/`

2.2.2.1 Lenguaje de ontologías web (OWL)

OWL es un lenguaje de marcas para definir ontologías compatibles con la arquitectura web actual (Word Wide Web) y recomendado por el consorcio W3C, está diseñado para usarse cuando la información web necesita ser interpretada por maquinas. Se basa en las tripletas RDF, pero posee un mayor poder expresivo que todos sus predecesores, incorporando mejoras en aspectos como la escalabilidad de la web, compatibilidad con estándares de accesibilidad y distribución, permitiendo una interoperabilidad entre ontologías. OWL incorpora el siguiente vocabulario, enriqueciendo aún más la semántica de RDF Schema:

- **Restricciones de cardinalidad:** Permite restringir el número de valores que una propiedad puede tomar, por ejemplo, definir que una persona solo puede tener dos padres, o que un curso es impartido al menos por un profesor.
- **Disyunción de clases:** Permite decir que las clases son disjuntas. Por ejemplo, hombres y mujeres son disjuntos, en el modelo RDFs solo podríamos afirmar relaciones de subclases, por ejemplo, la mujer es una subclase de persona.
- **Combinaciones booleanas:** Permite construir nuevas clases mediante la combinación de otras, utilizando la unión, la intersección y complemento, por ejemplo, definir una clase persona como la unión disjunta de las clases masculinas y femeninas.
- **Restricciones de rango:** Permite definir el rango de una propiedad, por ejemplo, si definimos "comen"podemos decir que las vacas comen plantas, mientras que otros animales pueden comer carne, también.
- **Características especiales de las propiedades:** Permite indicar que una propiedad es transitiva "mayor que", única "es la madre de", o inversa de otra propiedad como "come"y "es comido".
- **Tipos de datos:** Incorpora los tipos de datos de XML Schema los cuales son los tipos de los lenguajes de programación (RDFs solo posee datos literales como cadenas de caracteres) [I].

[I] Los tipos de datos aportados por XML Schema, se encuentran detallados en :`http://www.w3.org/TR/xmlschema-2/`

Además, es pertinente mencionar que la influencia más importante de OWL es a través de su predecesor DAML + OIL, del cual hereda el paradigma de marcos (frames) y la lógica descriptiva. Esta última se observa en la formalización de la semántica, en los constructores del lenguaje y en la integración de tipos de datos y valores. Un aspecto relevante, es que la semántica de este lenguaje, basada en lógica descriptiva, posee procedimientos de decisión que pueden ser ejecutados por sistemas de razonamiento automatizado [Horrocks, 2007](7). Por otro lado, el uso del paradigma de representación de conocimiento denominado frames, el lenguaje OWL proporciona beneficios para usuarios no expertos en lógica descriptiva. Estos beneficios se relacionan con la facilidad para leer y entender las ontologías al agrupar la información de cada clase en un esquema compacto. El paradigma de frames ha sido usado por diversas herramientas de diseño de ontologías y representación de conocimiento entre ellos, la herramienta Protégé Ontology design tool [Grosso, 1999](8).

OWL ofrece tres sub-lenguajes de expresividad incremental, diseñados para comunidades específicas de desarrollo en base a su nivel de expresividad:

- **OWL Lite[I]:** Es el lenguaje más básico de aplicar, ya que posee una expresividad restringida. Esta recomendado para usuarios que necesitan una clasificación jerárquica y restricciones simples. Por ejemplo, en las restricciones de cardinalidad, solo permite valores cardinales de 0 o 1. Si bien posee más baja expresividad y complejidad formal que su sucesor, pero supone una mayor eficiencia.

- **OWL DL[II]:** Contiene todo el vocabulario de OWL pero introduce restricciones en sus usos, por ejemplo, mientras una clase puede ser una subclase de otras clases, una clase no puede ser instancia de otra. Esta recomendado para usuarios que quieren la máxima expresividad asegurando una decidibilidad (todas las conclusiones son garantizadas para ser computables) y resolubilidad (las operaciones terminaran en tiempo finito). OWL DL viene de Description logics (lógica descriptiva), un campo de investigación que estudia la lógica que compone la base formal de OWL. OWL DL adiciona las siguientes expresiones o axiomas en el contexto de la presente investigación:

  - Permite el manejo de propiedades
  - y permite la definición de individuos

- **OWL Full:** Posee la mayor expresividad y es totalmente compatible con RDF tanto sintácticamente como semánticamente, por lo que, cualquier documento RDF y RDF Schema es compatible con OWL Full. Esta recomendado para usuarios que requieren el máximo de expresividad y libertad sintáctica de RDF, pero sin garantías computacionales. Actualmente es poco probable que algún software racional pueda soportar por completo el razonamiento para cada característica de OWL Full.

[I] Los constructores aportados por el lenguaje OWL Lite, se encuentran detallados en :`http://www.w3.org/TR/owl-features/#s2.1`

[II] Los constructores aportados por el lenguaje OWL DL, se encuentran detallados en :`http://www.w3.org/TR/owl-features/#s2.2`

No obstante, cabe destacar la compatibilidad ascendente entre los tres sub-lenguajes, en donde:

- Cada ontología legal OWL Lite es una ontología legal OWL DL.
- Cada ontología legal OWL DL es una ontología legal OWL Full.
- Cada conclusión válida OWL Lite es una conclusión válida OWL DL.
- Cada conclusión válida OWL DL es una conclusión válida OWL Full.

La compatibilidad entre los tres sub-lenguajes y su antecesor RDF es la siguiente:

- OWL Full es visto como una extensión de RDF.
- OWL Lite y OWL DL es visto como extensiones de forma restrictiva de RDF.
- Cada documento OWL (Lite, DL, Full) es un documento RDF, y cada documento RDF es un documento OWL Full, pero solamente los documentos RDF serán un documento OWL Lite u OWL DL legales.

Desde el punto de vista del desarrollador de ontologías, la elección entre OWL Lite y OWL DL dependerá de la mayor o menos extensión que los usuarios requieran de los constructores expresivos provistos por OWL DL. La elección entre OWL DL y OWL Full dependerá, principalmente, de la extensión que los usuarios requieran de RDF Schema. Cuando se usa OWL Full en comparación con OWL DL, la razón es menos predecible porque las implementaciones completas de OWL Full no existen actualmente.

2.2.3 Principales elementos de las taxonomías en OWL

- **Clase** *Class*: Define un grupo de individuos que pertenecen juntos porque comparten las mismas propiedades. Por ejemplo, *VegetarianPizza* son miembros de la clase *Pizza*. Figura 2.3

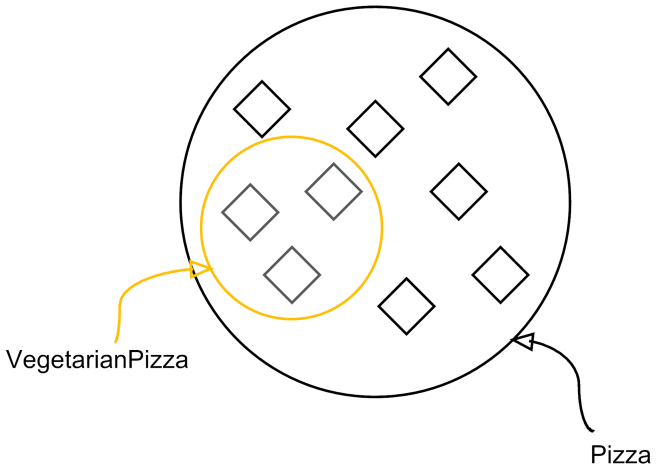

Figura 2.3: Clase - Diagrama de representación de una subclase(9)

- **Intersección** $intersectionOf(C_1, C_2...)$ $C_1 \cap C_2$: Permite establecer intersecciones entre clases. Por ejemplo *CheeseyVegetableTopping* se puede describir como la intersección de *CheeseTopping* y *VegetableTopping*.Figura 2.4

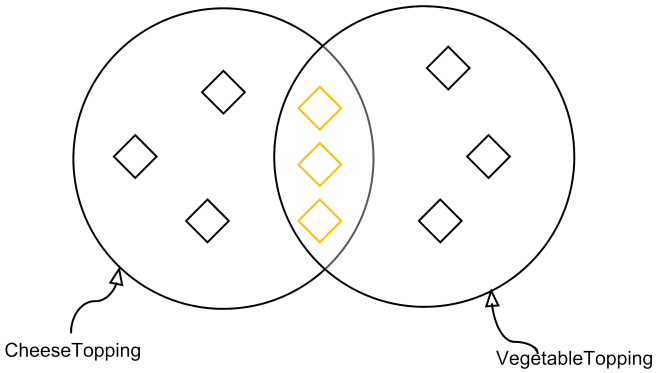

Figura 2.4: Intersección - Diagrama de representación de una intersección(9)

- **Unión** $unionOf(C_1, C_2...)$ $C_1 \cup C_2$**:** Una clase unión se produce utilizando el operador OR con dos o más clases. Por ejemplo podemos indicar que una clase contiene elementos que son *FruitTopping* y *SultanaTopping*. Figura 2.5

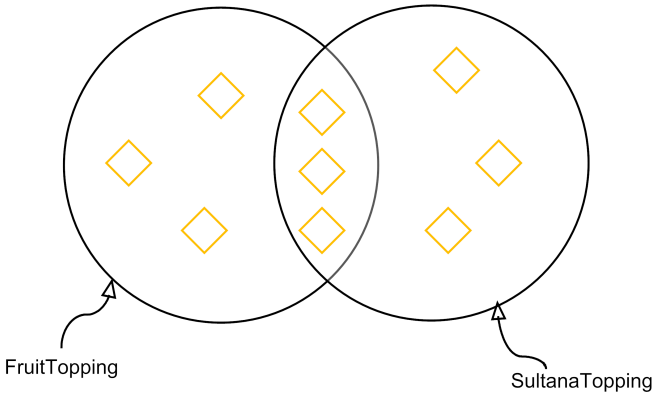

Figura 2.5: Unión - Diagrama de representación de una unión(9)

- **Complemento** $complementOf(C)$ $owl : Thing = \Delta$, $\Delta \backslash C$**:** Una clase complemento se especifica mediante la negación de otra clase. Esta contendrá a los individuos que no están en la clase negada. Por ejemplo, la clase complemento de la negación de la clase *VegetableTopping*, contendrá a los individuos que solo sean *CheesyPizza*. Figura 2.6

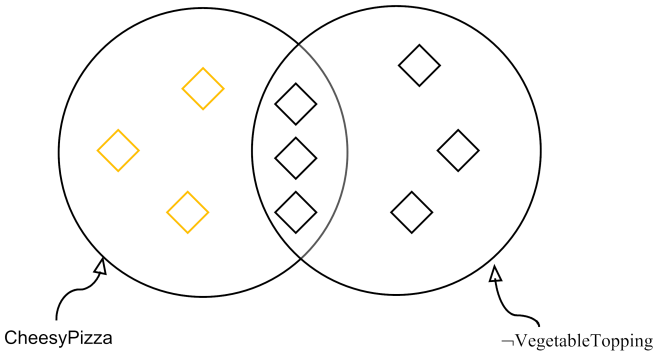

Figura 2.6: Complemento - Diagrama de representación de un complemento(9)

- **Restricciones de propiedad** *Restriction*: Las propiedades *Property* pueden utilizarse para establecer relaciones entre individuos o desde individuos a valores de datos. Una restricción de propiedad establece restricciones sobre la forma en que las propiedades son utilizadas por las instancias de una clase. Una propiedad restringida en RDF se expresa con *onProperty*.

  Las restricciones de propiedad pueden ser agrupadas en tres categorías:

  1. **Restricciones de cardinalidad** $U\ minCardinality(n)\ \{a \in \Delta |\ |\{b|(a,b) \in U\}| \geq n\}$**,** $U\ maxCardinality(n)\ \{a \in \Delta |\ |\{b|(a,b) \in U\}| \leq n\}$**:** Para una determinada propiedad, las restricciones de cardinalidad nos permiten hablar del número de relaciones de una clase de individuos.
  2. **Restricción de valores de propiedad** $U\ hasValue(v)\ \{x|(x,v) \in U\}$**:** Permite que una propiedad tenga a un determinado individuo como un valor. Figura 2.7

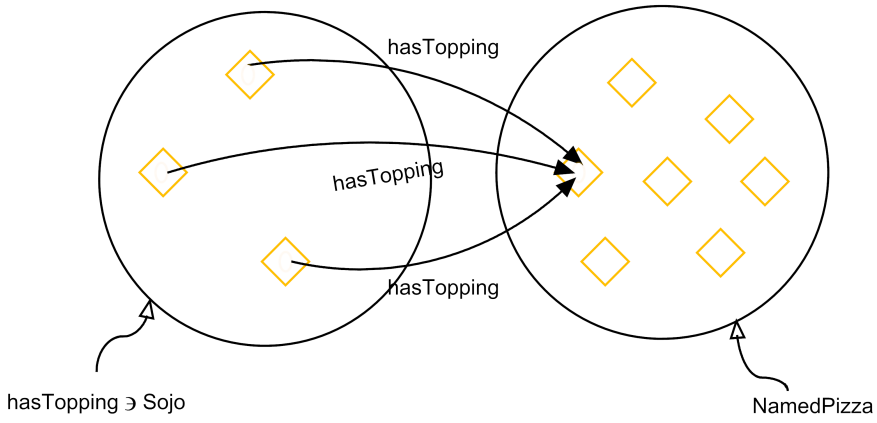

Figura 2.7: hasValue - Diagrama de representación de una restricción de valores de propiedad(9)

  3. **Restricciones de cuantificadores (Existencial** $\exists$**, Universal** $\forall$**):**
     - **Una restricción existencial** $U\ someValuesFrom(D)\ \{x|\exists y\ (x,y) \in U\ \cup\ y \in D\}$**:** Describe una clase de individuos que tiene al menos un tipo de relación a lo largo de una propiedad específica a un individuo que es miembro de una clase especificada.Figura 2.8

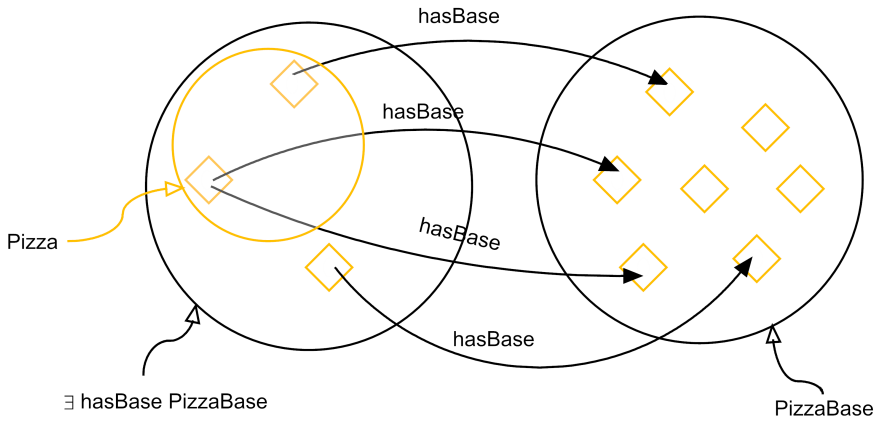

Figura 2.8: someValuesFrom - Diagrama de representación de una restricción existencial(9)

– **En la restricción universal** $U\ allValuesFrom(D)\ \{x|\forall y\ (x,y) \in U \rightarrow y \in D\}$**):** todos los individuos de una clase, deben ser miembros de una clase especificada. Figura 2.9

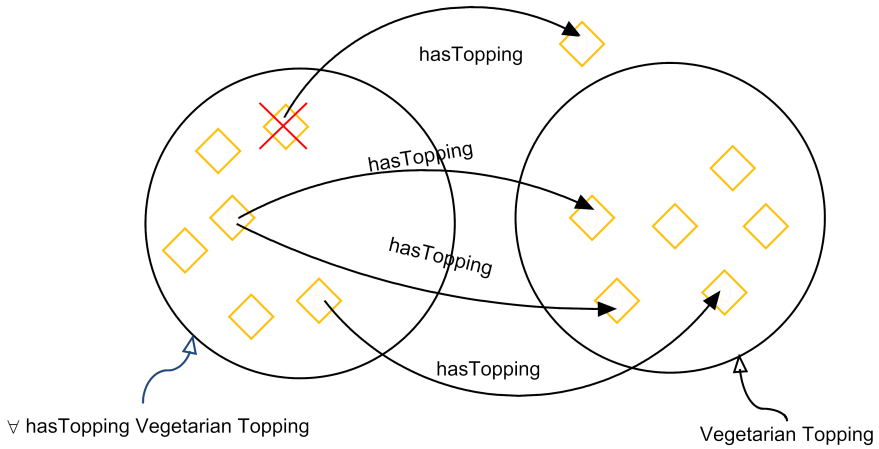

Figura 2.9: allValuesFrom - Diagrama de representación de una restricción universal(9)

- **Clases enumeradas** $EnumeratedClass(A_1\ o_1...o_n)\ A = \{o_1, ..., o_n\}$**:** las clases se pueden describir mediante la enumeración de los individuos que la componen. Los miembros de la clase son exactamente el grupo de los individuos enumerados. Figura 2.10

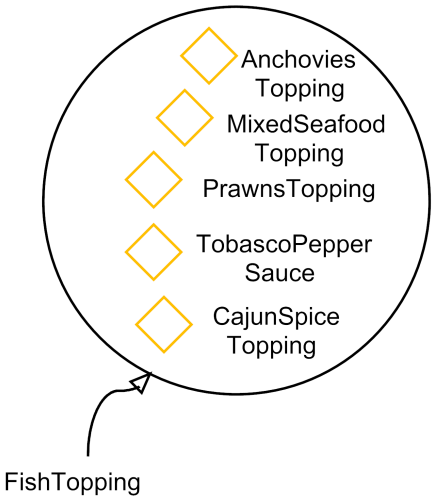

Figura 2.10: Clases Enumeradas - Representación de una clase enumerada(9)

2.2.4 Propiedades ontológicas *is_a* y *part_of* para el modelamiento del conocimiento

En el presente estudio, se analizan particularmente dos tipos de relaciones ontológicas necesarias para poder representar la Teoría de Elaboración, en base a la estructura de conocimiento conceptual que presenta en su modelamiento. Las relaciones de tipo y de partes que dicta esta teoría, son modeladas ontológicamente, como *is_a* y *part_of* respectivamente y las cuales sostienen los fundamentos que permiten la modelación.

Cabe señalar que los conceptos tanto *is_a* como *part_of*, son representaciones tentativa para poder distinguir estas relaciones durante la presente investigación, puesto que en la modelación ontológica difiere en cada ontología, siendo la principal critica la falta de una estandarización a la hora de modelar estas.

- **Relación *is_a*:** Es la más simple, se representan estructuralmente en la ontología, por medio de jerarquías de clases, las que se disponen indicando que una clase es subclase de otra. por ejemplo, si decimos A *is_a* B, significaría que A es un

"subtipo"de B y no una "instancia de". La figura 2.11, representa la relación *is_a*, de manera conceptual.

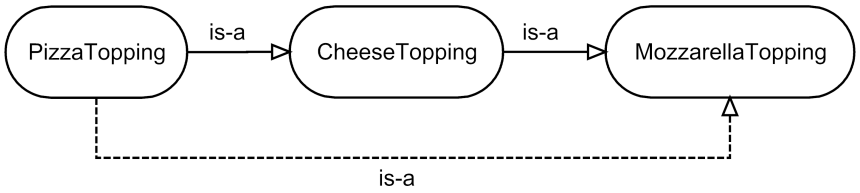

Figura 2.11: Relación *is_a* Valida - Diagrama que representa relaciones *is_a* validas

La dirección que tenga una relación *is_a* permitirá inferir que B es una subclase y un contenido específico, mientras que A sería una clase y posee contenido más general que B. Un ejemplo en contraposición, comprende la figura 2.12, en donde la doble dirección de la relación entre *PizzaTopping* y *CheeseTopping* hace imposible la inferencia, ya que debe existir una clase más general que otra.

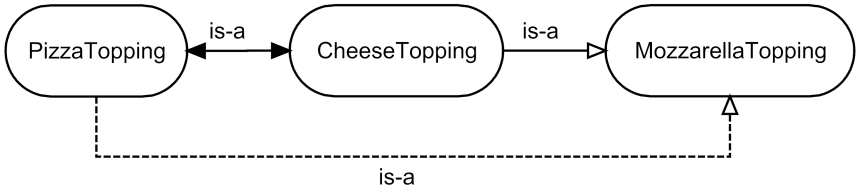

Figura 2.12: Relación *is_a* Invalida - Diagrama que representa relaciones *is_a* invalidas

- **Relación *part_of*:** Permite definir que un contenido es parte de otro, la figura 2.13, ejemplifica esta relación, en donde una relación entre A y B existiría solo si B es necesariamente una parte de A; donde B existe porque es una parte de A y la presencia de B implica la existencia de A.

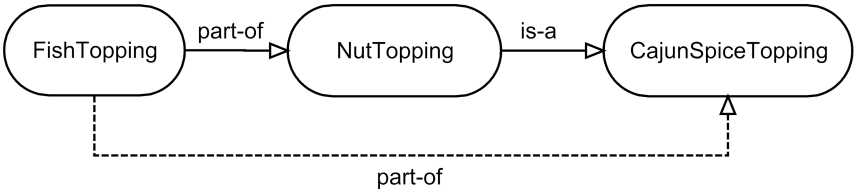

Figura 2.13: Relación *part_of* - Diagrama que representa una relación *part_of*

Si un *part_of* es seguido por un *is_a*, es equivalente a decir *part_of*, por ejemplo, si A *part_of* B y B *is_a* C, entonces podemos inferir que A *part_of* C.

Tanto *is_a* como *part_of*, poseen transitividad, reflexividad y anti-simetría. Pero la reflexividad y anti-simetría solo son a nivel lógico, y se generan por medio de la jerarquización, ya que OWL no las implementa ni incorpora una solución alternativa a ellos.

- **Transitividad:** si A es parte de B y B es parte de C, entonces A es parte de C.
- **Reflexividad:** A es parte de A.
- **Anti-simetría:** si A es parte de B y A es distinto a B, entonces B no es parte de A.

2.2.5 Diferencias entre las ontologías y otros modelos conceptuales

Sin duda, la concepción de una ontología como la representación de un dominio, presenta cierto desasociego interpretativo en comparación a los modelos tradicionales de representación, al tener alto grado de similitud. Las ontologías se suelen parecer a otros tipos de modelos conceptuales utilizados en las ciencias de la computación. Principios como la interrelación, la instanciación, generalización/especificación, también se pueden encontrar, en los diagramas de clase o entidad/relación; No obstante, hay una diferencia sutil entre ellas en términos de uso y propósito, ya que por un lado las ontologías son *descriptivas*; es decir capturan el conocimiento observado para representar un dominio, en contraposición a los modelos conceptuales, que presentan un enfoque *prescriptivo*, implicando una conceptualización anticipada en las que enmarca su diseño para la construcción de un sistema nuevo. El principal objetivo de los modelos conceptuales es prescribir los componentes técnicos de un sistema de información y una vez que el sistema funciona, el modelo conceptual detrás de él ha cumplido su función y no se consultará ni modificará en tiempos de ejecución. Mientras que el principal objetivo de una ontología es servir como fuente de conocimiento del dominio para un sistema de información que basa sus decisiones en tiempo de ejecución por medio de la realización de inferencias.

Desde un punto de vista más general, la relación de modelos conceptuales y de ontologías mantiene el dilema de similitud, no obstante, la distinción más generalizada

es concebir a una ontología como una especialización de los modelos, entendiendo este ultimo como una representación mental, que utiliza un lenguaje formal, evitando el carácter subjetivo y ambiguo que pueda tener un lenguaje natural. Sin embargo, tanto las ontologías como los modelos son generados a partir del mismo proceso mental pero centrado en objetivos diferentes. La figura 2.14, representa el proceso mental, y la especialización de la ontología sobre los modelos.

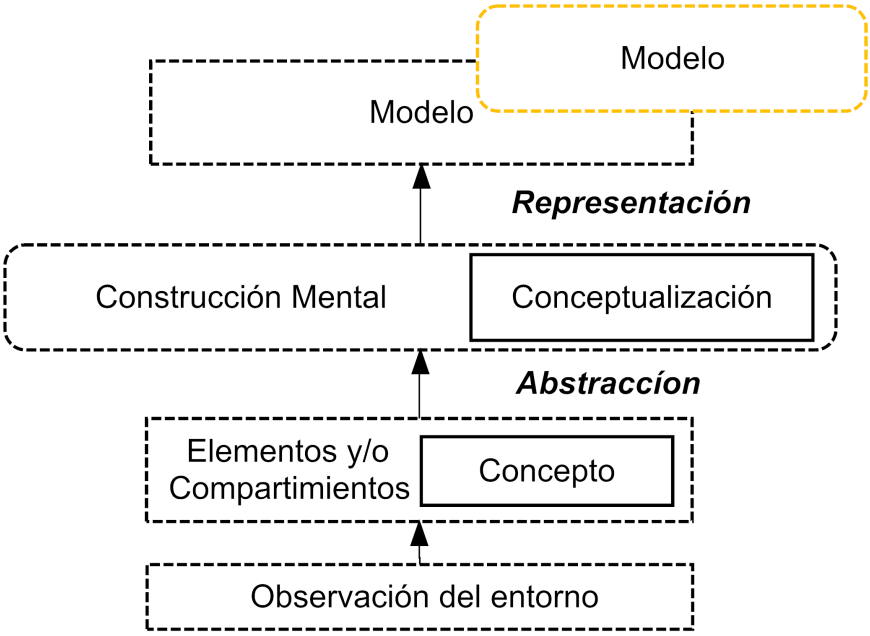

Figura 2.14: Especialización de la Ontología - Diagrama que representa la especialización de la ontología sobre los modelos(10)

2.2.6 Razones de aplicabilidad ontológica

La ingeniería del conocimiento y el procesamiento en base a ontologías son unos de los problemas actuales de la informática; esta última juega un rol significativo en el desarrollo de diversas áreas, principalmente orientadas hacia la web, destacando las siguientes:

- **Comercio electrónico**: Se utilizan para realizar una descripción del dominio, en este caso un sector comercial, que describe a los usuarios, sus tendencias y relaciones de los mismos, así como la categorización de los productos a vender

y sus características[I] . El uso de ontologías en esta área, facilita la agrupación (clustering) en conjuntos de usuarios con comportamientos e intereses similares. La obtención de estos grupos permitirá realizar estudios de mercado de forma más sencilla y también la personalización a mostrar al usuario según sus intereses.

- **Búsqueda de información**: Se utilizan para la anotación de recursos en internet (documentos, páginas web e imágenes) y para guiar la búsqueda en un dominio concreto. Esto proporciona una mayor flexibilidad de búsqueda, incrementando la precisión y la recuperación de los documentos buscados. Su fundamento esta en contar por medio del tratamiento ontológico con un nuevo paradigma web, basado en búsquedas semánticas, también conocido como web semántica o web 3.0. Ejemplo de esto es el Consorcio WWW (W3C) el cual está desarrollando RDF: Resource Description Framework [Brickely y Guha, 1999](29), un lenguaje para codificar conocimientos en páginas Web para hacerlas entendibles a los agentes electrónicos que buscan información. Por otro lado, la agencia de Proyectos de Investigación Avanzada en Defensa (en inglés, DARPA: Defense Advanced Research Projects Agency), conjuntamente con el W3C, está desarrollando el Lenguaje de Marcas de Agente DARPA (en inglés, DAML: DARPA Agent Markup Language) extendiendo RDF con construcciones más expresivas buscando facilitar la interacción de agentes en el Web [Hendler and McGuinness, 2000](30).

- **Biomedicina**: Este campo tiene el mayor interés por parte de los ingenieros de ontologías e ingenieros del conocimiento, ya que en el existen numerosos recursos, como artículos de investigación y base de datos públicas. Este campo se orienta principalmente a proporcionar a los médicos acceso a las aplicaciones para soporte a la toma de decisión, y para el descubrimiento de la información. Ejemplos de estos grandes, estandarizados y estructurados vocabularios son SNOMED[II] [Price y Spackman, 2000](31) y la red semántica Sistema de Lenguaje Médico Unificado (en inglés, UMLS: Unified Medical Language System)[III] [Humphreys y Lindberg, 1993](32).

[I] Un claro ejemplo de esta área de aplicabilidad ontologica es Amazon: `http://www.amazon.com`

[II] Para mayor información,SNOMED cuenta con una pagina web oficial: `http://www.ihtsdo.org/snomed-ct/`

[III] Para mayor información sobre UMLS, visite: `http://www.nlm.nih.gov/research/umls/`

- **Bibliotecas digitales**: Estas permiten el acceso a grandes cantidades de información en forma de documentos digitales, que pueden tener formatos muy variados y estar distribuidos en sistemas informáticos dispares. Las técnicas basadas en ontologías permiten tratar esta heterogeneidad mediante la descripción de objetos y repositorios, posibilitando un acceso sencillo, consistente y coherente a los recursos digitales.

- **E-learning**: Dentro de este marco, las ontologías han encontrado una excelente área de aplicación. De acuerdo a [Hernández & Saiz, 2007](11) se considera una ontología educacional a cualquiera que pueda ser utilizada en la enseñanza basada en tecnologías). En E-learning, las ontologías han sido utilizadas para describir sistemáticamente cada objeto de aprendizaje, permitir búsquedas semánticas y dar a los usuarios un punto de referencia para los conceptos y la terminología compartida [Marengo et al., 2006](12).

En general, la ontología en cada una de estas áreas, definen un vocabulario común para compartir información sobre un dominio en particular. Esta particularidad, implica que su uso esté proliferando de manera rauda, gracias a las ventajas que precisa:

- Compartir el entendimiento común de la estructura de información entre personas o agentes de software.

- Permitir la reutilización del conocimiento sobre un dominio.

- Explicitar suposiciones sobre un dominio.

- Separar el conocimiento del dominio del dominio operacional.

- Analizar el conocimiento de un dominio.

Durante la presente investigación se utilizaran ontologías de dominio correspondientes a un repositorio de diversos grupos y sus proyectos de representación de conocimiento[I]
.

[I]Para mayor detalle, el repositorio de ontologías se encuentra alojado en el sitio web:`http://www.cs.utexas.edu/users/mfkb/related.html#o`

2.3 Diseño Instruccional

Los cambios que han afectado a la sociedad en el último tiempo, han impactado también en la forma en que se concibe, planifica y ejecutan los procesos educativos. A raíz de esto, se ha producido un cambio dentro de los lineamientos en relación a la personalización de la enseñanza, relaciones de colaboración entre profesores, estudiantes y un proceso de aprendizaje centrado en el aprendiz, entre otros. Actualmente las nuevas teorías pretenden dar sustento a los procesos de aprendizaje, de igual forma que los avances tecnológicos; quienes procuran dar soportes a la creación de estos ambientes, basados en el aprendiz y flexibles a diversos estilos de aprendizajes. Actualmente, muchos han intentado modelar tales ambientes, confeccionando un marco de referencia para dar sustento al Diseño Instruccional.

En base a lo anterior, se apoya el gran reto de utilizar métodos provenientes de las teorías de Diseño Instruccional, que a través de un modelado computacional permitan la creación de recursos de aprendizajes, reusables, interoperables, flexibles e interactivos.

El objetivo de esta investigación, se enfoca en la representación de una ECC que precisa la Teoría de Elaboración, para lo cual, en las siguientes secciones se presenta una perspectiva de las TDI, para luego puntualizar en la Teoría Elaborativa.

2.4 Teorías del Diseño Instruccional.

El Diseño Instruccional es la aplicación sistemática de teorías y principios que guían el diseño de aprendizajes. La aplicación de los métodos de las teorías de Diseño Instruccional requiere de un enfoque disciplinado que indique por ejemplo, la secuencia de las actividades y los resultados de cada una de ellas. Reigeluth los denomina Procesos de Diseño Instruccional [Reigeluth, 1999](15). Estos procesos comprenden las etapas instructivas que van desde el análisis, hasta la puesta en marchar y evaluación, por medio de una secuencia lineal, iterativa o incremental.

En virtud de lo anterior, modelos como los de Desarrollo de Sistemas Instruccionales (en inglés, ISD: Instructional Systems Development), permiten confeccionar sistemas intruccionales con un enfoque de sistema, creando recursos de aprendizajes e incorporando las etapas instructivas. No obstante, el proceso de Diseño Instruccional, no contempla una guía en base a que si un método instruccional es aplicable a un contexto

educativo, ni tampoco detalles de cómo realizar las actividades de los métodos instruccionales. En contraposición a lo anterior, son las Teorías de Diseño Instruccional (TDI) las que ofrecen una guía explicita de cómo ayudar a las personas a aprender [Reigeluth, 1999](15).

Las TDI, se caracterizan por:

- Ser orientadas al diseño, es decir centradas en la obtención de objetivos de aprendizaje definidos.
- Identifican métodos y situaciones. Los métodos instruccionales son formas para apoyar y facilitar el aprendizaje; y las situaciones indican cuando usar un determinado método.
- Los métodos pueden ser descompuestos en componentes de niveles más detallados.
- Los métodos son más probabilísticos que determinísticos, en el sentido de que el uso de alguno de ellos no asegura el logro de los objetivos, sino que aumenta la posibilidad de lograrlos.

Charles Reigeluth [Reigeluth, 1999](15), compila varias TDI, convirtiéndolo en un referente en el tema y sus obras en una importante fuente de conocimiento acerca de estas teorías. Reigeluth clasifica estas en 3 categorías en base al dominio en que sustenta el aprendizaje, ya sea a nivel cognitivo, psicomotor y afectivo. Algunas de las teorías presentadas por Reigeluth son: Learning By Doing, Collaborative Problem Solving, Landamatics for Teaching General Methods of Thinking, Multiple Intelligences, Instructional Transaction Theory y Elaboration Theory, entre otras.

Dada la importancia para el desarrollo de la presente investigación, en la siguiente sección se enfatiza en las características que dicta la Teoría de Elaboración (en inglés, Elaboration Theory), proporcionando una visión íntegra, con énfasis en la estructura de conocimiento conceptual que precisa la literatura y la cual será objeto de modelación mediante lenguajes ontológicos formales.

2.4.1 Teoría de Elaboración: Guía para el alcance y secuencia de decisiones.

El cambio de paradigma de la enseñanza centrada en el profesor y el contenido centrado en el alumno está creando nuevas necesidades de formas de secuenciar la instrucción. La Teoría de Elaboración, fue propuesta por Charles Reigeluth, cuyo objetivo principal es apoyar la selección de los contenidos a aprender, proporcionando formas de secuenciar los tópicos, de manera que fomenten el logro de los objetivos instruccionales, enmarcándose dentro del aprendizaje cognitivo y psicomotor de los estudiantes, aunque no necesariamente en el afectivo. La Teoría de la Elaboración fue desarrollada para proporcionar un enfoque holístico de la secuenciación que hace al proceso de aprendizaje más significativo y motivador para los alumnos.

A nivel general, la Teoría de Elaboración posee las siguientes características generales.

- Propone enseñar a partir de los conceptos más amplios, es decir, usando un enfoque de lo general a lo particular.
- Agrupa los contenidos en "episodios de aprendizaje", basado en que estos no deben ser tan pequeños implicando la interferencia en el flujo del proceso de aprendizaje ni tan grandes que dificulten la revisión y síntesis de los mismos.
- Propone dos estrategias de secuenciación de contenidos: por tópico y en espiral. La secuenciación por tópico, fomenta el aprendizaje de un tópico en la profundidad requerida y luego, el aprendizaje del tópico siguiente. La principal ventaja es que los estudiantes pueden concentrarse en un tema (o tarea) para profundizar en el aprendizaje, sin saltar a uno nuevo. Además, los materiales y otros recursos para el aprendizaje son usados en un bloque de tiempo, en lugar de ser utilizado en diferentes puntos dispersos en varios meses o un año.

  En relación a la secuenciación en espiral, los tópicos son aprendidos gradualmente en cada oportunidad en que se revisa el tópico, aumentando gradualmente la profundidad y amplitud requerida. Su principal ventaja es su construcción sintetizada, lo que permite que las interrelaciones entre tópicos (tareas) puedan ser más fácilmente aprendidas, debido a que este enfoque permite que aspectos

similares de los diversos temas se puedan aprender en un marco temporal cercano. No obstante, su principal desventaja es la interrupción, cambiar con frecuencia los temas interrumpe el desarrollo del raciocinio del aprendiz, así como la gestión eficiente de los recursos materiales. La figura 2.15, representa un esquema de las secuencias descritas por la Teoría de Elaboración.

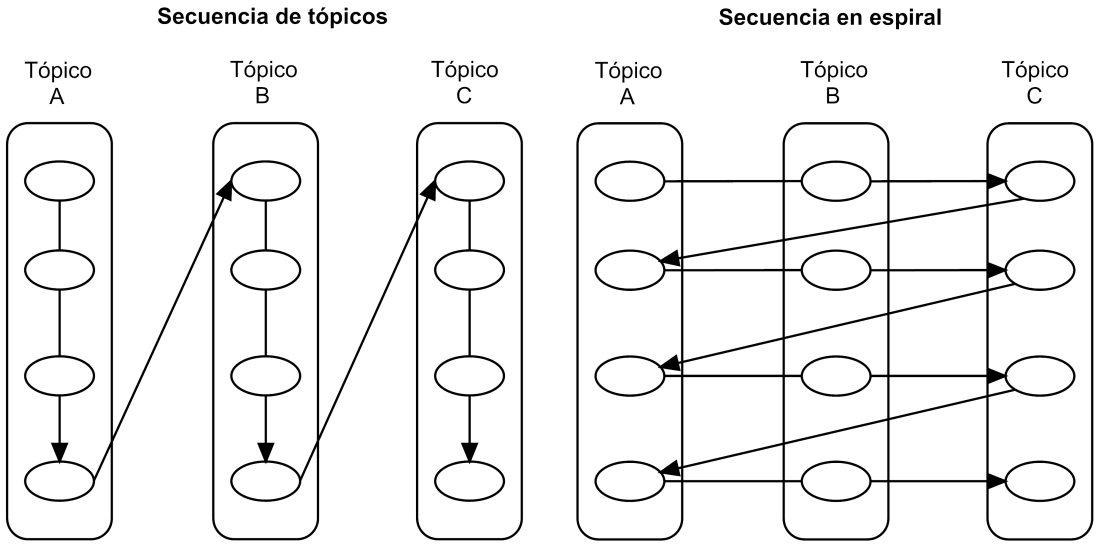

Figura 2.15: Secuencias descritas por la Teoría de Elaboración - Secuencia por tópico y espiral (Fuente: Reigeluth & Kim, 1993)

- Una última característica general que dicta la Teoría de Elaboración es proporcionar al estudiante la posibilidad de elegir el orden en que se realiza el aprendizaje de los contenidos.

La instrucción de la Teoría de Elaboración fue desarrollada para proveer una alternativa holística, para las partes de toda la secuenciación y la cobertura superficial de los contenidos que ha sido tan típica de la educación y formación en las últimas cinco a diez décadas [Reigeluth, 1999](15). Esto ha permitido el intento de ideas recientes acerca de la secuencia de instrucciones dentro de un marco coherente único. En consecuencia a lo anterior, esto tiene su origen a partir de la noción de que diferentes estrategias de secuencias esta basadas en diferentes tipos de interrelaciones dentro del contenido, y que diferentes interrelaciones son importantes para diferentes tipos de habilidad y/o pericia. De este modo, el tipo de secuencia que mayormente facilita el aprendizaje, variará

dependiendo del tipo de pericia desea desarrollar. En virtud de lo anterior, la Teoría de Elaboración hace una distinción entre *tarea especializada* y *dominio especializado.*

La tarea especializada, hace referencia a que el aprendiz llegará a convertirse en un experto en una tarea específica, como la gestión de proyectos, venta de productos o escribir un plan anual, etc. Se basa a partir de la observación de las complejas tareas cognitivas y psicomotoras, las que se realizan de manera diferente en distintas condiciones, y en donde cada una de estas condiciones define una versión diferente de tareas con distinto grado de complejidad. Varía de simple a complejo.

El dominio especializado, varía desde lo general a particular y este enfoque, el que permite también el diseño de secuencias holísticas que van desde lo simple a lo complejo. La guía de secuenciación para el dominio especializado tiene sus orígenes en el plan de estudio en espiral [Jerome Brunner, 1960](24), y la diferenciación progresiva [David Asubel, 1968](25); no obstante esta difiere en varios aspectos importantes de cada uno y también proporciona una mayor orientación en relación al diseño de tal secuencia. Una Secuencia de Elaboración comienza con lo más amplio, lo más inclusivo, las ideas más generales y gradualmente va progresando a más complejidad, ideas más precisas. Esto hace que una Secuencia de Elaboración sea ideal para descubrir un aprendizaje y otros enfoques hacia la construcción del conocimiento.

La Teoría de Elaboración estipula dos tipos principales de dominio especializado; el conceptual y el teórico. En su forma más simple, estos corresponden a conceptos y principios respectivamente; o en su forma más compleja, son estructuras de conocimiento conceptual (símil a un mapa conceptual), estrechamente relacionados entre sí.

A continuación se definen la Secuencia de Elaboración conceptual (en ingles, conceptual elaboration sequence), la Secuencia de Elaboración teórica (en ingles, theoretical elaboration sequence), y finalmente la simplificación de la secuencia de condiciones (en ingles, simplifying conditions method).

2.4.1.1 La Secuencia de Elaboración conceptual: The conceptual elaboration sequence.

La Secuencia de Elaboración conceptual está basada en varias observaciones. Lo principal, es que los conceptos son agrupaciones o clases de objetos, eventos o ideas que puede ser globales o generales hasta conceptos más acotados o menos inclusivos. El concepto más amplio proporciona lo que Ausubel denomina “cognitive scaffonlding”(en

español, andamiaje cognitivo), mientras que los conceptos más detallados los denominó "progressive differentiation"(en español, diferenciación progresiva) porque implica un proceso de hacer distinciones más sutiles progresivamente. El tipo de relaciones entre conceptos se basa particularmente en la inclusión de estos, con respecto a sus partes o tipos, constituyéndose lo que se denomina una estructura de conocimiento conceptual (ECC). La figura 2.16. se representa una ECC respecto a los tipos de música en donde las relaciones de inclusión de esta se conocen como relaciones superordinada, coordinada y relaciones subordinadas, es decir, de acuerdo a esta figura, la música clásica es subordinada de música, coordinada con música medieval y es superordinada a instrumental y vocal. Cabe destacar que en una ECC, no necesariamente, un concepto más bajo es más complejo o más difícil que uno superior, sino más bien suele ser más inclusivo respecto de su concepto superordinado.

La Secuencia de Elaboración conceptual se inicia mediante la enseñanza de los conceptos más amplio, más inclusivos y generales que el alumno no ha aprendido todavía, y procede a cada vez más estrecho, menos incluyente y conceptos más detallados, hasta alcanzar los niveles de detalles necesarios. Es un método recomendado cuando los objetivos de aprendizaje incluyen numerosos conceptos relacionados.

La definición de una ECC, es la base que enmarca las directrices de la presente investigación, puesto que, su representación será material de análisis que permitirá modelarla por medio de una semántica computacional. Su génesis se basa en la idea de poder generar una ECC a partir del cualquier dominio que manifiesta una ontología en particular, a través del tratamiento de las relaciones taxonómicas que esta presenta y donde solo son consideradas las relaciones de tipo y de partes que estas disponen, tal cual precisa la Teoría de Elaboración. El fin último de modelar una ECC, es marcar un precedente respecto de su implementación computacional, el cual pueda ser utilizado bajo cualquier marco pedagógico que lo requiera.

2.4.1.2 La Secuencia de Elaboración teórica: The theorical elaboration sequence.

La Secuencia de Elaboración teórica, es el segunda de las dos estrategias de secuenciación que actualmente precisa la Teoría de Elaboración para la construcción de un dominio especializado. Apunta a cursos que se centran en un conjunto de principios relacionados, los que son generalmente elaboraciones entre sí, como por ejemplo, un

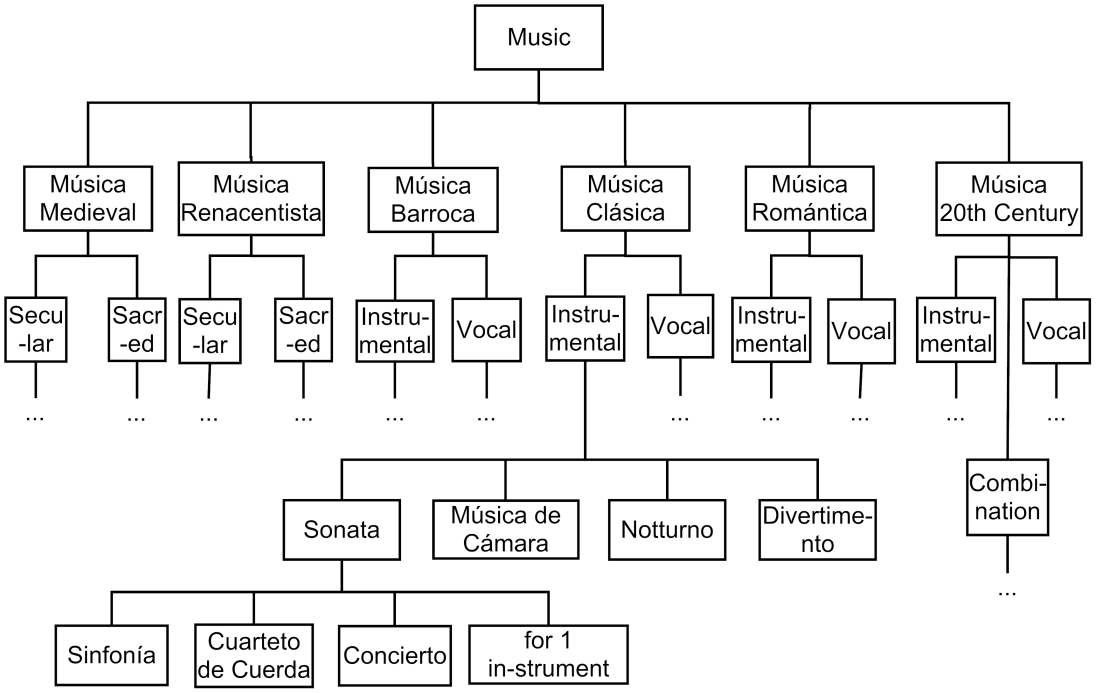

Figura 2.16: Tipos de Música - Ejemplo de una estructura conceptual (Fuente: Reigeluth, 1999)

curso de biología de la escuela que se centra en los principios de la genética, los ciclos vitales y las funciones corporales. Está basada en varias observaciones, siendo lo principal, que los principios son o bien una relación casual o relación entre procesos naturales. Ejemplo, la ley de la oferta y la demanda indica cómo los cambios en esta, pueden influir en su precio, y viceversa (como los cambios en su precio influyen en su oferta y demanda). Además, tanto los principios como conceptos, existen desde de una continuidad más amplia, más general e mas inclusiva a una más acotada, más específica y menos inclusiva. La forma de identificar todos los principios y sus relaciones de complejidad e inclusividad, es a través del análisis teórico, cuyo resultado es una estructura teórica, como lo demuestra la figura 2.17. Con respecto a su elaboración, esta puede ocurrir por medio del planteamiento de varias preguntas y sus diferentes tipos de respuestas, tales como:

- ¿Qué es lo que pasa? o ¿Qué puede causar esto?
- ¿Cuando esto causa un efecto?
- ¿De qué manera cambia las cosas?
- ¿Cómo ellos las cambian?
- ¿Cuánto ellos cambian?

La Secuencia de Elaboración teórica puede ser hecha en cualquiera de los formas, en tópicos o de manera espiral. Para la secuencia en tópicos, se podría ir todo el camino hasta una ramificación de la estructura teórica y ampliar gradualmente desde allí. Para una secuencia de espiral, se podía ir completamente a través de la fila superior y a continuación, a través de la siguiente fila hacia abajo, de manera sucesiva. La siguiente figura, representa una estructura teórica, por medio de un ejemplo en relación al principio de cuando los rayos de luz pasan de un medio a otro.

Codes: (A) ¿Qué más ocurre? (B) ¿Cuando? (B) ¿Porque? (C) ¿Por dónde? (D) ¿Cuánto?

0	Se comportan de forma inesperada.
1	Se doblan en la superficie.
2	Un objeto recto en ambos medios parece doblado en la superficie.
1.1	Los rayos se doblan, ellos reducen la velocidad en un medio más denso o aceleran en un medio menos denso (C).
1.2	Los rayos de doblan y cambian su distancia entre sí pero permanecen paralelos.(A)
1.3	Una parte de cada rayo se refleja fuera de la superficie mientras que el resto se refracta un nuevo medio (A).
2.1	La aparente posición y el tamaño del objeto suelen cambiar (A)
1.1.1	Si pasan a un medio más denso, los rayos de luz se curvan hacia la normal (B,D)
1.1.2	Cuanto mayor sea la diferencia de densidad óptica entre dos medios, los rayos de luz tendrán una mayor curvatura (D)
1.2.1	Cuando los rayos se doblan hacia la normal llegan a ser más separados (B,D)
1.2.2	Mientras más agudo sea el ángulo entre un rayo de luz y la superficie, el rayo tendrá una mayor curvatura (D)
1.3.1	Mientras más agudo sea el ángulo entre un rayo de luz y la superficie, el rayo tendrá un mayor reflejo y menos refracción (D)
1.3.2	Si el ángulo es igual a, o más agudo que, el ángulo crítico, todo el rayo de luz se refleja.
1.1.2.1	El índice de refracción $$(n) = c_i/c_r = (sini)/(sinr)$$ (D,E)
1.1.2.2	La relación entre el ángulo crítico y el índice de refracción es: $$sini_c = 1/n$$ (D,E)

Tabla 2.1: Ejemplo de una estructura teórica (Fuente: Reigeluth, 1999)

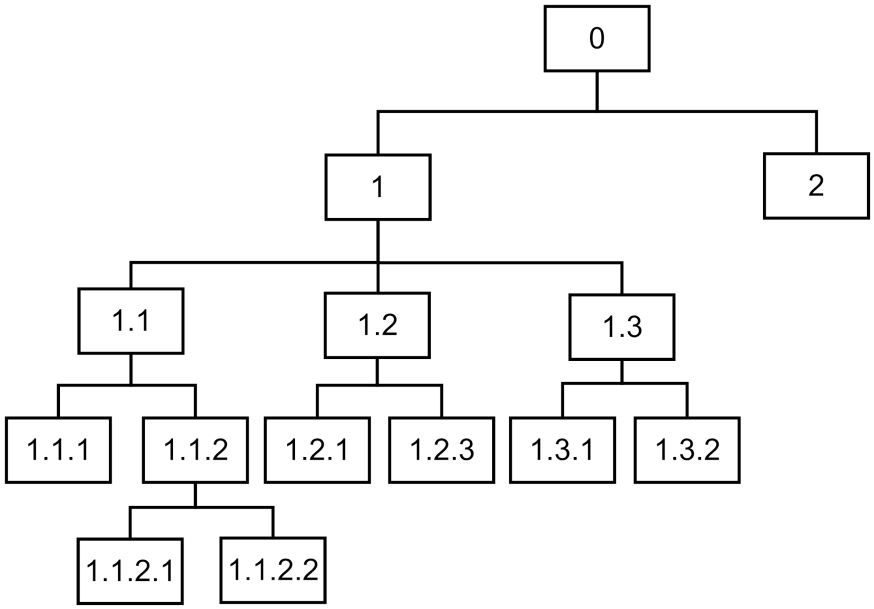

Figura 2.17: Estructura Teórica - Diagrama Conceptual de una estructura teórica

Basado en un análisis de ambos métodos, tanto secuencial como teórico, se determinó que ambos comparten los mismos métodos y su diferencia radica meramente en la situación de uso, específicamente en la naturaleza del objeto a aprender, es decir, si el objetivo de aprendizaje se centra en conceptos o en principios. En virtud de lo anterior, no existiendo en la especificación una distinción explícita para aprendizaje de conceptos o principios, hemos supuesto dentro del marco de modelado, que la representación ontológica del método Secuencia de Elaboración Teórica, es similar a los presentados por el método Secuencia de Elaboración Conceptual.

2.4.1.3 Método de simplificación de la secuencia de condiciones: The simplifying conditions method (SCM).

Dentro del contexto de construcción de tareas especializadas, the simplifying conditions method (en español, el método de simplicación de la secuencia de condiciones) es un nuevo enfoque (Aunque se ha practicado durante mucho tiempo intuitivamente) que ofrece una guía para analizar, seleccionar, y secuenciar lo que se "va a aprender"(contenido). El SCM ofrece una guía práctica para hacer diferentes tipos de secuencias de lo general a lo complejo, desde la secuencia jerárquica, una que es holística

que fragmentado [Reigheluth, 1999](15).

Dado que cualquier tarea compleja tiene unas condiciones bajo las cuales algunas son mucho más fáciles de realizar que otras. Una secuencia de SCM comienza con la versión más simple de la tarea (que resulta ser representativo de la tarea como un todo); luego se enseña versiones progresivamente más complejas de la tarea hasta que el nivel deseado de complejidad se alcanza, asegurándose de que el alumno conceptualice la relación de cada versión a las otras versiones. Cada versión de la tarea es un grupo o clase de información completa, en las representaciones del mundo real de la tarea.

Este proceso contrasta con el enfoque jerárquico para secuenciar, el cual enseña en primer lugar todos los prerrequisitos, pero no enseña por completo las tareas del mundo real hasta el final de la secuencia. La figura 2.18 muestra las diferencias entre el enfoque jerárquico y el enfoque de la SCM.

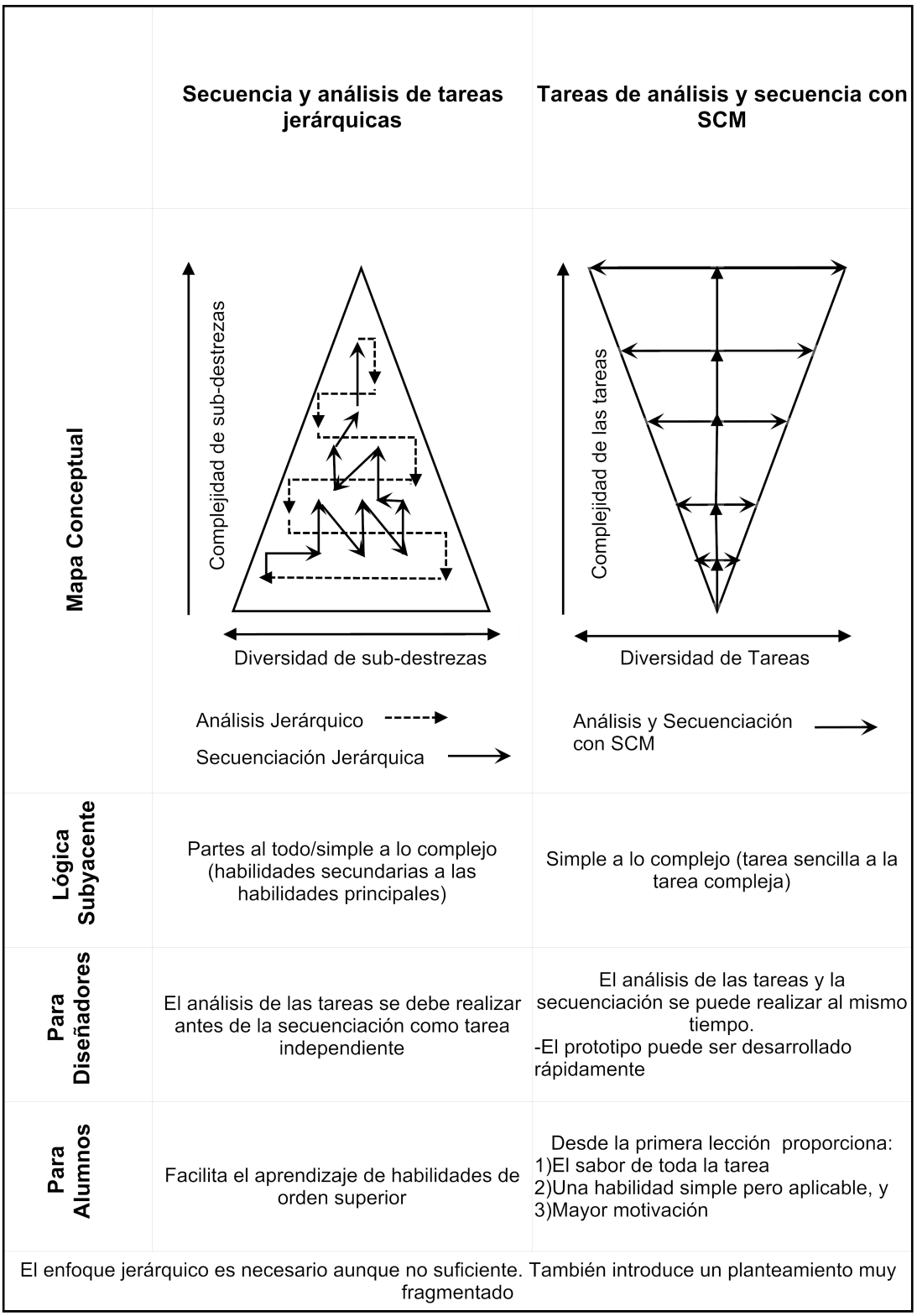

Figura 2.18: Diferencias entre enfoques - Diferencias entre enfoque jerárquico y el enfoque SCM (Fuente: Reigeluth, 1999)

2.5 El E-learning y tecnologías al servicio de la educación

El aumento en el uso de Internet y su mayor accesibilidad, han permitido que la educación en línea o E-learning sea una alternativa real a los procesos de aprendizaje actuales fundamentada en ser una opción que va más allá de las barreras geográficas, con flexibilidad temporal, sin olvidar la posibilidad de personalización del aprendizaje y la accesibilidad de los contenidos a usuarios con discapacidades. El E-learning fomenta el concepto de educación asistida por medio de un computador que utiliza tecnologías de información y comunicación conjuntamente con principios y métodos pedagógicos para facilitar el aprendizaje de estudiantes a distancia. De acuerdo a las orientaciones actuales, el E-learning utiliza principalmente los beneficios de la tecnología web, aunque también tecnologías provenientes de la Inteligencia Artificial e Ingeniería del Software.

En relación a los contenidos orientados al aprendizaje; estos se presentan a través de diversos medios, particularmente hipertextos y multimedia (imágenes, audio y videos) los cuales son encapsulados en los denominados Objetos de Aprendizaje (OA) y utilizados en diversos entornos E-learning. Como consecuencia, el gestionar cursos E-learning (incluyendo los OA adecuados para el propósito de aprendizaje), es realizado por múltiples plataformas denominadas Sistemas de Gestión de Aprendizaje (en inglés, LMS: Learning Management Systems) las que permiten no solo distribuir la información y recursos para el aprendizaje sino también preparar y aplicar variadas evaluaciones; todo desde un marco de aprendizaje tanto individual como colaborativo, este último, mediante foros, chats, almacenes de archivos, entre otras funcionalidades.

Diversas tecnologias han sido utilizadas en E-learning, ya sea con un enfoque hacia el alumno, o como apoyo al proceso instruccional de un profesor. En el primer caso el apoyo a los estudiantes se evidencia en los esfuerzos por mejorar las plataformas y los medios utilizados para alcanzar el aprendizaje estos, así como su adaptación a las características y necesidades de aprendizaje de los propios estudiantes [Guerrero, 2011](18). Un ejemplo de tecnologías que apoyan este enfoque son los Sistemas de Aprendizaje Inteligentes y Adaptativos o los Sistemas de Aprendizaje Colaborativos [Sánchez y Lama, 2007](19). Por otro lado el segundo enfoque, dirige la tecnología para asistir al profesor por medio del desarrollo y publicación de recursos para el aprendizaje en LMS, que ayudan a los estudiantes a resolver problemas, gestionando el proceso de aprendizaje y evaluando el rendimiento de los estudiantes, entre otras labores.

2.6 Recursos de Aprendizajes, Objetos de Aprendizaje y Metadatos

Una de las actividades realizadas por los profesores es la creación de recursos para el proceso de aprendizaje. Un recurso para el aprendizaje es cualquier medio creado con la intención de facilitar el proceso de aprendizaje; en este sentido, se trata de materiales que los educadores pueden usar y reutilizar en distintos ambientes de aprendizaje. Dentro del mismo entorno, los OA centran su atención en los contenidos educativos, proponiendo la descripción de estos recursos de aprendizajes a través de metadatos [Polsani, 2003](20), es decir, según el estándar LOM de IEEE [Learning Technologies Standards Committee, 2002](21) un OA es cualquier entidad digital o no digital, que puede ser usada para aprender, educar o enseñar. En virtud de esta amplia definición, textos impresos de estudio, documentos digitales, herramientas generadoras de test, presentaciones, videos, audio con fines educativos, entre otros podrían ser consideradas como OA. Del mismo modo, una definición más concreta es la de McGreal [McGreal, 2004](22) que los define como cualquier recurso digital reusable que tiene encapsulado una lección o ensamblado un grupo de lecciones en unidades, módulos, cursos e incluso programas, con el propósito de proporcionar un esquema modularizado basado en estándares, que permitan la flexibilidad, independencia de plataforma y el reúso de contenidos de aprendizaje. La estructura básica de un OA se compone de dos elementos: el contenido instruccional (un diagrama, un texto, una simulación, vídeo, etc.) y el Metadato [Millar, 2002](23); este último describe al recurso, indicando qué es, quién lo creó, cuál es su función, su objetivo, su duración entre otros aspectos, con el objetivo de recuperar y utilizar de estos recursos a través de colecciones en Repositorios de Objeto de Aprendizajes.

La conjugación de estos tópicos junto con un marco de conocimiento instruccional son la base para definir learning design (LD), entendiéndose este, múltiples formas de diseñar experiencias de aprendizaje orientados a estudiantes, es decir, una secuencia de tipos de actividades e interacciones que permiten definir y diseñar una instrucción en particular. Desde un punto de vista informático, el reto está en diseñar herramientas que no se enfoquen únicamente en la creación de contenidos y recursos de aprendizajes, sino que además modelen un LD. Además, un reto adicional es garantizar que los recursos y elementos definidos puedan utilizarse, compartirse e intercambiarse entre

diferentes lecciones o plataformas, y con ello se reduzcan el tiempo y recursos empleados. Este reto, es precisamente el objetivo que tiene los denominados lenguajes de modelado educativos (en ingles, EML: Educational Modeling Languages), ejemplos de estos son PALO, OUNL-EML y su derivado IMS-LD, este último, analizado en profundidad en el próximo capítulo debido a su aplicabilidad en la presente investigación.

2.7 Especificación IMS-LD, principios básicos de contextualización

El estándar IMS-LD no tiene injerencia en el proceso de generación de una ECC, sin embargo, su uso se limita a verificar la conformidad de los tópicos de aprendizajes confeccionados por medio de este estándar con la Teoría de Elaboración (que precisa de una ECC).

En 2003, el IMS Global Learning Consortium Inc.[I] publicó el IMS -Learning Design; esta especificación abierta es una forma flexible de representar y codificar escenarios de aprendizajes para múltiples alumnos, manera formal, interoperable y en una semántica entendible por un computador. Puede facilitar su comprensión, pensar en ella como una forma de crear planes de lecciones interoperables que pueden ser leídos por una aplicación denominada *player*. El *player* puede encargarse de coordinar a los alumnos, profesores, recursos de aprendizaje y actividades a medida que el proceso de aprendizaje evoluciona [Daniel Burgos, 2008](17).El IMS-LD permite modelar procesos de aprendizaje y de comunicación interactiva entre los actores participantes. Define quién, cuándo, cómo y para qué se utilizan recursos, servicios y actividades de aprendizaje [CVidal, 2011](27). Además, permite el modelado de escenarios de aprendizaje y la aplicación de diversos enfoques pedagógicos. Es pertinente señalar que IMS-LD no ofrece ningún modelo o modelos pedagógicos concretos, más bien pretende ser utilizado para definir múltiples escenarios educativos. En virtud de esto, se suele referenciar como un meta-modelo pedagógico [Daniel Burgos, 2008](17).

El concepto central en que se basan el IMS-LD, es el de Unidad de Aprendizaje (en inglés Unit of Learning, UoL). Cada UoL considera recursos (resources), objetivos

[I] Para mayor información sobre el Consortium IMS, visitar el sitio web oficial: `http://www.imsglobal.org/learningdesign/`

de aprendizaje (learning objectives), prerrequisitos (prerequisites), componentes como roles, actividades (activities), ambientes (environments) compuestos por Objetos de Aprendizaje (Learning Object) y servicios (services), métodos (methods), representaciones (play), actos (acts) y partes-roles (role-parts).A raíz de lo anterior la especificación IMS-LD se basa en la metáfora de la obra de teatro para realizar el diseño. La idea fundamental se resume en la figura 2.19. En dicha figura puede verse cómo un método de aprendizaje (method) se encuentra dividido en obras (play), que contienen diversos actos (acts) donde diferentes actores desempeñan papeles (role) sobre un escenario (roleparts) llevando a cabo actividades (activities) en un entorno (environments) específico. Estas actividades pueden clasificarse en tres tipos: (1) actividades de aprendizaje (learningactivities) que llevan al alumno a obtener conocimiento; (2) actividades de soporte (supportactivities) que no contribuyen al aprendizaje en sí mismo, pero que son necesarias para que las actividades de aprendizaje puedan desempeñarse con éxito; (3) estructuras de actividades (structureactivities) que permiten estructurar actividades de aprendizaje, soporte u otras estructuras de actividades, estableciendo orden de secuenciación o selección entre las mismas. Todas estas actividades pueden ser desempeñadas en entornos (environments) específicos donde se dispone de objetos de aprendizaje (learning objects) y servicios (services) como chats, foros y monitores que los actores pueden usar durante el desarrollo de sus actividades.

2.7.1 Niveles de Especificación en la IMS-LD

La especificación IMS Learning Design plantea un lenguaje potente para el modelamiento de los elementos estudiados, aunque es considerado por la comunidad académica como excesivamente complejo de emplear y, sobre todo, de implementar en un LMS. Para facilitar su adopción progresiva, IMS-LD propone tres niveles, a los que denomina nivel A, B y C, respectivamente [I] . En virtud de lo anterior, el primer nivel es bastante práctico de implementar, permitiendo crear Diseños Instruccionales sencillos. No obstante, un LMS que implemente solamente el nivel A no puede considerarse completamente compatible con la especificación IMS Learning Design, no obstante puede considerarse compatible con el Nivel A de la especificación. Los Niveles B y C añaden funcionalidad y potencia, construyendo siempre sobre el nivel anterior. Esto permite

[I] Para mayor información en relación a los aportes que precisan los niveles de IMS-LD, visitar el sitio web: `http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html`

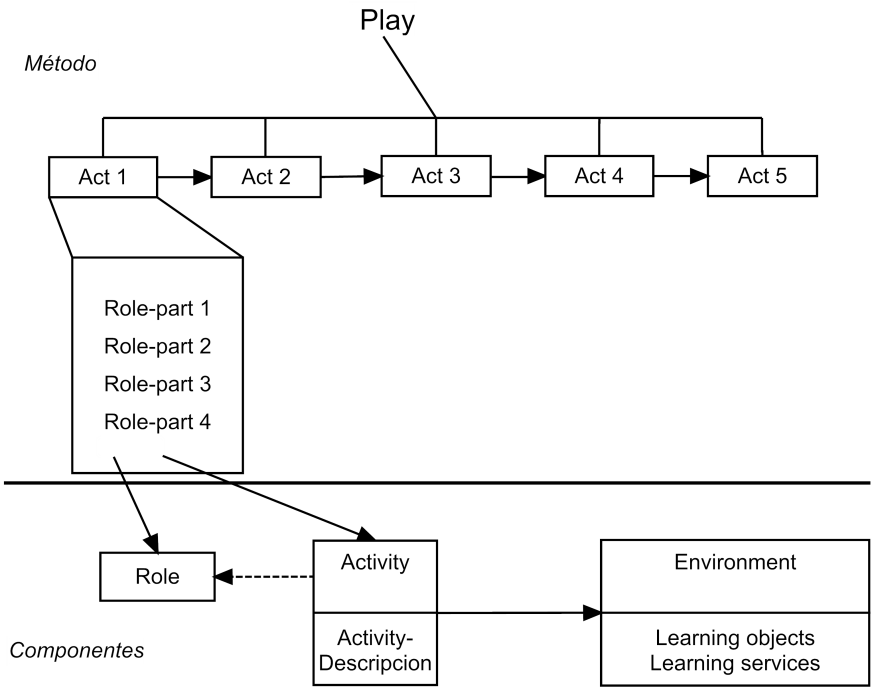

Figura 2.19: Método IMS-LD - Diagrama de un método IMS-LD (Fuente: Olivier, 2003)

que las organizaciones adopten IMS Learning Design incrementalmente y, si las necesidades de la organización no requieren de la adopción completa de la especificación, se puede optar por una adopción parcial llegando sólo al nivel que fuese necesario.

La siguiente figura 2.20, describe la funcionalidad especificada en cada uno de los niveles de IMS Learning Design.

- Nivel A (contiene al lenguaje núcleo del Learning Design).
- Nivel B (añade propiedades y condiciones al nivel A).
- Nivel C (añade notificaciones a los niveles A y B).

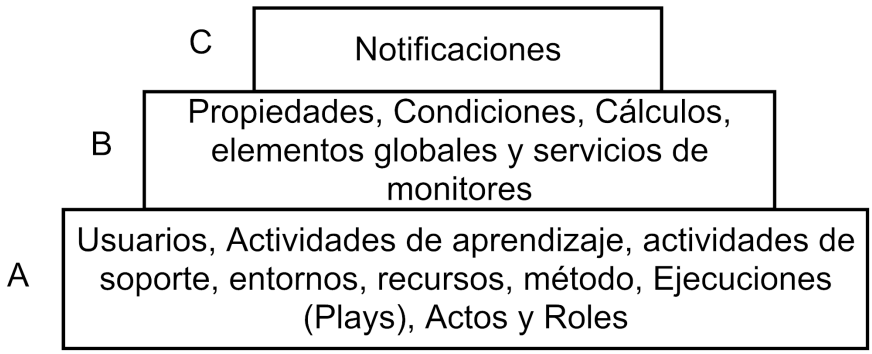

Figura 2.20: Niveles IMS-LD - Representación de niveles de la especificación IMS-LD (Fuente: Burgos, 2008))

Es destacable mencionar que la mayoría de las herramientas, solamente se basan en el nivel A. Debido a los siguientes factores:

- Es el nivel más sencillo de implementar tanto para los diseñadores de aprendizaje como para los desarrolladores de software.
- El método pedagógico mayoritario que utilizan los profesores se ajusta perfectamente al nivel A.
- No existen en la actualidad herramientas que permitan un diseño de alto nivel que abarquen los elementos de los niveles B y C, por lo que los creadores no técnicos no pueden hacer uso de él.

A continuación, se resumen las funcionalidades de IMS-Learning Design partir de los niveles implementados a través de la tabla 2.2.

Funcionalidad
Nivel A
Es posible crear unidades de aprendizaje en las que se define un proceso colaborativo.
Comprende la definición de usuarios, actividades de aprendizaje, actividades de soporte, entornos, recursos, método, ejecuciones (plays), actos, roles y la coordinación entre todos ellos.
Además los usuarios podrán utilizar recursos externos, enlaces web y diversos servicios (foros, chat..).
Nivel B
Añade los componentes de propiedades, condiciones, servicios de monitorización y elementos globales para gestión de la especificación desde ficheros externos a la misma.
Las propiedades almacenan información sobre personas (preferencias, resultados, información personal...), sobre un rol o sobre el diseño de aprendizaje en sí mismo.
El estado de las propiedades y de las condiciones puede modificar el flujo de trabajo e influir en el desarrollo de la unidad.
Permite esconder y mostrar elementos, condicionar el flujo de aprendizaje, almacenar datos del usuario y la instancia, bien a nivel local y personal, bien a nivel global y compartido.
Nivel C
Introduce un mecanismo de notificación o envío de mensajes automático como respuesta a eventos que se originan en el proceso de aprendizaje.

Tabla 2.2: Funcionalidad - Funcionalidades de IMS-Learning Design partir de los niveles implementados.

2.7.2 Herramientas de IMS-LD

Las herramientas orientadas a IMS-LD tienen la finalidad de entregar soporte a la creación de un LD, expresado en una unidad de aprendizaje, a través de un ambiente amigable, intuitivo y simple con el propósito de ayudar al usuario que no posea conocimientos en IMS-LD, ocultando los metadatos propio del estándar y sugiriendo un ambiente complaciente. Dependiendo de la intencionalidad de su desarrollo, la herramienta proporcionará un soporte para algún nivel de expresividad, destacándose en este campo la herramienta ReCourse [I] y el nivel A de soporte.

Las herramientas IMS-LD se organizan en tres categorías:

1. **Herramientas de Autoría:** Este tipo de herramientas apoyan a los diseñadores en las tareas de desarrollar una UoL. Estas herramientas difieren en los niveles de apoyo a la especificación. Por ejemplo: Recourse, Reload [II] y LAMS [III] entre las más relevantes.

2. **Motores de Ejecución:** Este tipo de herramientas funcionan como servidor en donde los UoL son ejecutado por los usuarios, quienes asumen algunos de los roles definidos en el LD interactuando mediante la realización de actividades. Por ejemplo: Edubox y CopperCore [IV] .

3. **Repositorios:** Este tipo de herramientas se orienta a proporcionar servicios útiles a organizaciones o comunidades que compartan y reutilicen resultados de sus diseños de aprendizaje. [V] .

[I] Para mayor detalle en relación a la herramienta ReCourse, visitar la pagina oficial de esta: `http://tencompetence-project.bolton.ac.uk/ldauthor/`

[II] Para mayor información sobre la herramienta de Autoría Reload, visitar: `http://www.reload.ac.uk/`

[III] Para Mayor detalle sobre LAMS, visitar el sitio web oficial: `http://www.lamsinternational.com/`

[IV] Para Mayor especificación del motor de ejecución CopperCore, visitar: `http://coppercore.sourceforge.net/`

[V] Los principales repositorios de diseños de aprendizajes pueden ser visitados en los sitios web: `http://dspace.learningnetworks.org/` y `http://www.merlot.org/`

2.8 Conclusiones del estado del arte

Durante este capítulo se presentaron los temas más relevantes que sustentan el objeto central de la presente investigación. En relación a cada uno de estos tópicos, se destacaron aspectos, cuyos aportes, permitirán modelar e implementar la construcción de una ECC con soporte computacional, que la Teoría de Elaboración precisa.

En primera instancia se destaca la importancia que tienen las ontologías al servicio de la representación del conocimiento y los beneficios de la aplicabilidad de estas en ambiente E-learning. Se detalla su composición y características, cimentando las directrices para el tratamiento de sus relaciones taxonómicas, por medio del análisis y estudios de lenguajes ontológicos pertinentes. Por otra parte, se profundiza en el Diseño Instruccional y las teorías asociadas a este, particularmente sobre la Teoría de Elaboración, principal tópico de la presente investigación y las características que este precisa; una estructura de conocimiento conceptual, que dicta la literatura y del cual se vislumbra al término de esta tesis, una representación con soporte semántico computacional. Para tal efecto, las ontologías de dominio cumplen un rol vital, a la hora de representar un ECC, puesto que, su tratamiento modelado a partir de relaciones de tipos y de partes además de la taxonomía de conceptos del dominio estructurados de lo más general a lo particular, cumplen con los criterios que precisa una ECC.

En virtud de lo anterior, basados en dar un marco de aplicabilidad pedagógica al modelado de una ECC, se presentaron tópicos relacionados al área , contemplando en primera instancia, la proliferación del E-learning y las tecnologías al servicio del aprendizaje, así como los principales conceptos que involucran esta materia; recursos de aprendizaje, objetos de aprendizaje y metadatos. A posteriori, se enfatizó en la especificación IMS-LD, la cual define un lenguaje de modelado de actividades de aprendizaje. Esta especificación mejora la flexibilidad y la expresividad, permitiendo representar una gama más amplia de enfoques pedagógicos. El tratamiento de esta especificación en la investigación, está sujeta a la vinculación de la ECC dentro de un entorno pedagógico, específicamente dentro de la herramienta IMS-LD ReCourse y la herramienta que analiza la conformidad de un LD respecto a teorías de Diseño Instruccional, Terpsicore, quien no cuenta con un soporte computacional para modelar la Teoría de Elaboración y su ECC que precisa.

Cabe destacar que si bien es posible modelar una ECC a partir del estudio de los diversos tópicos vistos; Reigeluth, sugiere dentro de los principios de la teoría de elaboración, mantener la atención y motivación del aprendiz [Reigheluth, 1999](15). Este hecho se ha obviado a raíz de la utopía que esto conlleva producto de la imposibilidad de modelar estos ámbitos desde una óptica computacional hoy en día.

2. ESTADO DEL ARTE

3

Modelado para la representación de una Estructura de Conocimiento Conceptual

3.1 Introducción

Esta sección comienza presentando el enfoque actual para la representación un ECC; vislumbrando una carencia en la automatización del proceso, basado en una técnica manual e improvisada por medio de la herramienta para la confección de ontologías, Protégé[I]. Luego, se presenta una propuesta formal para dar soporte automatizado a la creación de una ECC a través del modelado de ontologías de dominio representada solo por relaciones de tipo y de partes, *part_of*, *is_a*, respectivamente y que la Teoría de Elaboración precisa. Para efecto de comprensión del modelado, ambas relaciones se presentan en este capítulo de manera independiente, basadas en la idea de modelar en primera instancia su relación de tipo y asentada en esta su relación de partes.

3.2 Representación actual de una ECC por medio de TGS

De acuerdo a la literatura, una ECC se limita solo a los principios que Reigeluth precisa dentro de la Teoría de Elaboración, por lo que fundado en esto, los intentos de modelarla han carecido de un soporte computacional, siendo el trabajo [Cvidal, 2011](27) el que

[I]Para mayor información respecto a la herramienta ontológica Protégé visitar: `http://protege.stanford.edu/`

se aproxima a esto, modelando diversas TDI entre ellas la Teoría Elaborativa; sin embargo, no posee un soporte automatizado de esta, debido a la falta de una ECC modelada computacionalmente. Como consecuencia,en este trabajo, para dar soporte a esta teoría, se realiza previamente un proceso manual e improvisado denominado Esquema General de Tópicos (en inglés TGS: Topics General Schema), el cual permite generar una ECC a partir de un dominio de aprendizaje plasmado en una ontología. Tal proceso se realiza mediante la herramienta para la confección ontológica, Protégé, en la cual, se realiza un proceso que permita tratar la ontología de dominio taxonómicamente y adaptarla de acuerdo a los principios que Reigeluth dicta: "usando un enfoque de lo general a lo particular"y respecto a las relaciones: "el tipo de relación se basa en la inclusión de conceptos, con respecto a sus tipos y partes".

El proceso de la TGS requiere de amplios conocimientos en materia de confección ontológica sobre dicha herramienta. En primera instancia, se debe considerar una ontología pertinente al Learning Design que se requiere analizar según la Teoría de Elaboración. La figura 3.1, esquematiza el proceso la elección de una ontología pertinente a un LD específico. De acuerdo a esto, determinado el LD (A), se realiza la búsqueda Web de una ontología de dominio en conformidad a este (B).

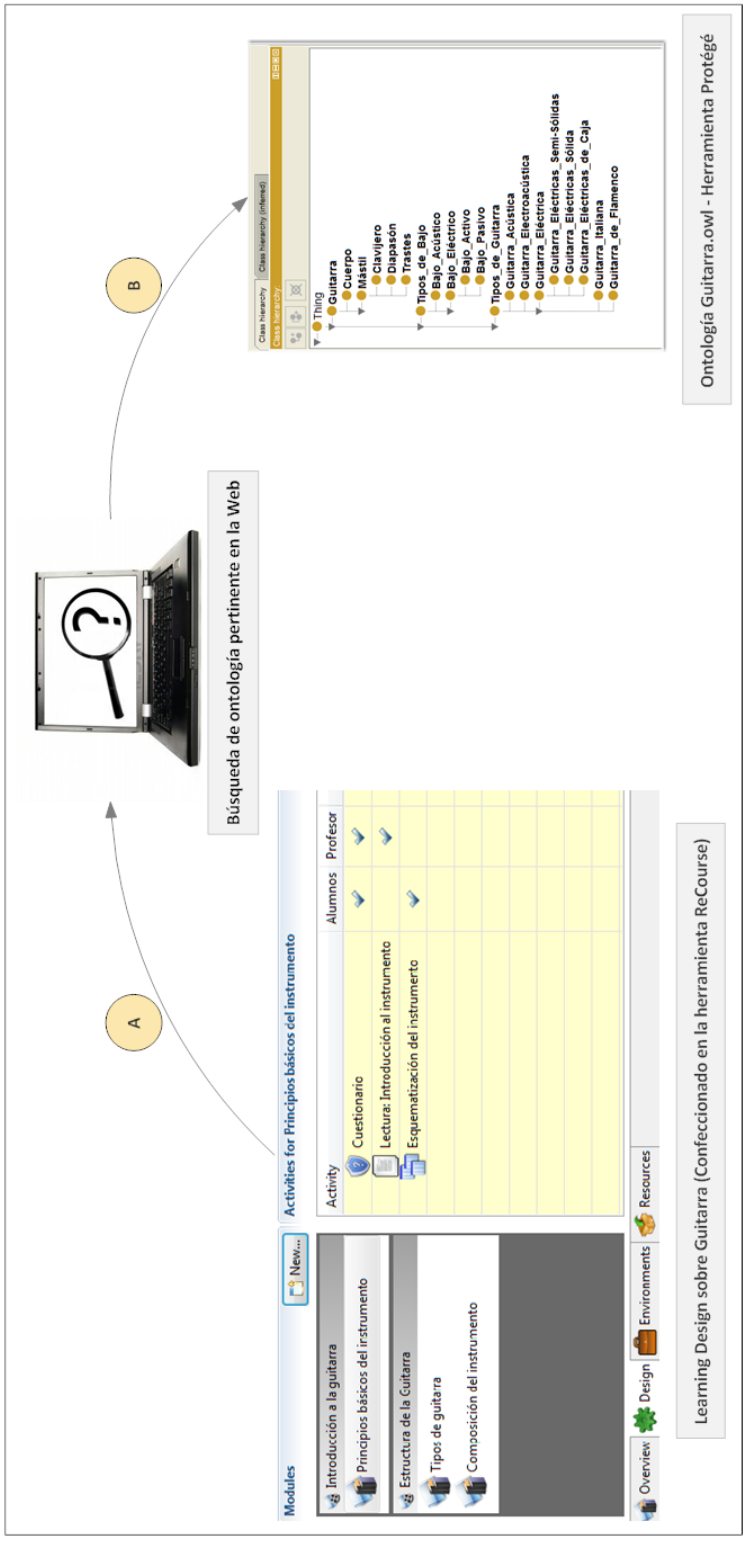

Figura 3.1: Relacion LD y ontología - Representación de búsqueda de ontología en conformidad a un LD.

3. MODELADO PARA LA REPRESENTACIÓN DE UNA ESTRUCTURA DE CONOCIMIENTO CONCEPTUAL

Una vez seleccionada la ontología de dominio, se determina una búsqueda manual o producto de lo extensa de alguna de estas, por medio de la ayuda de Protégé, se realiza la búsqueda del tópico que más se adapte a la conformidad del LD a instruir; siempre y cuando sea de tamaño cuantificable computacionalmente, puesto que muchas superan el límite de procesamiento de una maquina promedio. La figura 3.2, representa, la elección del tópico más apropiado, descartando las demás clases. En este ejemplo, basado en el LD acerca de Guitarra, se determina en la ontología seleccionada, que la clase Guitarra se aproxima más al diseño de aprendizaje propuesto.

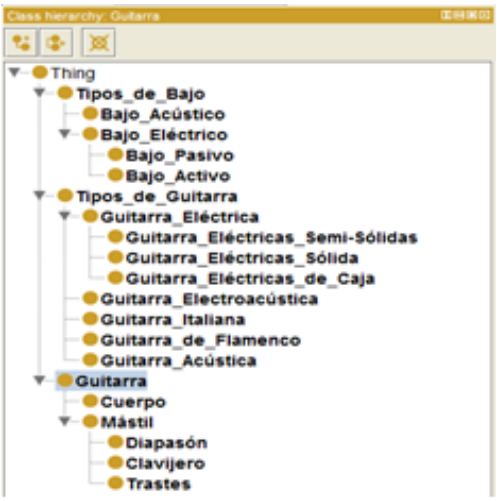

Figura 3.2: Selección del Tópico - Selección del tópico más apropiado a la conformidad del LD.

En el mejor de los casos, encontrado el tópico adecuado al LD en la ontología, se debe analizar su estructura taxonómica, identificando en primera instancia aquellas relaciones de tipo, *is_a*, y la distancia conceptual (profundidad de tópicos ontológicos) que comprenden a los temas que el LD precisa. Aquellas clases ontológicas que no comprenden los tópicos del diseño de aprendizaje, se descartan. Una segunda etapa del proceso de TGS, es identificar aquellas relaciones de composición entre conceptos de la ontología, analizando su dominio y rango entre las clases, de manera que estén contenidas dentro de las relaciones de tipo, previamente seleccionadas, siempre dentro del marco del tópico seleccionado como adecuado para la conformidad del LD.

Finalizada la selección de relaciones de tipo y de partes, se realiza un proceso manual e improvisado, que permita emular una ontología de dominio compuesta solo por las

clases y relaciones obtenidas, fundamentadas en el tópico seleccionado que manifiesta conformidad al LD. Para tal efecto, se utiliza una ontología especial denominada ret.owl que contiene clases que modelan distintas especificaciones, estándares, como IMS-LD y LOM respectivamente, además de TDI, entre otros tópicos. Dentro de ret.owl, la clase reservada *tgs: KnowledgeItem* cumple un rol vital, puesto que, en ella se debe crear una subclase que represente al tópico seleccionado y sus relaciones, es decir, al tópico seleccionado como adecuado para la conformidad del LD. Comúnmente, se utiliza el prefijo *knowledgeitem* seguido de un guion y el nombre del tópico. La siguiente figura 3.3, representa la creación de la subclase, dentro de la clase *tgs: KnowledgeItem.*

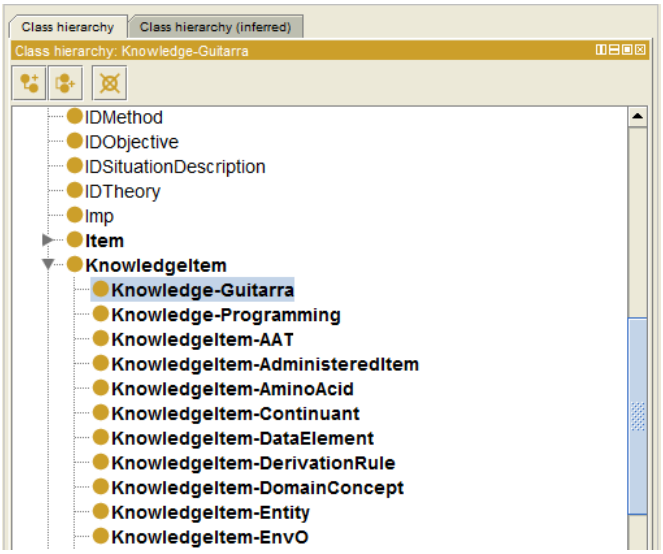

Figura 3.3: Instanciación del Tópico - Creación de la subclase dentro de tgs: KnowledgeItem en la ontología ret.owl .

La creación de esta subclase, tiene por objeto representar aquella pseudo-ontología compuesta solo por las relaciones de tipo y de partes seleccionadas. Para tal efecto, representar dichas relaciones, requiere de un proceso de instanciación. En este proceso cada clase se representa como un individuo de la subclase creada, mientras que sus relaciones, son sometidas a un proceso de correspondencia, en donde, de acuerdo al tipo de relación tanto *is_a* como *part_of* se clasifican, en *concept-hasKind* y *concept-hasPart*, respectivamente, por medio de la etiqueta tgs, que la herramienta Protégé sostiene. La tabla 3.1 sostiene dicha conversión.

Ontología de Dominio	Topics General Schema
is-a	tgs:concept-hasKind
part-of	tgs:concept-hasPart

Tabla 3.1: Conversión las relaciones de tipo y partes dentro en el Esquema General de Tópicos.

El proceso en sí de instanciación de clases, se realiza por medio del panel *individuals*, en este, cada clase deberá agregarse como individuo de la subclase creada en *KnowledgeItem*, por medio de la opción *Add Individual*. La figura 3.4, representa dicho proceso de instanciación.

Figura 3.4: Panel Individuals - Representación del proceso de instanciación de clases.

Una vez terminada la instanciación de cada clase, se debe establecer sus relaciones, para esto se debe realizar por medio del *Property assertions* en la opción *Object Property assertions*. Esta opción permite la relación entre individuos. Mientras que *Data property assertions* permite relacionar individuos con valores de datos. La figura 3.5, representa las funcionalidades descritas.

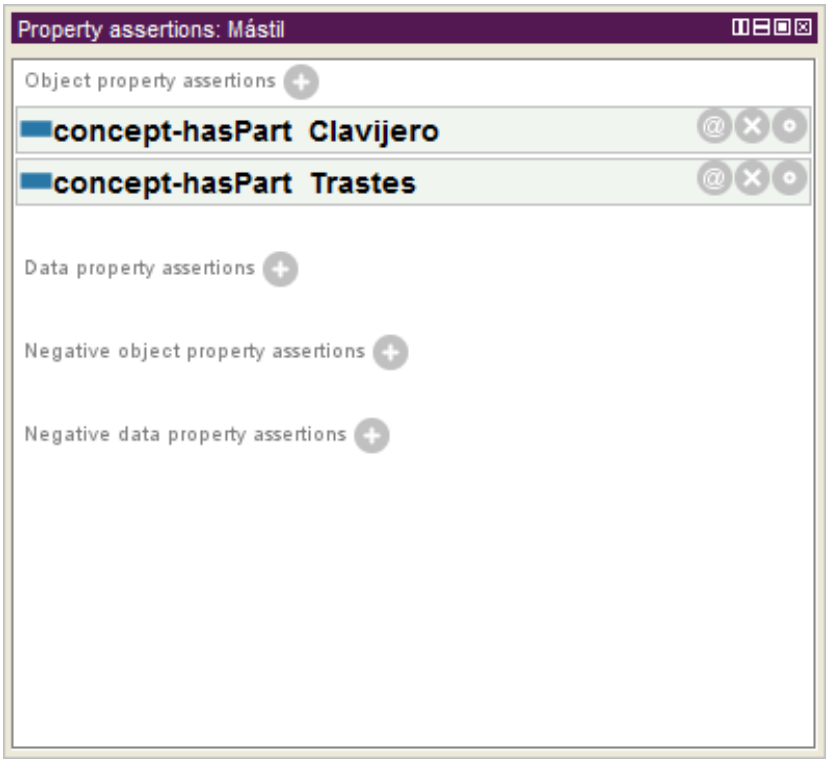

Figura 3.5: Property Assertions - Proceso de instanciación entre individuos o valores de datos.

Una vez seleccionada la opción *Object property assertions (+)*, esta permite vincular una relación con un individuo seleccionado. Estas relaciones *concept-hasKind* y *concept-hasPart* se encuentran dentro de la categoría *concept-includes*, en la ontología ret.owl. La figura 3.6, representa el proceso de relación sobre individuos.

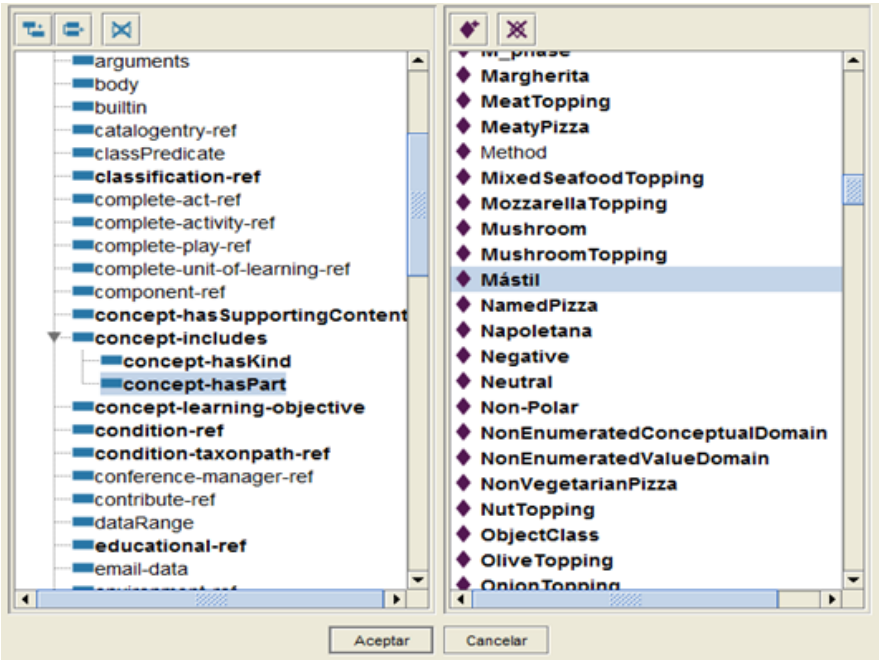

Figura 3.6: Object property assertions - Proceso de vinculación entre individuos y sus relaciones.

Una vez terminado el proceso de instanciación; por medio de la opción gráfica de Protégé, *OntoGraf*, se puede visualizar el resultado de la ontología o la Estructura de Conocimiento Conceptual creada a partir del proceso de TGS, cimentado en la elección de un concepto ontológico acorde a un LD y el análisis taxonómico de sus relaciones de tipo y de composición. La siguiente figura 3.7 representa, en primera instancia la ECC gráficamente por medio de la función *OntoGraf*, mientras que la figura siguiente, representa el esquema general del proceso TGS.

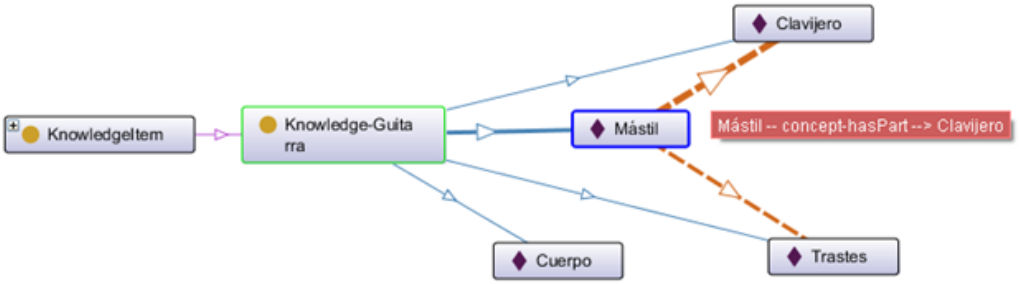

Figura 3.7: OntoGraf - Representa la Estructura de Conocimiento Conceptual de forma gráfica.

3. MODELADO PARA LA REPRESENTACIÓN DE UNA ESTRUCTURA DE CONOCIMIENTO CONCEPTUAL

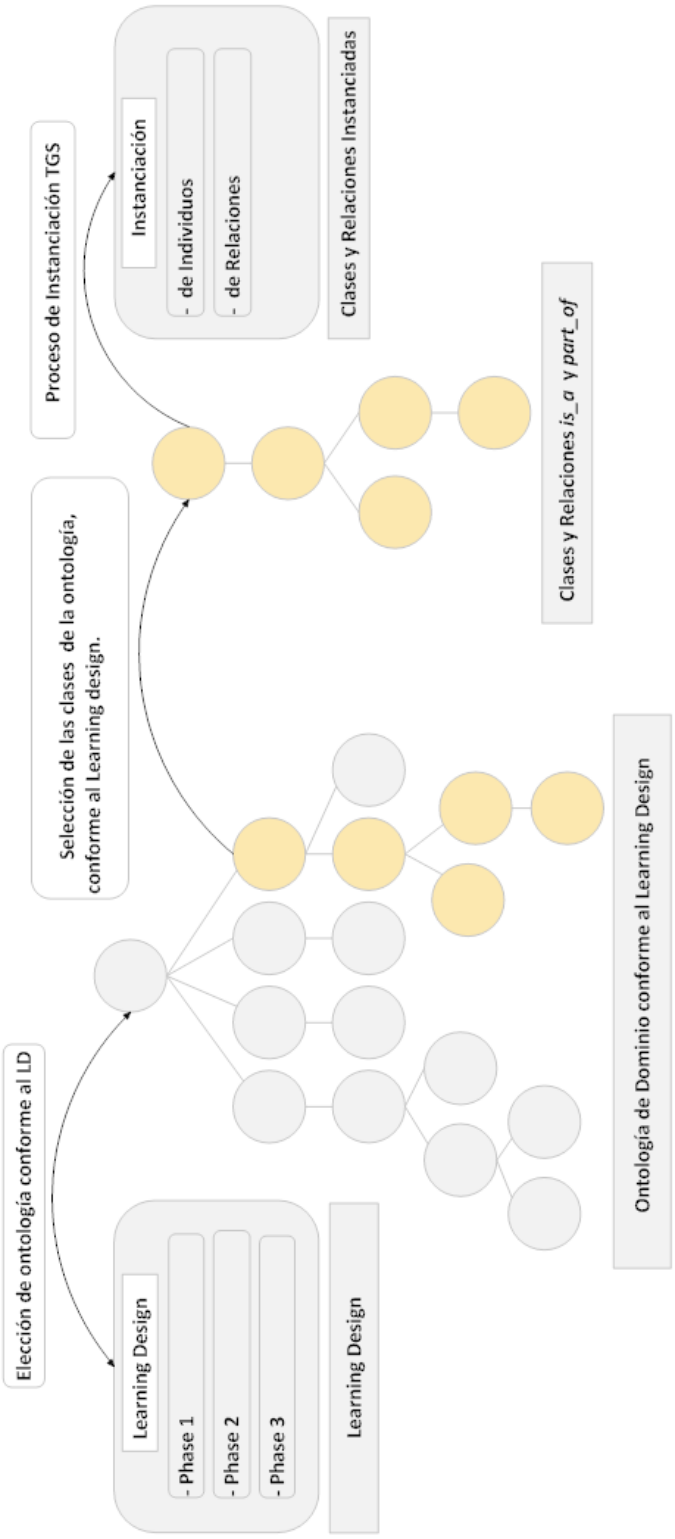

Figura 3.8: Esquema General Tópicos - Modelo General del proceso Esquema General de Tópicos.

3.3 Propuesta para el diseño computacional de la ECC

A partir de los estudios previos, las siguientes secciones presentan la base para la modelación automatizada de que soporte a una ECC a partir de una ontología de dominio existente. En este sentido, se propone el tratamiento de ontologías, a través de la recomposición de estas, solo por relaciones de tipo y de partes, y bajo el cimiento de una estructura jerárquica ontológica que va de lo más general a lo particular. Para tal efecto, se propone en primera instancia, modelar las relaciones de tipo, para luego asentado en estas, tratar las relaciones de partes. Por ende, se considera fundamental en la presente investigación, abarcarlas de manera autónomas de manera tal que sea posible comprender su estructura, facilitando el tratamiento y comprensión de estas.

Cabe señalar que en la presente investigación, dichas relaciones se entenderán como *is_a* y *part_of*, respectivamente, siendo estas su representante sintáctica que las identificara a lo largo del estudio, aunque no representan bajo ningún punto una estandarizacón formal, puesto que su identificación difiere entre diversas ontologías de dominio, siendo la principal criticidad a la construcción de estas.

Con respecto al tratamiento tanto en el modelado y la posterior implementación de la ECC, la investigación utiliza los lenguajes y herramientas ontológicas basadas en los resultados del análisis de artículos referente al desarrollo de ontologías dentro del campo de E-learning [CVidal, 2011](27). En este, se detalla que el lenguaje de representación ontológico más utilizado es OWL, con un 47% de preferencias e incrementa a 59% si se considera a los artículos que utilizan OWL por si solo o como complemento a un segundo lenguaje. La tabla 3.2, representa los lenguajes más utilizados, según el estudio.

Representation Language	Quantity
OWL	11
RDF	4
DAML+OIL	1
CycL+OWL	1
KIF	1
HOZO+OWL	1
OWL+SWRL	1
WSML	1
XTM	1
UML	1
OIL	1

Tabla 3.2: Lenguajes de representación ontológicos (Fuente: Handbook of Metadata, Semantics and Ontologies, SICILIA, M-A (por editar)).

Además, el 44% de los artículos informan el uso de Protégé, como la herramienta de diseño ontológico más usada. La tabla 3.3, representa las herramientas de diseño ontológico más usadas.

Tool used	Quantity
Protégé	10
N/R	10
OilEd	1
Ontololingua	1
Topic Maps	1

Tabla 3.3: Herramientas de diseño ontológico(Fuente: Handbook of Metadata, Semantics and Ontologies, SICILIA, M-A (por editar)).

3.3.1 Relación taxonómica *is_a*

Las relaciones *is_a*, parte de la taxonomía ontológica, no están dentro de los lenguajes ontológicos de manera explícita, por lo que a priori a su modelado, se percibe fundamental conceptualizarla a partir del concepto de taxonomía, entendida esta como: una jerarquía semántica en donde entidades de información son relacionadas a través de clases y subclases; donde la primera de ellas es semánticamente más fuerte que la segunda; sin embargo las taxonomías de primer orden carecen de dificultad para expresar riqueza en el significado, en contraposición a las de segundo orden, quienes utilizan la noción de propiedades o atributos para diferenciar una clase-hija (subclase), de una clase-padre (clase) denominada también superclase. Como consecuencia, para poder modelar esta relación, se requiere estructurar la nueva ontología (ECC) taxonómicamente dentro de un marco conceptual basado en clases y subclases, las que sí presentan sustento en lenguajes ontológicos capaces de representarlas formalmente. Para tal efecto, se hace fundamental conocer la sintaxis referente al lenguaje ontológico determinado en la presente investigación: OWL.

Para representar la relación en este lenguaje, es necesario poseer conocimientos sobre las características del esquema RDF, principalmente en cómo se representan las clases y las subclases. A nivel sintáctico una clase se describe mediante la etiqueta *owl:Class* y se organizan en una jerarquía de especialización mediante la etiqueta *rdfs:subClassOf*.

Cabe destacar que en toda ontología existen clases generales o abstractas como *Thing*, que es una clase,de la cual heredan todos los individuos y superclase de todas las clases de OWL; mientras que *Nothing*, es la clase que no tiene instancias y es una subclase de todas las clases de OWL.

De este modo, la taxonomía de una ontología compuesta solo por (*is_a*), se plasma en el siguiente ejemplo; donde se desea modelar una estructura que indique que tanto los conceptos Guitarra Acústica como Guitarra Eléctrica son un tipo (*is_a*) de Guitarra. O desde otro punto de vista, indicar que Guitarra es una superclase de las clases Guitarra Acústica y Guitarra Eléctrica o estas últimas son subclases de la clase Guitarra. La figura 3.9, representa este ejemplo.

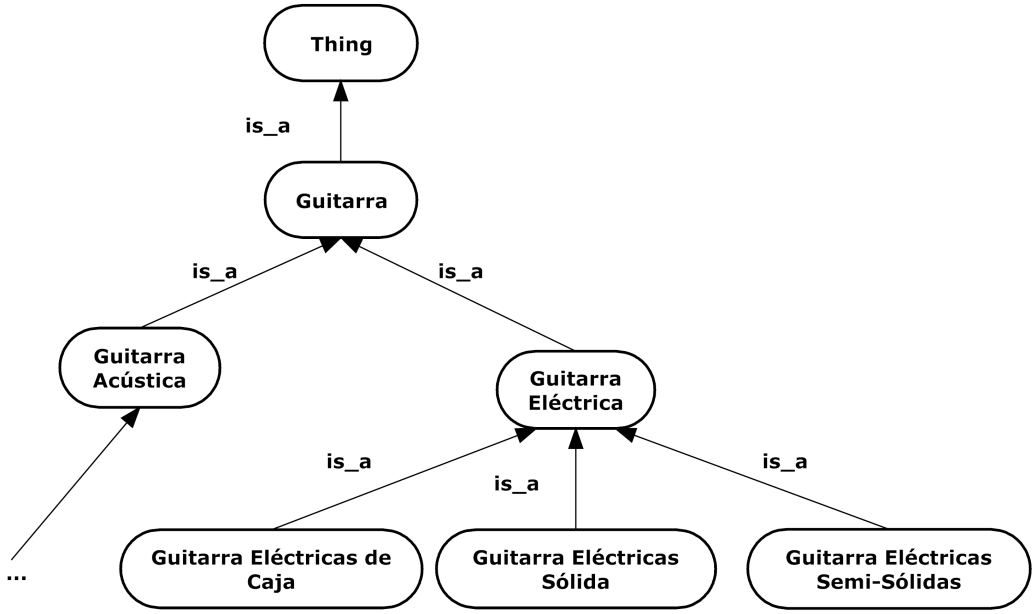

Figura 3.9: Representacion *is_a* - Representación taxonómica de relaciones de tipo (*is_a*) en una ontología sobre los tipos de guitarras.

De acuerdo al esquema anterior, su representación a través del lenguaje ontológico seria:

```
1  <owl:Class rdf:ID="Guitarra"/>
2  <owl:Class rdf:ID="Guitarra-Acústica">
3    <rdfs:subClassOf rdf:resource="#Guitarra"/>
4  </owl:Class>
5  <owl:Class rdf:ID="Guitarra-Eléctrica">
6    <rdfs:subClassOf rdf:resource="#Guitarra"/>
7  </owl:Class>
8  <owl:Class rdf:ID="Guitarra-Eléctricas-de-Caja">
9    <rdfs:subClassOf rdf:resource="#Guitarra-Eléctrica"/>
10 </owl:Class>
11 <owl:Class rdf:ID="Guitarra-Eléctricas-Sólida">
12   <rdfs:subClassOf rdf:resource="#Guitarra-Eléctrica"/>
13 </owl:Class>
14 <owl:Class rdf:ID="Guitarra-Eléctricas-Semi-Sólidas">
15   <rdfs:subClassOf rdf:resource=="#Guitarra-Eléctrica"/>
16 </owl:Class>
```

Donde, las líneas (1), (2), (5), (8), (11) y (14) representan la declaración de las clases de la ontología, y las líneas (3), (6), (9), (12) y (15) representan las subclases,

por medio de la vinculación de estas respecto a su súper-clase. Las líneas (4), (7), (10), (13) y (16) comprenden el cierre de sintaxis de una declaración.

A la hora de construir ontologías desde un nivel alto de programación, estas se pueden efectuar a través de múltiples herramientas de diseño ontológico. Como consecuencia, su enfoque suele ser intuitivo a través de interfaces amigables, tal cual precisa la herramienta Protégé.En esta, su construcción se ve plasmada en el siguiente esquema, creandose relaciones de tipo, a partir de la opciones del panel *Classes*, que permite la creación tanto de clases como subclases. La figura 3.10, representa la interfaz y opción que permite crear clases y subclases de una ontología de dominio en Protégé.

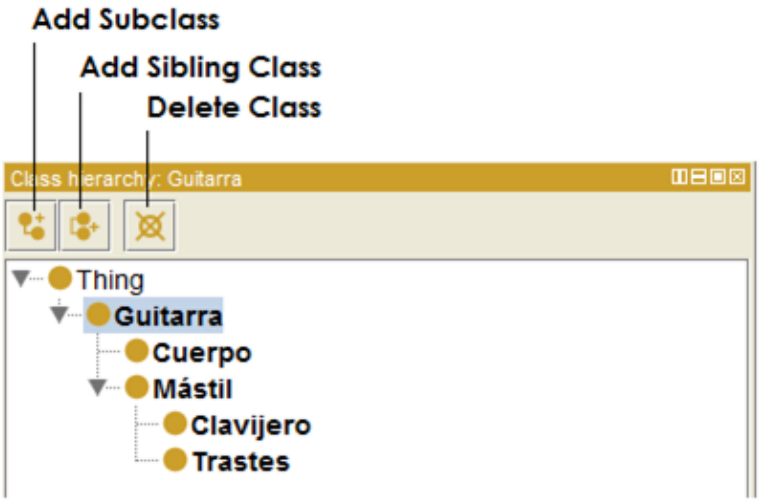

Figura 3.10: Construcción de Clases - Construcción de clases y subclases a partir de la herramienta Protégé para representar una ontología de dominio.

De acuerdo a la figura, la opción *Add Subclass* permite crear una nueva clase (en una primera instancia, al ser la primera clase de la ontología, esta será subclase de *Thing*), además la opción *Add Sibling Class* permite crear una clase hermana, mientras que la opción *Delete Class* elimina una clase. En consecuencia de lo anterior, lo suscitado en el análisis de la herramienta Protégé respecto al tratamiento ontológico de la relación de tipo, se aprecia que su tratamiento a nivel bajo de programación utiliza el lenguaje OWL.

Por otro lado, es relevante mencionar que la elaboración de una ontología de dominio, requiere de un amplio consenso; siendo mundial en algunas materias, debido a la inexistencia de una jerarquía estándar para un dominio dado, dependiendo solo

de quien la modele; por lo que su construcción debe contar con beneplácito de entidades relacionadas al dominio. Basado en esto, a la hora de diseñar ontologías, se debe conocer ciertas reglas generales que competen a la relación *is_a* y que se adoptan como vitales, a la hora de representar estas, en una ECC:

- Al momento de jerarquizar clases, es posible que ocurra transitividad entre ellas, es decir "Si B es una subclase de A y C es una subclase de B, entonces C es una subclase de A", por lo que es primordial tener conocimientos sobre las subclases directas y subclases indirectas. Una subclase directa es la subclase que esta "más cerca"de la clase, es decir, no hay clases entre la clase y sus subclases directas en la jerarquía.

- A medida que la jerarquización de clases va aumentando mantener consistencias entre ellas puede convertirse en un verdadero reto a medida que el domino evoluciona. Por lo tanto, se debe poseer una claridad sobre los tópicos del domino.

- Es importante distinguir entre una clase y su nombre, es decir, "Las clases representan conceptos en el dominio y no las palabras que denotan estos conceptos", ante esta premisa, el nombre de una clase puede cambiar si se elige una terminología diferente, pero el término como tal representa la realidad objetiva en el mundo.

- En relación a los sinónimos, "Los sinónimos para el mismo concepto no representan clases diferentes", es decir, los sinónimos son solo nombres diferentes para un concepto o término. Una buena práctica en el desarrollo de una ontología es listar los sinónimos en la documentación de la clase.

- Las jerarquías de clases no deben poseer ciclos, es decir, "Si la clase A tiene una subclase B y al mismo tiempo B es una superclase de A", esto en Protégé debe ser abstraído como una clase equivalente, tema que no es preponderante en esta investigación.

- Hay una premisa que debe ser siempre considerada, esta indica, "Si una clase tiene solamente una subclase directa, puede existir un problema de modelamiento o sino la ontología no está completa. Si hay más de una docena de subclases para una clase dada, entonces categorías intermedias adicionales pueden ser necesarias".

- Muchas veces los conceptos forman una jerarquía natural, por lo tanto, se debe representar como clases.

3.3.2 Relación taxonómica *part_of*

Al igual que las relaciones de tipo, las relaciones de partes, establecen vínculos entre los conceptos del dominio, fundadas en relaciones de composición entre sus clases, es decir, definen cuando un contenido es parte de otra.

La representación ontológica de estas relaciones, es abstracta y dependerá principalmente del diseñador ontológico, el nombre que le asigne a la relación; por lo que en base a esto, nace la criticidad a la falta de estandarización de nomenclaturas en cuanto a la asignación de su nombre, puesto que es posible encontrar relaciones de partes en diversas ontologías, asignándoles nombres como: *part-of*, *PartOF*, *part_of*, entre otras, incluso sin cohesión alguna. Como consecuencia, para efectos de estandarizar la noción de esta relación en la presente investigación, se asume el nombre de: *part_of*. Cabe mencionar que muchas ontologías pueden prescindir de esta relación, optando por otras que acentúen más su dominio, sin embargo, esta investigación solo se enfoca a las relaciones ya estudiadas, de acuerdo a los principio de Reigeluth en la Teoría de Elaboración y su ECC.

Una representación *part_of*, tiene un grado de complejidad mayor en su modelación, por lo que es esencial a la hora modelarla, comprender los conceptos tanto de propiedad como de restricciones sobre clases; el primero, se entiende como la forma en que las clases se describen a partir de una cualidad (*Property*) que representan características, atributos y relaciones entre ellas. Así mismo, el concepto de restricción de propiedad, establece restricciones sobre la forma en que las propiedades son utilizadas por las instancias de una clase.

Respecto al modelado a proponer por medio del lenguaje ontológico OWL; este presenta una serie de etiquetas que permiten modelar tanto las propiedades como las restricciones en toda ontología. Las siguientes etiquetas representan tales afirmaciones:

- *rdf:Property*, presente en OWL Lite, permite que las propiedades puedan utilizarse para establecer relaciones entre individuos (*owl:ObjectProperty*) o de individuos a valores de datos (*owl:DatatypeProperty*).

- *owl:onProperty*, presente en OWL Lite permite indicar la propiedad restringida.

- *owl:Restriction*, estas pueden ser del tipo:
  - *owl:allValuesFrom*, presente en OWL Lite, se establece sobre una propiedad con respecto a una clase. Esto significa que esta propiedad sobre una determinada clase tiene una restricción de rango local asociada a ella. De este modo, si una instancia de la clase está relacionada con un segundo individuo mediante esa propiedad, entonces puede deducirse que el segundo individuo es una instancia de una clase de la restricción de rango local.
  - *owl:someValuesFrom*, presente en OWL Lite, se establece sobre una propiedad con respecto a una clase. Una clase particular puede tener una restricción sobre una propiedad que haga que al menos un valor para esa propiedad sea de un tipo concreto. A diferencia de *owl:allValuesFrom* no restringe que todos los valores de una propiedad sean instancias de una misma clase.
  - *owl:hasValue*, presente en OWL DL y Full permite que una propiedad tenga a un determinado individuo como un valor.
- *rdfs:domain*, presente en OWL Lite; un dominio de una propiedad reduce los individuos a los que puede aplicarse una propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a esa clase.
- *rdfs:range*, presente en OWL Lite, el rango de una propiedad reduce los individuos que una propiedad puede tener como valor. Si una propiedad relaciona a un individuo con otro individuo y esta como rango a una clase, entonces el otro individuo debe pertenecer a dicha clase.

En virtud de lo anterior, la jerarquía semántica de una ontología compuesta por relaciones de tipo (*is_a*) y de partes (*part_of*), se plasman en la figura 3.11; donde se modela una pequeña ontología, que representa la composición de una guitarra. En esta, se aprecia que la clase guitarra, está compuesta por dos partes; el cuerpo o caja de resonancia y el mástil. Este último a su vez posee dos relaciones de composición por las clases clavijeros y traste. Cabe recordar que, la clase guitarra es una subclase de *Thing*(superclase de todas las clases).

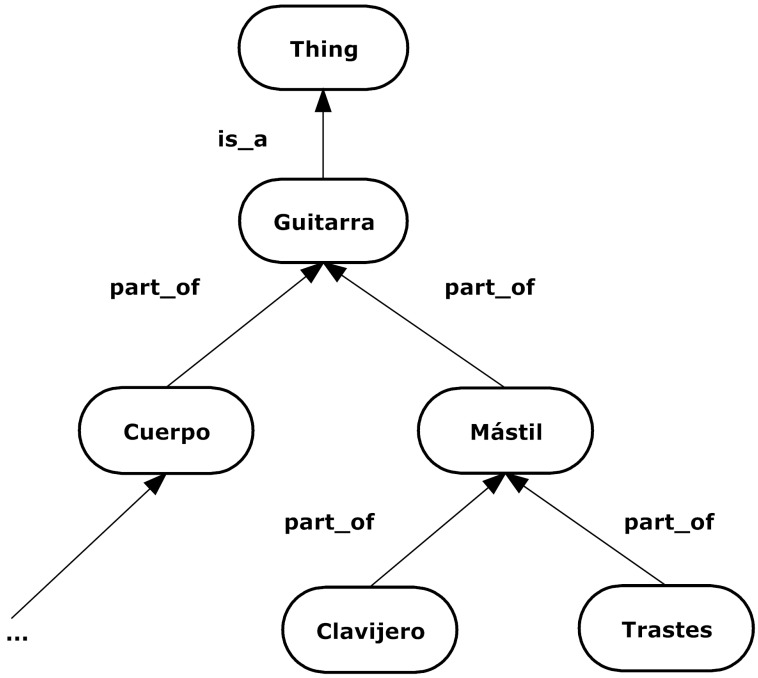

Figura 3.11: Representación *part_of* - Representación taxonómica de relaciones de partes (*part_of*) en una ontología.

De acuerdo al esquema anterior, su representación a través del lenguaje ontológico OWL seria:

```
<owl:ObjectProperty rdf:about="#part_of"/>
<owl:Class rdf:ID="Guitarra"/>
<owl:Class rdf:ID="Cuerpo"/>
  <rdfs:subClassOf rdf:resource="#Guitarra"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#part_of"/>
      <owl:allValuesFrom rdf:resource="#Guitarra"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Mástil"/>
  <rdfs:subClassOf rdf:resource="#Guitarra"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#part_of"/>
      <owl:allValuesFrom rdf:resource="#Guitarra"/>
```

```
18      </owl:Restriction>
19    </rdfs:subClassOf>
20  </owl:Class>
21  <owl:Class rdf:ID="Clavijero"/>
22    <rdfs:subClassOf rdf:resource="#Mástil"/>
23    <rdfs:subClassOf>
24      <owl:Restriction>
25        <owl:onProperty rdf:resource="#part_of"/>
26        <owl:allValuesFrom rdf:resource="#Mástil"/>
27      </owl:Restriction>
28    </rdfs:subClassOf>
29  </owl:Class>
30  <owl:Class rdf:ID="Trastes"/>
31    <rdfs:subClassOf rdf:resource="#Mástil"/>
32    <rdfs:subClassOf>
33      <owl:Restriction>
34        <owl:onProperty rdf:resource="#part_of"/>
35        <owl:allValuesFrom rdf:resource="#Mástil"/>
36      </owl:Restriction>
37    </rdfs:subClassOf>
38  </owl:Class>
```

En donde, la línea (1) indica que la propiedad *part_of* permite relaciones entre individuos al ser *ObjectProperty*; la línea (2) corresponde a la declaración de la clase Guitarra la cual tendrá implícitamente la superclase *Thing*. Las líneas (3-11) indican la composición de partes, en donde la clase Cuerpo, es parte de Guitarra. Dentro de estas, la línea (7) indica que la propiedad de restricción será *part_of*, mientras que la línea (8) indica que la restricción es de tipo *allValuesFrom* (ver capítulo II) sobre la clase Guitarra. Las siguientes líneas tienen el mismo tratamiento que lo anterior, en estas, las líneas (12-20) declaran que la clase Mástil es parte de guitarra. Las líneas (21-29) indican la composición de Mástil; declarando que Clavijero es parte de esta. Las líneas (30-38), declaran que la clase Trastes, es parte del Mástil.

Desde un nivel alto de programación, por medio de Protégé, se pueden crear las relaciones de partes mediante el panel *Object Properties* de la barra de herramientas. La figura 3.12, representa la interfaz de la herramienta.

En esta funcionalidad permite crear una propiedad, además de crear una sub propiedad por medio de la función *Add sub property*, lo que implica la jerarquización de propiedades. Adicionalmente, la funcion *Delete Property* permite eliminar propiedades.

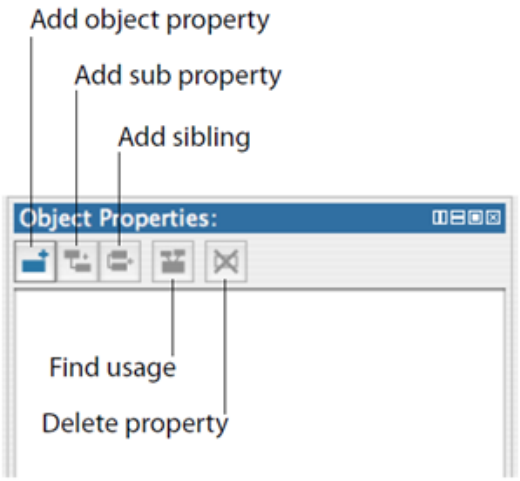

Figura 3.12: Construcción de Propiedades - Construcción de propiedades y sub propiedades a partir de la herramienta Protégé para representar una ontología de dominio.

Al momento de crear una propiedad, emana la falta de estandarización en la confección de las ontologías; la figura 3.13, representa tal proceso, denotando la libertad de creación que tiene el diseñador ontológico a la hora de asignar un nombre. Sin embargo, en la presente investigación, se utilizará el nombre de *part_of* para representar sintácticamente esta relación.

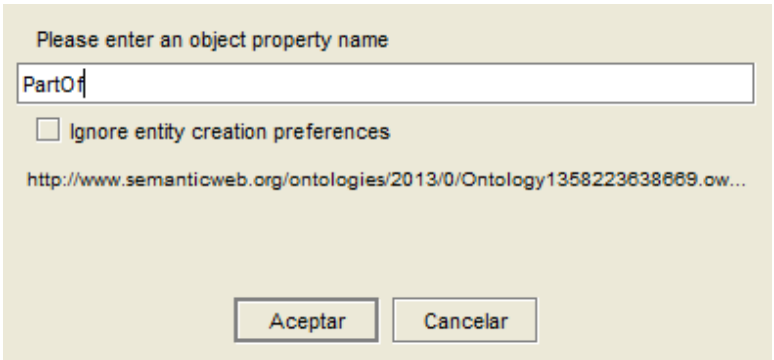

Figura 3.13: Falta de Estandarización - Representa la libertad de creación del diseñador instruccional a la hora de asignar una relación de partes con nombre *PartOf*.

En cuanto a relacionar clases mediante una propiedad, el proceso es similar al agregar superclases, diferenciándose en este proceso, que se debe optar por un tipo de restricción, *only* para *owl:allValuesFrom*, *some* para *owl:someValuesFrom* y *value* para

owl:hasValue. La figura 3.14, representa como la clase Clavijero tiene una relación de parte con Mástil, siendo su restricción *owl:allValuesFrom*, lo que indica la dependencia única de este.

Figura 3.14: Relación de Partes - Representación de la propiedad (*part_of*) y restricción entre el concepto Clavijero como parte de Mástil.

El proceso de asignar una restricción, ya sea, *only*, *some* o *value*, se debe realizar al momento de crear la relación. La figura 3.15, representa la creación de la relación de partes entre Mástil y Clavijero, en donde la funcionalidad *Restricted property* del panel de Protégé permite seleccionar una restricción a la relación. La funcionalidad *Restriction filter* permite elegir la clase con la cual se quiere hacer la relación y *Restriction type* permite elegir el tipo de restricción.

3.4 Conclusiones del Capítulo

Se presentaron en este capítulo, en primera instancia, los intentos actuales de representación de una ECC, por medio de un proceso manual e improvisado denominado TGS y elaborado a través de la herramienta Protégé. Como consecuencia de lo anterior, se presenta la propuesta que permite modelar computacionalmente una ECC, basado en el enfoque que permite obtener un modelo ontológico general para representar la Estructura de Conocimiento Conceptual, de acuerdo a los principios que Reigeluth precisa de esta dentro de la Teoría Elaborativa. Este modelo, utiliza las restricciones y propiedades ontológicas para definir las relaciones (*is_a*) y (*part_of*) de cualquier ontología de dominio confeccionada por medio del lenguaje OWL. A nivel de detalle, en primer lugar, se analiza la relación de tipo, considerando la definición de clases y

Figura 3.15: Asignación de Restricción - Representación de la asignación de restricción y la relación de partes ente conceptos.

subclases, para luego demostrar su modelación a partir de etiquetas en lenguaje OWL. Adicionalmente, se representa esta taxonomía de clases por medio de la herramienta Protégé. Luego, basado en las relaciones de tipo, las relaciones de partes (*part_of*) se modelan a partir de las de propiedades y restricciónes que existen sobre las clases que permiten modelar relaciones de composición. Adicionalmente, se presentan los pasos que permiten modelar esta por medio de la herramienta Protégé.

Envirtud de lo anterior, se concluye que ambas relacionen tienen tratamiento a través del lenguaje ontológico OWL, similar al que la herramienta Protégé genera a bajo nivel de programación, respecto al modelado de ontologías robustas.

Además se destaca la criticidad a la hora de confeccionar ontologías de dominio, debido a la carencia de una estandarización. Sin embargo, es pertinente señalar que existen intentos por remediar esta situación dentro de múltiples disciplinas cimentadas en la búsqueda de estandarización entre sus conceptos; es el caso de especialidades como medicina que posee su propio diccionario semántico denominado SNOMED `http://www.ihtsdo.org/snomed-ct/` [Price y Spackman, 2000](31), que contiene una gran lista de términos estandarizados mediante un consenso mundial; también biomedicina posee su propio vocabulario llamado UMLS (Unified Medical Language System)`http://www.nlm.nih.gov/research/umls/`, [Humphreys y Lindberg, 1993](32) y quizás el más significativo es la dirigida por las naciones unidas, que provee una terminología

exclusiva para productos y servicios denominado UNSPSC (United Nations Standard Products and Services Code) `http://www.unspsc.org/`.

El próximo capítulo, presenta en detalle la implementación del análisis del modelado de la ECC, utilizando nociones vistas en la presente capítulo.

4

Implementación de la solución

4.1 Introducción

Este capítulo tiene por objetivo mostrar la factibilidad de la implementación de la Estructura de Conocimiento Conceptual automatizada y la construcción de una aplicación que la utiliza para dar soporte a la Teoría de Elaboración. En primera instancia, se abordan las principales premisas que constituyen el proceso de generación de una ECC basadas en el otrora proceso manual TGS, abordando los principales desafíos enfrentados y plasmando futuros trabajos. Estas premisas se emprenden por separados basado en el proceso natural de construcción, comenzando con *is_a* y en base a este, las relaciones *part_of*. Luego, por cada una de estas premias, se sugiere implementación que más se aproxima basadas en la API Protégé OWL. Finalmente, bajo el alero de las sugerencias de implementación, se realiza la aplicación, explicando en detalle los códigos más relevantes para el tratamiento ontológico y que conciben la ECC.

4.2 Descripción de la solución implementada

La implementación de la propuesta se divide en dos aspectos; el primero de ellos corresponde a la **construcción de las representaciones de las relaciones de tipo y de partes de ontologías de dominio; mientras que el segundo, se enfoca en la confección de una aplicación** que hace uso de dicha representaciones. Implementar estas relaciones mediante un tratamiento ontológico computable, es la base para demostrar la aplicabilidad de esta investigación; como consiguiente, la utilización de la API, API Protégé OWL, biblioteca de código abierto de Java para el lenguaje de

ontologías Web y RDF, proporciona los cimientos que permiten dicha implementación a través de clases y métodos para cargar y guardar archivos OWL, así como consultar y manipular el modelo de datos y poder realizar inferencias. Por otro parte, la utilidad de las representaciones, se sustentan en una aplicación que hace uso y razonamiento de estas. Con este objetivo, se construye un generador de ECC adaptable a los requerimientos que el diseñador Instruccional estime pertinente. Esta aplicación, denominada **OntoConcept**, está integrada a un entorno real de Diseño Instruccional, específicamente a un editor de LD utilizado por la comunidad de E-learning. La implementación de esta solución, se centra en el tratamiento de múltiples ontologías de dominio, con extensión .OWL correspondiente al principal lenguaje ontológico.

Es importante mencionar que existen otras ontologías cuyas confecciones imposibilitan su tratamiento producto de la utilización de distintos formatos ontológicos y desemejante modalidad en la creación de clases, relaciones e instancias; considerado su tratamiento para futuros trabajos. Adicionalmente, aspectos relacionados con la carga completa de las ontologías de dominios no son implementados en su totalidad y son considerados también como parte de trabajos venideros.

4.3 Propuesta de construcción para relaciones de tipo y de partes

Las representaciones requieren el tratamiento taxonómico de ontologías de dominio originalmente confeccionadas. Como consiguiente, se precisa abordar cada una de estas relaciones por separado, de manera que sean inteligibles y poder fácilmente vislumbrar su construcción, desafíos enfrentados en la implementación así como plasmar futuras investigaciones. En primera instancia se aborda las relaciones de tipo, detallando los problemas a enfrentar, propuesta de soluciones y desafíos venideros. En segunda instancia, se abordan las relaciones de partes, la que precisan de relaciones de tipo para su implementación. En esta, también se analizan los principales obstáculos, pregonando sus soluciones y desafíos futuros. Para efectos de ejemplificar su tratamiento, durante este capítulo se utiliza como ejemplo la ontología de apoyo pedagógico **pizza.owl**.

4.3.1 Propuesta de implementación a la relación *is_a*

La relación *is_a*, precisa una concordancia jerárquica y de herencia entre dos conceptos; indicando que el concepto menor es un subtipo, una subclase del concepto padre del cual hereda sus propiedades. Como consiguiente, crear dichas relaciones jerárquicas, a partir de un tópico de la ontología original electo por el diseñador instruccional requería de un proceso manual e improvisado. Para enfrentar esta situación y basado en el análisis del modelado ontológico del capítulo anterior, se aborda una propuesta, la cual, apoyada en la API de Protégé, permitirá crear este proceso de forma automática. Para efecto de detalle en la implementación de la representación *is_a* , a continuación se formulan las premisas que abordan las directrices del proceso de modelación, plasmando en cada una las soluciones que proyectan su automatización, a través del tratamiento brindado por la API y siguiendo la idea del otrora proceso TGS.

- En relación a la búsqueda manual del tópico a tratar en una ontología de dominio que propone el proceso TGS, se sugiere un recorrido automatizado de las clases a partir de un filtro previo del tópico, cuyo resultado alistara aquellas clases que contengan al tópico definido previamente dentro de sus nombres, con la idea de recomendar tópicos que puedan ser más apropiados al dominio a tratar.

  De manera que si el diseñador instruccional desea realizar una búsqueda en alusión al tópico *Pizza* , las clases recomendadas serian: *CheeseyPizza*, *VegetarianPizza*, *PizzaBase*, *PizzaTopping* , *NamedPizza* , las cuales contienen al tópico y ofrecen una especificación del dominio. La siguiente figura 4.1, representa la propuesta para la implementación.

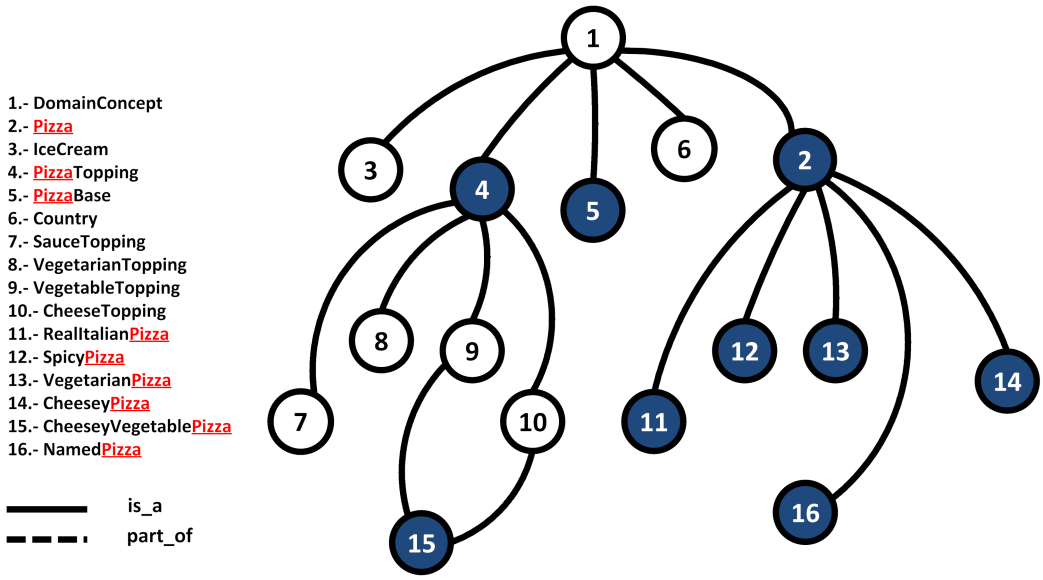

Figura 4.1: Esquema representativo de la ontología de *Pizza* - Propuesta de implementación de búsqueda automatizada

En esta imagen, se ilustra el proceso de búsqueda automatizado que se propone, representando el dominio acotado de la ontología *Pizza* basado en el tópico seleccionado. En este ejemplo, el proceso de búsqueda comienza previa elección del tópico *Pizza*; alistándose todas las clases de la ontología que comprenden al tópico buscado. En la figura, los nodos azules correspondientes a las clases {2, 4 ,5, 11, 12, 13, 14, 15, 16} contienen al tópico dentro de su nombre de clase y son listados, como posibles sugerencias.

- En cuanto a la generación de una ECC compuesta sólo de relaciones *is_a* a partir del tópico electo, se propone realizar el proceso denominado análisis interno relacional de clases, AIRC[I], es decir, a partir de la lista de clases sugeridas en la búsqueda, se opta por aquel que más se acerque a la voluntad del diseñador instruccional, comenzando el proceso AIRC que tiene como resultado la generación de una ECC particular para el dominio de clases del tópico seleccionado.

  De ahí que, la figura 4.2 representa la elección del tópico *NamedPizza*, dentro del listado de sugerencias.

[I]El acrónimo AIRC fue designado en esta investigación para representar el procedimiento de vinculación entre clases y cuyo resultante es la ECC automatizada.

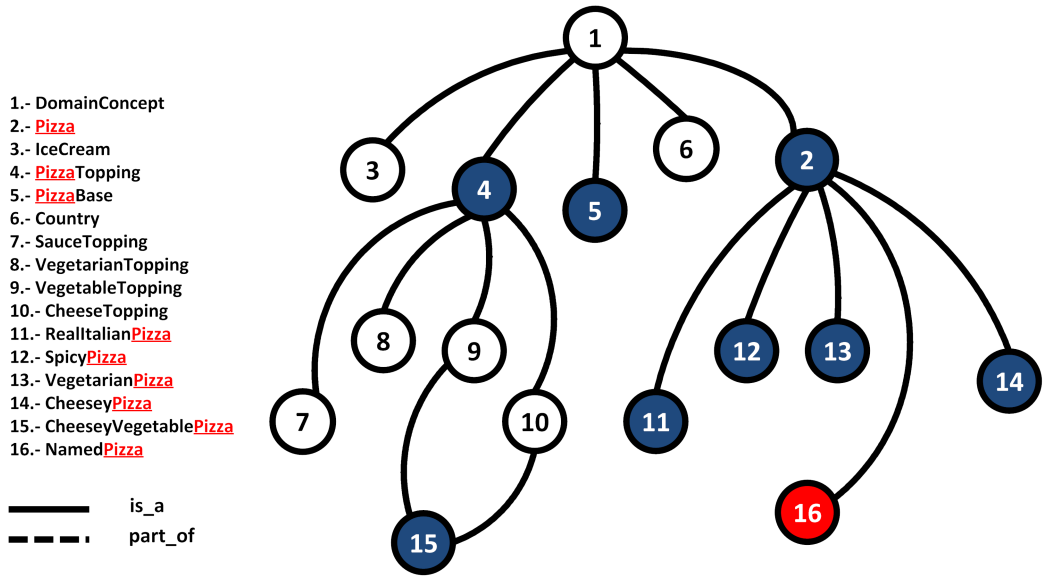

Figura 4.2: Esquema representativo de la elección de un tópico - Representación de la elección del tópico para generar la ECC, a partir de clases sugeridas.

En esta imagen, el nodo {16} que contiene a la clase *NamedPizza* es el tópico electo dentro de la lista de sugeridos para generar la ECC de relaciones de tipo.

- En cuanto al proceso AIRC, una vez electo el tópico para construir la ECC, la jerarquización de clases del dominio del tópico seleccionado se realiza de manera automática, sin intervención del usuario.

  Es decir, las clases hijas o subclases del tópico *NamedPizza* que son detalladas como *América*, *Cajun*, *LaReine*, *Soho*, generan el ECC solo por relaciones de tipo. La figura 4.3, representa el proceso de confección del dominio del *NamedPizza*.

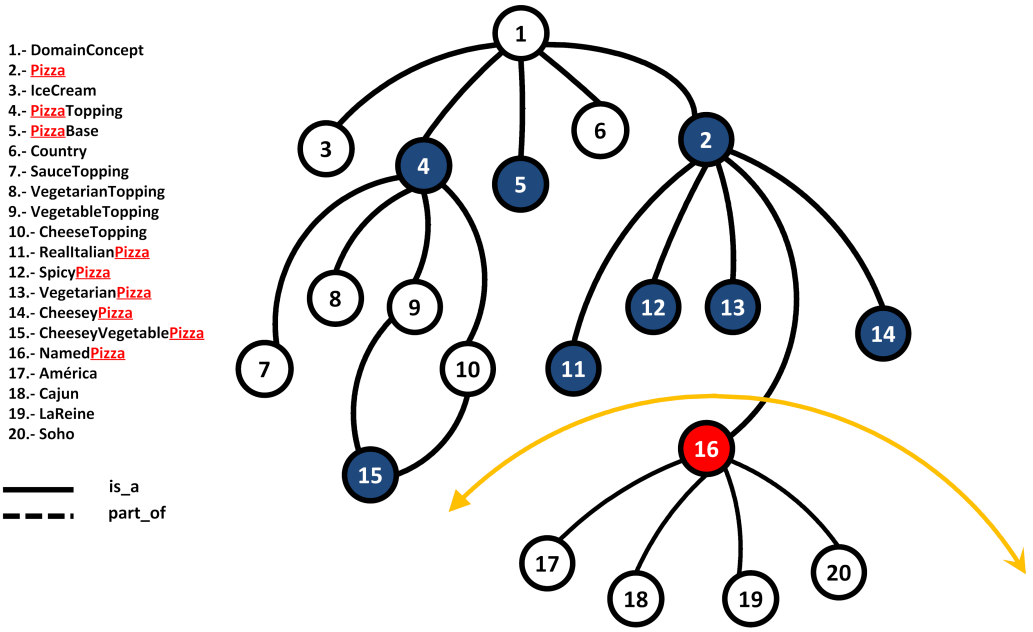

Figura 4.3: Esquema representativo del procedimiento AIRC - Procedimiento del AIRC.

Como indica la figura, al elegir el tópico *NamedPizza*, durante el proceso AIRC todas sus subclases serán exploradas para generar la ECC considerando sólo relaciones de tipo, compuesto el dominio solo por {16} y sus clases hijas {17,18,19,20}

- Un aspecto relevante en la confección de la ECC, es la profundidad semántica, es decir los niveles de clases que conforman el ECC. La presente investigación sugiere que esta dependerá del dominio del tópico seleccionado, finalizando su confección cuando son encontradas las clases *Nothing*.

  El nivel de clases, que genera el ECC, corresponderá al tamaño que abarque el dominio del tópico seleccionado. En este sentido, aquellas clases que se encuentren fuera del dominio, no son consideradas, siendo las clases *Nothing*, aquellas que indiquen cuando el último nivel de profundidad.

4.3.1.1 Desafíos enfrentados en la implementación de la relación *is_a*

El tratamiento taxonómico para esta relación, requiere que la implementación comprenda en profundidad el proceso de confección de clases, particularmente el tratamiento de subclases y los vínculos con sus superclases respectivas. Por consiguiente, durante este trabajo, la implementación se ve enfrentada a una serie de impedimentos que sugieren errores ontológicos y que obstaculizan su modelación. En virtud de lo anterior, las siguiente secciones, analizan los principales desafíos enfrentados durante el proceso de implementación de la ECC para relaciones *is_a*.

Propuesta al problema de jerarquías intermedias

La propuesta de implementación descrita, requiere del procesamiento jerárquico de clases. Es decir, relaciones de tipo que generen vínculos entre ellas, en donde una clase padre a su vez es clase hijo de otras. En este sentido, en particular la implementación del AIRC, significó un importante análisis ontológico para representar el árbol de clases pertenecientes al dominio del tópico seleccionado, tornándose esencial adoptar una técnica de recorrido de clases y de las subclases, que sea eficiente y sustentada en los parámetros que la API ofrece para su tratamiento.

Propuesta al problema de múltiples padres

Dentro del proceso de análisis interno concluyente, la ocurrencia de clases con múltiples padres fue un punto a abordar durante la presente investigación. En este sentido, el peor de los casos se podría dar en una clase conformada por múltiples superclases y todas ellas en diferentes niveles de profundidad. Esta complejidad se acentúa cuando las clases padres aún no son analizadas por el proceso de recorrido ni menos jerarquizadas. A raíz de esto se propone la creación de una lista temporal de clases, con el propósito de almacenar aquellas clases en donde sus padres aún no son jerarquizados de manera que en su momento, puedan vincularse por medio de la relación de tipo, con sus padres respectivos. La figura 4.4, ilustra el proceso de recorrido de clases y el rol de la lista temporal.

Solución a los problemas enfrentados

Para dar sustento a los problemas descritos, se implementaron dos tipos de procedimientos. El primer de ellos, ofrece una solución al problema de las jerarquías intermedias almacenando los padres directos de una clase dentro de una lista, tal cual precisa figura 4.4, en donde, cuando se recorre el nodo {15}, se almacenan dentro de la lista los padres de este {9, 10, 6} respectivamente con el propósito de situarlos correctamente dentro de la jerarquía. Sin embargo, se aprecia en la figura, que el análisis no es simultáneo para todas las clases, sino más bien secuencial, por ende, cuando se extiendan las ramificaciones del nodo {15} a sus padres, la única clase que estará jerarquizada dentro del ECC será el nodo-padre {9} quedando los otros dos {10, 6} como nodos-fantasmas, es decir, sin jerarquía ni consistencia. Ante esta dificultad se estableció un segundo procedimiento con el propósito de almacenar las relaciones jerárquicas correspondientes a los nodos-fantasma para aquellas clases que tengan múltiples padres. En este, se empleó una colección de objetos citada en el lenguaje Java como HashMap<subclase, lista de padres no jerarquizados>, en donde, la clave es la clase que posee nodos-fantasmas y el valor la lista de todos los nodos-fantasmas de la clave.

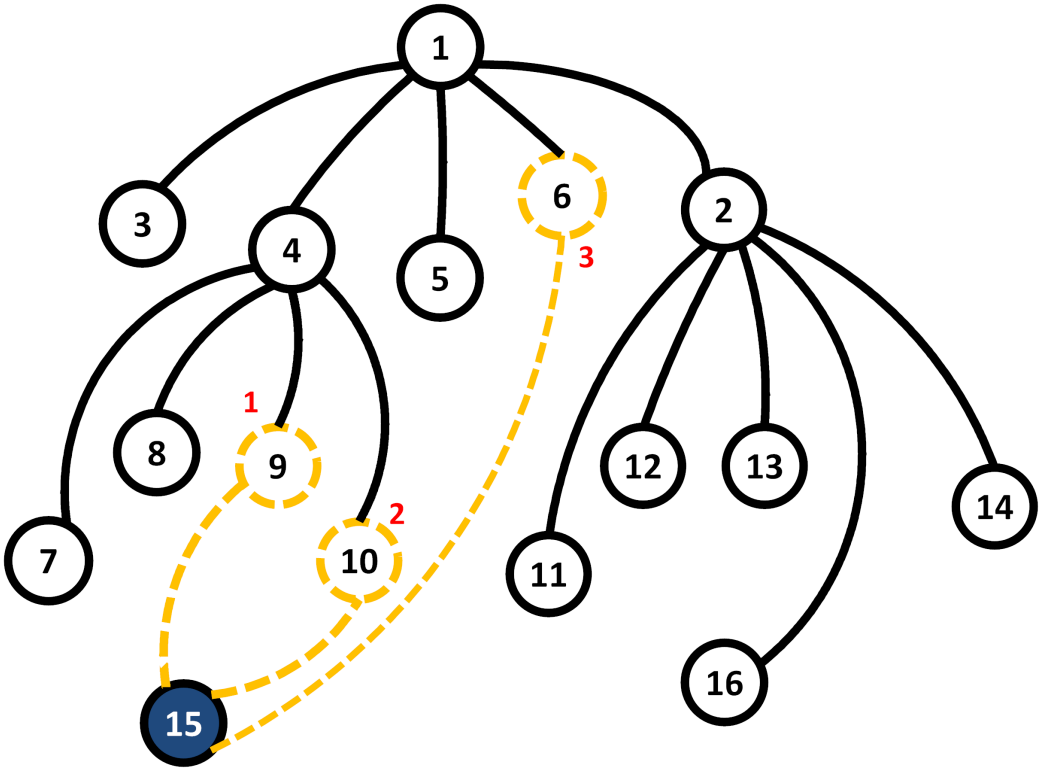

Figura 4.4: Esquema explicativo para los desafíos enfrentados en la relación *is_a* - Representación de los problemas de jerarquías intermedias y múltiples padres.

4.3.2 Propuesta de implementación a la relación *part_of*

La relación *part_of*, quien precisa una jerarquía entre dos conceptos indicando un vínculo de composición o de parte entre ellos, requiere para su implementación la ECC ya generada por relaciones de tipo. En virtud de lo anterior y basado en el análisis del modelado ontológico del capítulo anterior, se aborda una propuesta, la cual, bajo el alero de la API de Protégé permitirá extraer aquellas relaciones de partes dentro de ontologías y vincularlas si lo dispone el dominio establecido por el tópico seleccionado y el cual contempla el ECC de tipo. Cabe destacar, que el proceso de *part_of*, se realiza de manera interna y automática a partir de la ECC de tipo ya generada, siendo esta, el último paso para generar la ECC definitiva.

Para efecto de detalle en la implementación de esta propuesta, se formula las siguientes premisas que abordan las principales directrices del proceso de modelación de un *part_of*, plasmando las soluciones que proyectan a una automatización computable:

- La identificación de relaciones de composición dentro de las múltiples relaciones que constituyen a las ontologías de dominio, carecía de una automatización. Bajo esta premisa, se propone, extraer, por medio de un algoritmo el nombre que la identifique dentro de la ontología, de manera que sus clases puedan vincularse, a partir de la ECC de tipo, previamente generada.

- Sobre el tipo de restricción a asignar a la relación *part_of* es decir *only*, *some* o *value*, se diseñó que el análisis reconociera de forma automática estas, con el propósito de generar correctamente las reglas de una propiedad a nivel OWL.

4.3.2.1 Desafíos enfrentados en la implementación de la relación *part_of*

El tratamiento taxonómico para relaciones de composición en ontologías de dominios, requiere de una implementación que en primera instancia las identifique dentro de las múltiples relaciones que conforman las ontologías mediante su propiedad. En virtud de lo anterior, la siguiente sección analiza el principal desafío enfrentado durante el proceso de implementación de la solución a la relación *part_of*.

Solución a la multiplicidad de nombres asignados a una relación *part_of*

A la hora de realizar una búsqueda de la relación dentro de ontologías de dominios, la principal contrariedad, emerge de la carencia actual de una estandarización a la hora de confeccionar relaciones, lo cual dificulta abordarlas de manera directa, debido a los diferentes nombres utilizados para una relación. En este sentido, se propone, un algoritmo que mediante una expresión regular elimine todo tipo de símbolos y mayúsculas existentes, de manera que simplifique los nombres de dicha relación. Es decir, convierte a *part_of* en *partof* al igual que nombres como *part-of*, *PartOf*, entre otros para motivos de comparación.

4.4 Características generales y entorno de la aplicación

La intención de esta investigación, es abordar la carencia de una ECC automatizada, para dar soporte a lo que precisa la Teoría Elaborativa. En este sentido, resulta importante que esta representación sea utilizada en un ambiente que se integre al entorno de diseño de los LDs y a la vez capaz de vincularse con Terpsícore, herramienta que analiza la conformidad de estos con TDI, de manera que en conjunto logren dar sustento a la Teoría de Elaboración. Bajo esta premisa, el entorno de trabajo electo, corresponde al de la herramienta más utilizada por la comunidad de E-learning, ReCourse y en el cual Terpsícore también se encuentra integrado. Terpsícore, carece de un proceso automatizado para abordar dicha teoría, debido a la falta de la representación de una ECC a partir de ontologías de dominio. De esta forma, se construyó una aplicación, denominada *OntoConcept*, mediante la cual un diseñador puede confeccionar una ECC, a partir de una ontología según lo estime pertinente, de forma integrada y en un entorno ya conocido.

Los requisitos generales de la aplicación *OntoConcept* se enumeran a continuación:

- Confeccionar una ECC a partir de ontologías de dominio, flexible a la voluntad del diseñador.
- Trabajar de manera autónoma, generando ECC, dentro del entorno de trabajo.
- Vincularse a Terpsícore, para en conjunto dar soporte a la Teoría Elaborativa.
- Estar integrado funcionalmente con el entorno de diseño que proporciona ReCourse.

Estos requisitos guiaron la construcción de la aplicación, cuya funcionalidad puede apoyar tanto a diseñadores incipientes como expertos en el diseño de aprendizajes.

4.4.1 Arquitectura Técnica

Para la implementación de las soluciones *is_a* y *part_of* se utilizó Java como lenguaje de programación mediante el IDE Eclipse, versión Juno. Además se analizaron e integraron diferentes herramientas, frameworks y APIs entre las que destacan:

- ***Protégé*** *(Versión 3.x)*: Es un editor gratuito y open-source de ontologías y modelos de conocimiento. Protégé está basado en Java, es extensible y proporciona un entorno de plug-and-play que lo convierte en una base flexible para la creación rápida de prototipos y desarrollo de aplicaciones. Actualmente Protégé permite crear y mantener ontologías (definir clases, relaciones, instancias, propiedades y reglas de razonamiento) a través de sus dos formas de modelado: *The Protégé-Frames* y *The Protégé-OWL*. ***The Protégé-Frames***, permite construir ontologías basado en marcos en conformidad con el protocolo *Open Knowledge Base Connectivity (OKBC)*[I]. En este, una ontología consiste en un conjunto de clases organizadas en una jerarquía y en donde un conjunto de apartados describen las propiedades, relaciones e instancias de una clase. ***Protégé-OWL***, permite construir ontologías para la Web Semántica, en particular en el lenguaje de la W3C (OWL).

- ***Protégé-OWL API***[II] *(Usado hasta las versiones 3.x de Protégé, en las versiones posteriores se utiliza OWL API)*: Es una biblioteca open-source sobre Java para el lenguaje OWL y RDFs. La API proporciona clases y métodos para cargar, guardar, consultar, modificar y razonar sobre archivos OWL, además de estar optimizada para la implementación de interfaces gráficas. Cualquier ontología diseñada por esta API es válidas para Protégé. A diferencia de su sucesora OWL API [III], Protégé-OWL API es dependiente de Protégé y no entrega soporte a OWL 2.

[I] Para mayor información, el siguiente vinculo: `http://www.stanford.edu/class/cs227/Readings/405.pdf` ofrece un paper desarrollado por académicos de la Universidad de Stanford que profundizan sobre el protocolo OKBC.

[II] Para mayor información sobre la API visitar: `http://protege.stanford.edu/plugins/owl/api/`

[III] Para mayor información, el siguiente vinculo: `http://owlapi.sourceforge.net` ofrece una guía extensa de OWL API.

4.4.2 Implementación de la Aplicación

La integración funcional de la aplicación al editor ReCourse, precisa restricciones técnicas relevantes. ReCourse está construido en Java RCP Eclipse por lo que para lograr la integración con este editor, se debió mantenerse la filosofía de plugins utilizada en la construcción de este editor de LD. Luego del análisis del código fuente de ReCourse, se decidió construir un paquete de clases que se integrará al resto de los paquetes de clases originales. Este paquete se denominó *org.tencompetence.ldauthor.ui.views.ontoconcept*. En virtud de lo anterior y basado en la propuesta de construcción de relaciones de tipo y de partes descrita en la sección 4.2, a continuación se detalla la implementación, la cual es plasmada en el paquete de clases mencionado, cuyo resultado originan una ECC automatizada a partir de una ontología de dominio: Generar una ECC requiere de una ontología de dominio OWL . Como consecuencia la idea central para lograrlo consiste en construir una nueva ontología a partir de una original, pero cumpliendo con las condiciones taxonómicas estudiada sobre las relaciones que precisa la Teoría de Elaboración respecto al ECC. Bajo esta premisa, la confección comienza con la carga de la ontología, dando inicio a su tratamiento, el cual se plasmado en los siguientes pasos tal cual se detalló en la propuesta de implementación, específicamente en el capítulo 4.2:

4.4.2.1 Implementación *is_a*

- ***Recorrido de Clases:*** Para realizar el recorrido automatizado de las clases de una ontología, se utiliza el método *getUserDefinedOWLNamedClasses()*[I] proporcionado por la API de Protégé-OWL; este permite obtener la colección clases que dispone una ontología excluyendo las clases del sistema como *owl:Thing* y *owl:Nothing*. Para llevar a cabo el recorrido, se implementó la siguiente solución en lenguaje de alto nivel:

[I]Para mayor información sobre el método *getUserDefinedOWLNamedClasses()* visitar: `http://goo.gl/t8ySs`

```
1  for(Iterator<?> it = jenaModel.getUserDefinedOWLNamedClasses().
2    iterator();it.hasNext();) {
3    OWLNamedClass searchClass = (OWLNamedClass)it.next();
4    if(searchClass instanceof OWLNamedClass) {
5      if(!(searchClass.isMetaclass())  (searchClass.isVisible())){
6        if(searchClass.getLocalName().toLowerCase().contains(topic.
             toLowerCase())) {
7          setResultSearch(searchClass.getLocalName());
8        }
9      }
10   }
11 }
```

En donde las líneas (1) y (2) recorren todas las clases obtenidas mediante el método *getUserDefinedOWLNamedClasses()*, en la línea (5) el método *!isMetaclass()*[I] permite excluir aquellas clases que sean subclases de *rdfs:Class*. Esta premisa se sostiene en la naturaleza compleja que presenta el lenguaje OWL en relación a la conformidad de clases con RDF, dándose por aceptada su utilización para efectos de aplicabilidad del método, basados en evitar errores de sintaxis propias de la compatibilidad entre *rdfs:Class* y *owl:Class*, esta última con mayor grado de expresividad computacional. El método *isVisible()*[II] permite comprobar que una clase ha sido declarada como invisible al usuario, mientras que en la línea (7) los resultados de la búsqueda son guardados en una Lista, llamada *setResultSearch* de tipo *ArrayList<String>*, en la cual cabe destacar que el método *getLocalName()*[III] permite obtener la parte local de una URI es decir el nombre, por ejemplo, para *owl:Class* el método entregaría como resultado *Class*.

[I] Para mayor información sobre el método *isMetaclass()* visitar: `http://goo.gl/EfSh2`
[II] Para mayor información sobre el método *isVisible()* visitar: `http://goo.gl/gOUgX`
[III] Para mayor información sobre el método *getLocalName()* visitar: `http://goo.gl/LhQ3l`

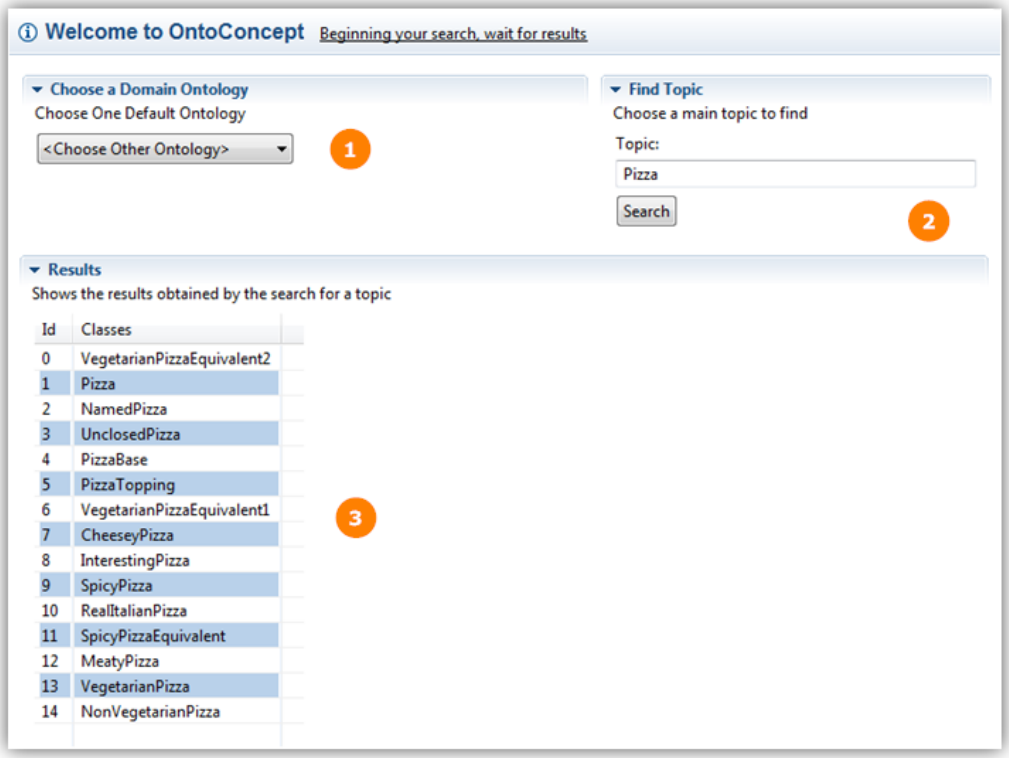

Figura 4.5: Panel de búsqueda - Representación del recorrido de clases en aplicación *OntoConcept*.

La imagen 4.5 reproduce dicho proceso, a partir de la interfaz dispuesta para la aplicación. En esta se destacan los siguientes puntos:

1. Permite seleccionar una ontología almacenada a través del directorio de archivos
2. Permite ingresar el Filtro de tópicos, con el cual se realizara el recorrido
3. Clases listadas que contienen el tópico definido en (2)

- ***Análisis Interno Relacional de Clases***: Seleccionado un tópico (clase) de la lista *setResultSearch*, se realiza un recorrido a todas sus subclases dentro de *jenaModel*[I] (incluyendo las subclases de las subclases) hasta llegar a las clases reservadas *owl:Nothing*, generando de este modo, el dominio de clases del tópico seleccionado. Para lograr esto se utilizó en primera instancia el método *getSubClasses (**boolean transitive**)*[II] (línea 1) que permite datos tipo *booleanos* como argumento; ***true*** para obtener las subclases de las subclases, mientras que ***false*** se utiliza para obtener las subclases directas. Durante el recorrido, por cada clase retornada mediante el método *getSubClasses(**true**)* se obtendrán sus superclases respectivas a través el método *getNamedSuperclasses (**boolean transitive**)*[III](línea 6), el cual admite datos tipo *booleanos* como argumento; ***true*** para obtener todas las superclases hasta llegar a la clase *owl:Thing* y ***false*** para obtener las superclases directas.

  Estas superclases son almacenadas en la lista *arrayAddSuperclass* (línea 14) de tipo *ArrayList<**OWLNamedClass**>*, ssi, las superclases aún no están presentes en el ECC resultante, para esto se utiliza el método *getOWLNamedClass(**String name**)* (línea 8) que permite buscar una clase dentro de una ontología a través de su nombre, retornando *null* en caso que no exista.

```
1 for(Iterator<?> it = searchedTopic.getSubclasses(true).iterator();it.
    hasNext();) {
2   RDFSClass searchSubClass = (RDFSClass)it.next();
3   if((searchSubClass instanceof OWLNamedClass)) {
4     if(!(searchSubClass.isMetaclass()) (searchSubClass.isVisible()))
        {
5       List<OWLNamedClass> arrayAddSuperclass = new ArrayList<
          OWLNamedClass>();
6       for(Iterator<?> jt = searchSubClass.getNamedSuperclasses(false
          ).iterator(); jt.hasNext();) {
7         OWLNamedClass searchSuperClass = (OWLNamedClass)jt.next();
```

[I] *jenaModel* hace referencia a la ontología original, mientras que *eccModel* hace referencia al ECC resultante que representa a la ontología que se está originando mediante el tratamiento taxonómico de relaciones de tipo

[II] Para mayor información sobre el método *getSubClases(boolean transitive)* visitar: `http://goo.gl/vDJQ1`

[III] Para mayor información sobre el método *getNamedSuperclasses(boolean transitive)* visitar: `http://goo.gl/C4Ur0`

```
8           if(eccModel.getOWLNamedClass(searchSuperClass.getLocalName())
                != null) {
9             OWLNamedClass var1 = eccModel.getOWLNamedClass(
                  searchSuperClass.getLocalName());
10            OWLNamedClass var2 = eccModel.createOWLNamedClass(
                  searchSubClass.getLocalName());
11            var2.addSuperclass(var1);
12            var2.removeSuperclass(eccModel.getOWLThingClass());
13          }else {
14            arrayAddSuperclass.add(searchSuperClass);
15            hasMapArray.put(searchSubClass, arrayAddSuperclass);
16          }
17        }
18      }
19    }
20  }
21  if(hasMapArray.isEmpty() != true) {
22    for(Iterator<?> it = hasMapArray.entrySet().iterator();it.hasNext()
        ;) {
23      Map.Entry<?, ?> e = (Map.Entry<?, ?>)it.next();
24      RDFSClass consultSuperclass = eccModel.getOWLNamedClass(((
          RDFSClass) e.getKey()).getLocalName());
25      for(Iterator<?> jt = ((List<OWLNamedClass>) e.getValue()).
          iterator();jt.hasNext();) {
26        OWLNamedClass superClass = (OWLNamedClass)jt.next();
27        RDFSClass consultSubclass = eccModel.getOWLNamedClass(
            consultSuperclass.getLocalName());
28        RDFSClass ind = eccModel.getOWLNamedClass(superClass.
            getLocalName());
29        if(ind != null) {
30          consultSubclass.addSuperclass(ind);
31        }
32      }
33    }
34  }
```

Si la superclase esta dentro de la ECC resultante, entonces, la clase se deberá crear y jerarquizar, formando así una relación *is_a* . Para crearla dentro del ECC resultante se utiliza el método *createOWLNamedClass(**String name**)* (línea 9-10), que permite crear una clase ingresando su nombre como parámetro. Para jerarquizarla, se utiliza el método *addSuperclass(**RDFSClass superclass**)* (línea 11) que permite crear jerarquías mediante la etiqueta *rdfs:subClassOf*, por ejemplo,

A.addSuperclass(B.getLocalName()) en donde A será una subclase de B.

Para evitar jerarquías inválidas con la clase *owl:Thing*, se utilizan los métodos *removeSuperclass(**RDFSClass superclass**)* que elimina una determinada superclase y *getOWLThingClass()* que permite hacer referencia a *owl:Thing*, en conjunto estos métodos eliminan cualquier jerarquía de una clase con *owl:Thing* (línea 12).

Para evitar el problema de los nodos-fantasmas mencionado en capítulos anteriores, las superclases que no estén aún dentro de la ECC resultante, serán almacenadas dentro
hashMapArray de tipo *HashMap<**RDFSClass, List<OWLNamedClass**>>*.

Para finalizar, la jerarquización de las clases y en consecuencia generar un ECC con relaciones de tipo (*is_a*), se recorre el *hashMapArray* con el propósito de crear las jerarquías pendientes(líneas 22-34). Para esto se utilizan los métodos proporcionados por la API de Java, respecto al uso de *HashMap*.[I]

[I]Para mayor información, la dirección: `http://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html` ofrece el material oficial respecto a los métodos *HashMap* de la API de Java.

La imagen 4.6 reproduce el proceso de Análisis Interno Relacional de Clases, al elegir la clase *NamedPizza*.

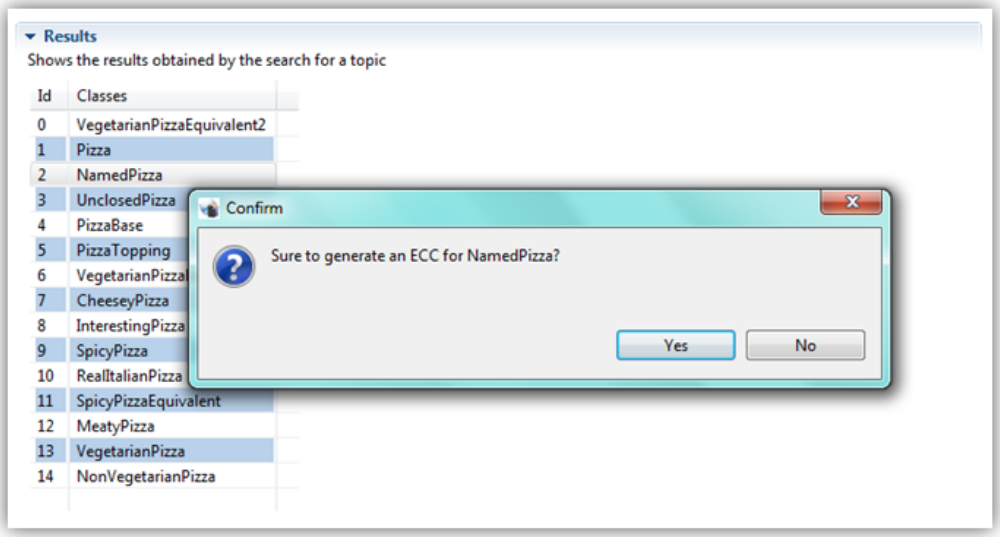

Figura 4.6: Panel de búsqueda - Representación del Análisis Interno Relacional de Clases en *OntoConcept*.

4.4.2.2 Implementación *part_of*

- ***Construcción de relaciones de composición:*** Para identificar las relaciones de partes en la ontología original y construirlas dentro del contexto del ECC de relaciones de tipo, se utilizaron las definiciones avanzadas proporcionadas por Protégé-OWL API para el manejo de OWL DL. En una primera instancia se utilizó el método *createOWLObjectProperty(**String name**)* para crear una restricción de tipo *OWLRestriccion* para la propiedad *part_of* mediante la etiqueta *owl:ObjectProperty*. Para determinar qué clases poseen relaciones de partes, se recorren las subclases del tópico con la finalidad de buscar sus restricciones mediante el método *getRestrictions(**boolean transitive**)* que permite obtener todas las restricciones de una clase; este método permite datos de tipo booleano como argumento; ***true*** para incluir las restricciones de las superclases, útil para el manejo de *OWLCardinalityBase*(*OWLCardinality*, *OWLMinCardinality* y *OWLMaxCardinality*) y ***false*** para obtener las restricciones directas de una clase.

Para identificar que las restricciones tengan la propiedad *part_of*, se utiliza el método *getOnProperty()* que permite obtener la propiedad a través de la etiqueta *owl:onProperty*. Al poseer la propiedad es necesario compararla con el dato de tipo *OWLObjectProperty* devuelto por el método *createOWLObjectProperty(**String name**)* con el propósito de filtrar aquellas propiedades de tipo *part_of*. Para evitar problemas de estandarización es necesario evitar las mayúsculas y símbolos; para lograr esto se usan los métodos de la API de Java *toLowerCase()* que convierte la cadena a minúsculas y *replaceAll(**String regex, String replacement**)* que permite reemplazar las cadenas(que coinciden con las expresión regular dada por ***regex*** [I]) por la cadena ***replacement***. En nuestro caso se utiliza la expresión regular [∧a-z] para eliminar todo tipo de símbolo o número.

Si la clase posee una propiedad *part_of*, hay que identificar a continuación el tipo de restricción (*OWLSomeValuesFrom*, *OWLAllValuesFrom* y *OWLHasValue*) ya que el tratamiento difiere para cada uno. Si el tipo de restricción es OWLSomeValuesFrom, para construir la relación es necesario conocer las clases que la componen a través del método *getSomeValuesFrom()* que permite obtener las superclases que forman la relación de partes mediante la etiqueta *owl:someValuesFrom*. Obtenidas las superclases es necesario confirmar que su dominio esté contenido en el ECC de relaciones de tipo, si el dominio corresponde la relación es creada mediante los métodos *createOWLSomeValuesFrom(**RDFProperty property, RDFResource filler**)* y *addSuperclass(**RDFSClass superclass**)*.

```
1 OWLObjectProperty createProperty = eccModel.createOWLObjectProperty("
    partof");
2 for(Iterator<?> it = searchedTopic.getSubclasses(true).iterator();it.
    hasNext();) {
3   RDFSClass searchSubClass = (RDFSClass)it.next();
4   if( searchSubClass instanceof OWLNamedClass){
5     for(Iterator<?> jt = ((OWLNamedClass)searchSubClass).
        getRestrictions(false).iterator();jt.hasNext();) {
6       OWLRestriction searchProperty = (OWLRestriction)jt.next();
7       RDFProperty property = searchProperty.getOnProperty();
8       if(property.getLocalName().toLowerCase().replaceAll("[^a-z]", "
          ").equals(createProperty.getLocalName().toLowerCase())) {
```

[I](1) Para más información sobre regex, la siguiente dirección proporciona material asociado: `http://regexpal.com/`

```
9           OWLNamedClass  subClass = eccModel.getOWLNamedClass(
                searchSubClass.getBrowserText());
10          createProperty = eccModel.createOWLObjectProperty("part_of");
11          if(searchProperty instanceof OWLSomeValuesFrom) {
12            RDFResource someValuesFrom = (RDFResource) ((
                  OWLSomeValuesFrom) searchProperty).getSomeValuesFrom();
13            if(eccModel.getOWLNamedClass(someValuesFrom.getLocalName())
                   !=null) {
14              RDFSClass superClass = eccModel.getOWLNamedClass(
                    someValuesFrom.getLocalName());
15              OWLSomeValuesFrom superRestriction = eccModel.
                    createOWLSomeValuesFrom(createProperty, superClass);
16              subClass.addSuperclass(superRestriction);
17            }
18          }
19          if(searchProperty instanceof OWLAllValuesFrom) {
20            RDFResource AllValuesFrom = (RDFResource) ((
                  OWLAllValuesFrom) searchProperty).getAllValuesFrom();
21            if(eccModel.getOWLNamedClass(AllValuesFrom.getLocalName())
                   !=null) {
22              RDFSClass superClass = eccModel.getOWLNamedClass(
                    AllValuesFrom.getLocalName());
23              OWLAllValuesFrom superRestriction = eccModel.
                    createOWLAllValuesFrom(createProperty, superClass);
24              subClass.addSuperclass(superRestriction);
25            }
26          }
27          if(searchProperty instanceof OWLHasValue) {
28            RDFResource hasValue = (RDFResource) ((OWLHasValue)
                  searchProperty).getHasValue();
29            if(eccModel.getOWLNamedClass(hasValue.getLocalName()) !=
                  null) {
30              RDFSClass superClass = eccModel.getOWLNamedClass(hasValue
                    .getLocalName());
31              OWLHasValue superRestriction = eccModel.createOWLHasValue
                    (createProperty, superClass);
32              subClass.addSuperclass(superRestriction);
33            }
34          }
35        }
36      }
37    }
38  }
```

4.5 Vinculación con Terpsícore

La siguiente sección, presenta los cimientos que encauzan la integración de *OntoConcept* con la herramienta Terpsícore y cuyo resultado permite dar conformidad a la Teoría de Elaborativa. En primera instancia se analiza la arquitectura técnica necesaria para dicha vinculación, en esta, se establecen los criterios en el uso de versiones de lenguajes y del entorno de trabajo, el editor ReCourse. Luego, se analiza el marco teórico en que desenvuelve Terpsícore, de manera de contextualizar su aporte al área instruccional, indicando su propósito y su interacción con LDs, dando conformidad a su construcción por medio de teorías de DI, particularmente, la Teoría Elaborativa. Finalmente, se especifica el proceso para la integración; preponderante lo sustancial que resulta ser *OntoConcept_Graph*, debido a la representación gráfica que precisa y el apoyo que este presta para la confección de LDs en conformidad con la Teoría de Elaboración.

4.5.1 Arquitectura técnica de la vinculación

- ***The Eclipse Visualization Toolkit*** *(Versión 1.2.x)*: Es un conjunto de componentes de visualización para Eclipse construidos sobre las librerías *SWT* y *Draw2D*. *Zest* esta conformado de los siguientes componentes:

  - *Graph*: es una extensión de *FigureCanvas* y permite mantener los nodos y sus conexiones.
  - *GraphNode*: nodo simple que posee propiedades tales como color, tamaño, ubicación y etiqueta.
  - *GraphConnection*: conexión gráfica que almacenan los nodos de origen y destino y las propiedades de la conexión, tales como color y ancho de línea.

- ***Java Architecture for XML Binding(JAXB)****(Versión 2.2.6)*: Es una API que permite representar un documento XML mediante clases en Java, de manera de poder recuperar y almacenar datos XML sin la necesidad de implementar un conjunto de métodos específicos como en las APIs JDOM, DOM4J y SAX entre las más representativas.

4.5.2 Aplicación Terpsícore

Terpsícore es una herramienta desarrollada bajo el alero de la arquitectura del editor ReCourse, y cuya funcionalidad es analizar la conformidad de un LD en relación a una teoría de diseño instruccional, mediante la utilización de reglas e inferencia SWRL que permiten representar los métodos de un DI, generando un archivo en formato XML con el resultado de la validación de cumplimiento de cada regla y sugerencias, de manera que se pueda adoptar un LD que cumpla con las indicaciones de los métodos que las teorías precisan, con un soporte computacional, lo que no existía en la literatura. La siguiente figura representa de análisis de conformidad de un LD con una TDI, a través de Terpsícore.

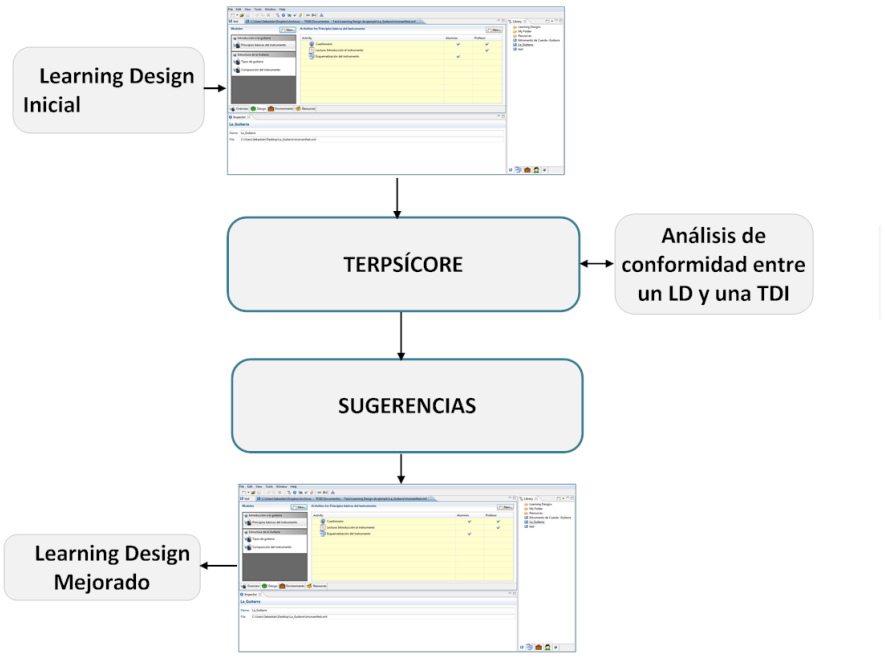

Figura 4.7: Esquema Terpsícore - Proceso General de Terpsícore.

Actualmente Terpsícore permite analizar un LD en relación a la Teoría de Elaboración mediante la elección de alguno de sus métodos de secuenciación de dominio: Conceptual Elaboration Sequence, Theorical Elaboration Sequence y Simplifying Conditions Sequence. Para poder llevar a cabo esta inferencia mediante reglas, es necesario seleccionar una ontología de dominio, en la cual se extraerán los tópicos de aprendizaje

a través de la representación de esta por medio de estructura de conocimiento conceptual que en la literatura precisa la Teoría de Elaboración. Debido al tamaño de las ontologías de dominio Terpsícore utiliza una estructura llamada Topics General Schema (TGS) destinada a ser intermediaria entre una ontología de dominio y un TDI.

Actualmente este proceso es completamente manual, basado en descomponer la ontología original, y forzarla a una estructura temporal con las relaciones pertinentes. La siguiente imagen ilustra la relación entre ontologías de dominio, TGS y el Catalogo de TDI (33).

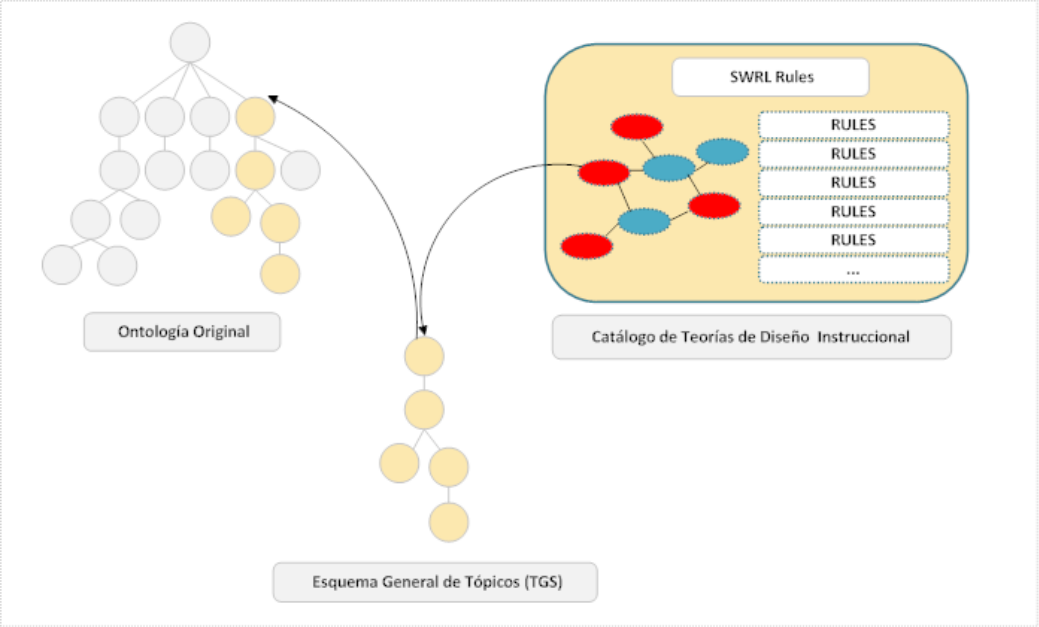

Figura 4.8: Esquema TGS - Relación entre ontologías de dominio, TGS y Catalogo de TDI

4.5.3 Propuesta de implementación de la vinculación

La propuesta de vinculación apunta al tratamiento actual del Topic General Schema; con la finalidad de evitar el proceso manual e improvisado de instanciación de clases y relaciones que ostenta por uno más bien automatizado. La vinculación hacia Terpsícore parte de la premisa de otorgar a diseñadores instruccionales y usuarios no expertos en el área ontológica una interfaz intuitiva y amigable, abstrayéndose de mecanismos técnicos informáticos que dificultan su tratamiento. En virtud de lo anterior, la ECC imple-

mentada precisa ser vinculada de tal forma que la representación de las relaciones *is_a* y *part_of* gocen de dicha interfaz dentro del entorno de Terpsícore. Por consiguiente, a partir de la implementación de OntoConcept, se propuso diseñar una extensión de ésta, de manera que sea posible representar gráficamente la ECC automatizada. Dicha extensión se conoce como *OntoConcept_Graph* y la estrategia utilizada en su construcción y vinculación con Terpsícore se aborda en la siguiente sección:

4.5.3.1 Estrategia de generación gráfica:

Representar de forma gráfica, requiere del uso de la librería ***Zest***; la cual permite que cada clase de la ECC fuese representada como un nodo que indica el nombre de la clase y la cantidad de subclases que esta posee. Con respecto a las relaciones, tanto *is_a* como *part_of* son representadas mediante la figura de un rombo y dentro del cual contiene el nombre de la relación respectiva. Sin embargo, para lograr esta gráfica a partir de la ECC implementada cuya extensión es ***.OWL***, fue necesario en primera instancia convertir esta ontología a un formato ***XML***, siendo la única forma que pudiese ser interpretado por de ***Zest***. Para esto se utilizó ***JAXB***, que mediante un algoritmo convierte un ***OWL*** en un ***XML*** interpretable.

El siguiente código permite representar a la etiqueta *<owl:Class>* en ***XML*** como *<class id="0" name="NamedPizza" type="SUPER"> <note>5 </note></class>* en donde:

- *id:* Es un identificador único por clase
- *name:* Indica el nombre de la clase
- *type:* Indica si la clase es padre o hijo
- *note:* Muestra la cantidad de sub-clases de la clase.

```
1  @XmlRootElement(name = "class")
2  @XmlType(propOrder = {"note", "type", "name", "id"})
3  public class Class {
4    private String id;
5    private String name;
6    private String type;
7    private String note;
8
9    @XmlAttribute(name="id")
10   public String getId() {
11     return id;
12   }
13
14   public void setId(String id) {
15     this.id = id;
16   }
17   @XmlAttribute(name="name")
18   public String getName() {
19     return name;
20   }
21
22   public void setName(String name) {
23     this.name = name;
24   }
25   @XmlAttribute(name="type")
26   public String getType() {
27     return type;
28   }
29
30   public void setType(String type) {
31     this.type = type;
32   }
33
34   @XmlElement
35   public String getNote() {
36     return note;
37   }
38
39   public void setNote(String note) {
40     this.note = note;
41   }
42 }
```

4. IMPLEMENTACIÓN DE LA SOLUCIÓN

Para mayor comprensión de la estrategia utilizada en relación a la conversión de OWL a XML. La siguiente figura representa dicho proceso.

Figura 4.9: Representación gráfica de la ECC - Esquema que describe el proceso de conversión de OWL a XML

4.5.3.2 Proceso de vinculación en TGS

Para que la vinculación se pueda desarrollar es necesario que la ECC implementada se integre a la estructura ontológica conocida como Esquema General de Tópicos (TGS), cumpliendo cabalmente el proceso de instanciación de sus clases y relaciones pero desde una visión automatizada. Para llevar a cabo este proceso, en primera instancia tanto las las relaciones *is_a* y *part_of* como las clase de esta ontología (ECC) deben ser instanciada mediante la API Protégé-OWL, utilizando métodos que ésta dispone para el manejo de individuos.

En primer lugar, se recorren todas las clases de la ECC transformándolas en instancias mediante el método *createOWLIndividual(**String name**)*[I]. La siguiente imagen representa el segmento de código que realiza dicha aseveración.

```
1 for(Iterator<?> it = eccModel.getUserDefinedOWLNamedClasses().iterator();
      it.hasNext();) {
2   OWLNamedClass searchClass = (OWLNamedClass)it.next();
3   if(retModel.getOWLIndividual(searchClass.getLocalName()) != null) {
4     retModel.getOWLIndividual(searchClass.getLocalName()).delete();
5   }
6   knowledgeItemInstance.createOWLIndividual(searchClass.getLocalName());
7 }
```

Por otro lado, las relaciones fueron transformadas a instancias mediante el método *getOWLProperty(**String name**)*[II] que permite una propiedad de instanciación. La siguiente imagen representa el segmento de código que realiza dicha aseveración.

```
1 OWLProperty hasKind = retModel.getOWLProperty("concept-hasKind");
2 OWLProperty hasPart = retModel.getOWLProperty("concept-hasPart");
```

Para simular las relaciones *is_a* en los individuos, tal cual precisa TGS en su proceso manual, se utilizó el método *addPropertyValue(**RDFProperty property, Object value**)*[III] que al igual que el método *addSuperClass(**RDFSClass superclass**)* usado

[I]Para mayor información sobre el método *createOWLIndividual(String name)* visitar: `http://goo.gl/jfbuF`

[II]Para mayor información sobre el método *getOWLProperty(String name)* visitar: `http://goo.gl/g4qHW`

[III]Para mayor información sobre el método *addPropertyValue(RDFProperty property, Object value)* visitar:`http://goo.gl/JJsCc`

para las clases, permite crear jerarquías entre individuos; por ejemplo una sintaxis: *A.addPropertyValue(hasKind,B)* en donde B será un sub-individuo de A. Cabe destacar que *hasKind* es equivalente que *is_a* dentro de la instanciación.

Para simular las relaciones *part_of* en los individuos es idéntico al procedimiento anterior, pero es necesario además, identificar el tipo de restricción: *OWLSomeValuesFrom*, *OWLAllValuesFrom* u *OWLHasValue*. Cabe destacar que *hasPart* es equivalente que *part_of* dentro de la instanciación.

```
1  for(Iterator<?> it = eccModel.getUserDefinedOWLNamedClasses().iterator();
       it.hasNext();) {
2    OWLNamedClass searchClass = (OWLNamedClass)it.next();
3    OWLIndividual superClass = retModel.getOWLIndividual(searchClass.
         getLocalName());
4    for(Iterator<?> jt = searchClass.getSubclasses(false).iterator();jt.
         hasNext();) {
5      OWLNamedClass searchSubClass = (OWLNamedClass)jt.next();
6      OWLIndividual subClass = retModel.getOWLIndividual(searchSubClass.
           getLocalName());
7      superClass.addPropertyValue(hasKind, subClass);
8    }
9    for(Iterator<?> kt = ((OWLNamedClass)searchClass).getRestrictions(false)
         .iterator();kt.hasNext();) {
10     OWLRestriction searchProperty = (OWLRestriction)kt.next();
11     if(searchProperty instanceof OWLSomeValuesFrom) {
12       RDFResource someValuesFrom = (RDFResource) ((OWLSomeValuesFrom)
             searchProperty).getSomeValuesFrom();
13       OWLIndividual subClass = retModel.getOWLIndividual(someValuesFrom.
             getLocalName());
14       superClass.addPropertyValue(hasPart, subClass);
15     }
16     if(searchProperty instanceof OWLAllValuesFrom) {
17       RDFResource allValuesFrom = (RDFResource) ((OWLAllValuesFrom)
             searchProperty).getAllValuesFrom();
18       OWLIndividual subClass = retModel.getOWLIndividual(allValuesFrom.
             getLocalName());
19       superClass.addPropertyValue(hasPart, subClass);
20     }
21     if(searchProperty instanceof OWLHasValue) {
22       RDFResource hasValue = (RDFResource) ((OWLHasValue) searchProperty).
             getHasValue();
23       OWLIndividual subClass = retModel.getOWLIndividual(hasValue.
             getLocalName());
```

```
24            superClass.addPropertyValue(hasPart, subClass);
25        }
26    }
27 }
```

4.5.3.3 Integración al entorno de Terpsícore

Como complemento al proceso automatizado de la nueva estructura TGS, se sugiere una representación gráfica, por medio de *OntoConcept_Graph*, la cual facilite la interacción del diseñador instruccional al momento de vincular una Actividad de Aprendizaje de un LD con un Tópico de Aprendizaje proveniente de la ECC implementada, dentro entorno Terpsícore, como parte del análisis de conformidad del LD con la Teoría Elaborativa.

El otrora proceso de vinculación de tópicos que Terpsícore realizaba, requería llamar a la estructura TGS manual, obteniendo las clases instanciadas previamente y de forma manual para luego realizar dicha vinculación. Este proceso se plasma en la siguiente figura en donde se representa el antiguo tratamiento:

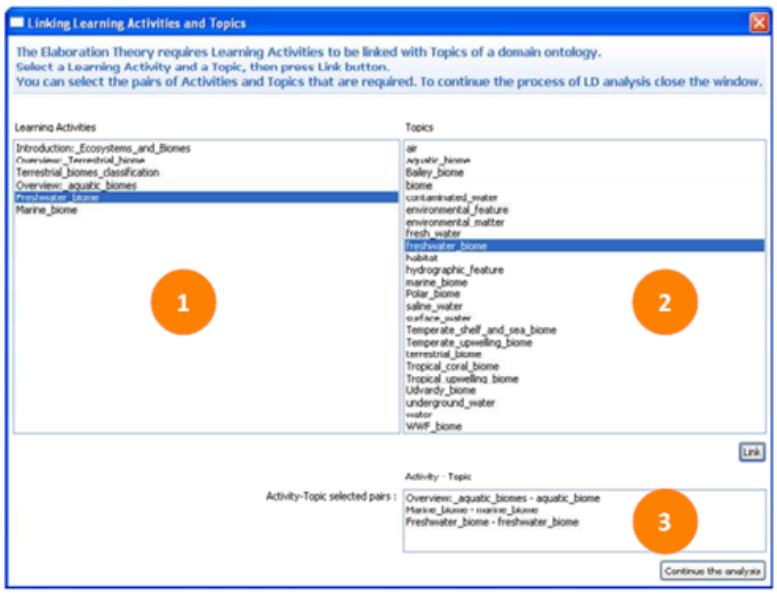

Figura 4.10: Herramienta Terpsícore - Otrora pantalla de Terpsícore que permite definir la vinculación entre Actividades de Aprendizaje y Tópicos de Aprendizaje de la ía de dominio,

En esta figura se aprecian tres secciones de interacción con el diseñador instruccional dentro de la pantalla de Terpsícore:

1. Corresponde a la lista de Actividades de Aprendizajes del LD, los cuales deben ser vinculados con los Tópicos de Aprendizajes para su posterior análisis de conformidad con la Teoría Elaborativa.

2. Corresponde a la lista de Tópicos de Aprendizajes de la ECC, obtenidos de antemano, por medio del otrora proceso manual TGS.

3. Permite vincular cada Actividad con su Tópico pertinente. Se realiza por cada vinculación.

Por otro lado, la aplicación *OntoConcept* en conjunto a su extensión gráfica permite no solo la automatización del proceso TGS, sino también representa una ECC gráficamente dentro del entorno de Terpsícore, permitiendo la vinculación de los Tópicos de Aprendizajes dispuestos en esta ECC con las Actividades de Aprendizajes del LD. Este proceso se plasma en la siguiente figura en donde se representa su nuevo tratamiento.

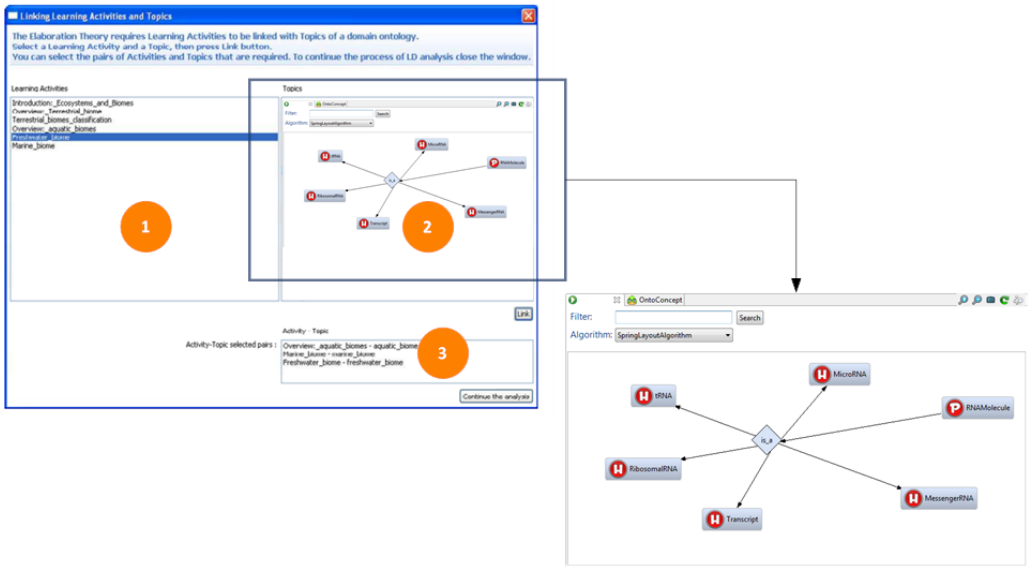

Figura 4.11: Herramienta Terpsícore y su vinculación con *OntoConcept_Graph* - Nueva pantalla de Terpsícore que permite definir la vinculación entre Actividades de Aprendizaje y Tópicos de Aprendizaje de la ontología de dominio,

En esta figura se aprecian también tres secciones de interacción con el diseñador instruccional dentro de la pantalla de Terpsícore:

1. Corresponde a la lista de Actividades de Aprendizajes del LD, los cuales deben ser vinculados con los Tópicos de Aprendizajes para su posterior análisis de conformidad con la Teoría Elaborativa.

2. Corresponde a la lista de Tópicos de Aprendizajes de la ECC ahora implementada por medio de *OntoConcept* e instanciada de manera automatizada en el proceso TGS. Además, se representa gráficamente la ECC dentro del entorno de Terpsícore, de manera que sea posible vincular de forma más amigable al diseñador y con mayor flexible debido a la incorporación de una barra de herramientas inteligente, la cual incluye:

   - filtros de búsqueda de tópicos
   - ordenamientos de la ECC
   - capturas de imagen en formato digital

3. Permite vincular cada Actividad con su tópico dispuesto en la representación gráfica. Se realiza por cada vinculación.

4.6 Conclusiones del Capítulo.

En este capítulo se han presentado los principales aspectos de la implementación para la representación de una ECC automatizada, a través de una aplicación que permite generarla mediante los análisis taxonómicos en ontologías de dominio. En relación a esta implementación, se logro obtener representaciones interoperables lo que posibilita que puedan ser utilizadas por diversas aplicaciones, entre ellas, Terpsícore. Las últimas secciones de este capítulo comprenden el proceso de integración de la aplicación realizada dentro del entorno de Terpsícore. En esta, se precisa el tratamiento para lograr su vinculación y cuyo resultado comprende la representación en su totalidad de la Teoría Elaborativa. Cabe destacar que la aplicación *OntoConcept* se encuentra en su primera versión y aunque cumple con el objetivo de demostrar la factibilidad de representar una ECC a partir de diversas ontologías .OWL según precisa la teoría descrita, su integración con Terpsícore posibilita el éxito de esta investigación.

4. IMPLEMENTACIÓN DE LA SOLUCIÓN

5

Líneas de trabajo futuras

5.1 Introducción

Durante esta sección se plasman las directrices que dan origen a trabajos futuros. Estas apuntan a: El perfeccionamiento de la representación de la ECC por medio de herramienta *OntoConcept* y a la proyección que sostendrá esta la investigación en el tiempo.

5.2 Trabajo futuros en relación al perfeccionamiento de la ECC

Estos apuntan a dos direcciones: al perfeccionamiento de la representación de la Estructura de Conocimiento Conceptual y a la construcción de sistemas que den soporte a la Teoría Elaborativa utilizando la ECC como base de conocimiento. En relación al primer punto, para el mejoramiento de la representación que actualmente forma la ECC podría incluir:

- Generar un tratamiento para aquellas ontologías que utilizan la etiqueta label <*rdfs:label*> en su confección para identificar el nombre nombres de clases, relaciones y propiedades. Por consiguiente, se sugiere su modelación de manera que pueda ser aplicada a *OntoConcept*.

- En *OntoConcept*, la selección de conceptos de una ontología es explícita, de otro modo el proceso de búsqueda de estos dentro de la aplicación no sería satisfactoria.

Sin embargo, se sugiere el tratamiento de sinonimias entre otras técnicas, de manera que sea posible lograr filtros menos rígidos y basados en una semejanza de significados entre dos o más conceptos.

- Muchas ontologías poseen una distancia conceptual de clases desmesurada, por lo que se sugiere un tratamiento que permita administrarlas con facilidad, de manera que el diseñador instruccional pueda seleccionarla según estime pertinente, ya sea para visualizarla gráficamente o para generar una ECC acotada a una distancia conceptual precisa.

5.3 Proyección de la investigación

El presente trabajo sostiene dos lineamientos respecto a su proyección en el tiempo: En primera instancia, se trabajará conjuntamente en la evaluación de *OntoConcept* dentro del entorno de la herramienta *Terpsícore* y en forma integrada con su diseñador. Este trabajo se efectuará en los próximos meses como parte del Proyecto de Investigación: "Uso de ontologias formales en Diseño Instruccional"código: 125512 3/RS, del Departamento de Investigación de la Universidad del Bío-Bío,[1]cuyo responsable es el profesor Christian Vidal Castro.

Por otro lado, a partir de los resultados generados en esa investigación, se está trabajando en la generación de un artículo científico que será enviado a una revista relacionada con la temática de: *ontological engineering and knowledge representation.*

5.4 Conclusiones del Capítulo

En este capítulo se han presentado las principales proyecciones que sostiene esta investigación, así como la trascendencia que esta generó. En el primero de los casos, se vislumbran posibles investigaciones dentro del área de representación de conocimiento y el perfeccionamiento de la aplicación, incluyendo una evaluación durante los próximos meses, como parte de un proyecto de investigación de la Universidad del Bío-Bío.

Por otro lado, en relación a su trascendencia, se precisa el alcance que ha tenido este trabajo y como este ha dado cabida a un artículo científico actualmente en desarrollo y

[1]Para mayor información visitar: `http://www.dgi.ubiobio.cl/`

que a mediano plazo será presentado en conjunto al diseñador de *Terpsícore*, el profesor Christian Vidal Castro.

5. LÍNEAS DE TRABAJO FUTURAS

6

Conclusiones

6.1 Introducción

En los capítulos anteriores se presentaron los objetivos y la problemática asociada a esta investigación, además, se realizó un marco teórico que concentra los principales tópicos que envuelven este trabajo. Luego, se realizó un enfoque del modelado general utilizado para construir la representación de la ECC y a posteriori a esta modelación se plasmaron aspectos de la implementación además del proceso de evaluación la propuesta de esta tesis.

En este capítulo se muestran las conclusiones obtenidas a partir de la realización de este trabajo. En primera instancia se verifica el cumplimiento de los objetivos y de la hipótesis de esta investigación. Luego, se presentan las conclusiones generales, las aportaciones y las líneas de trabajo futuro que deja abierta esta investigación. Además, se presenta la publicación en desarrollo durante la presente investigación.

6.2 Verificación de los Objetivos y la Hipótesis

En el Capítulo 1, específicamente en la sección 1.4, se definieron los objetivos específicos que en conjunto dan forma al objetivo general de esta tesis: desarrollar una herramienta, utilizando un lenguaje con semántica computacional que analiza las relaciones ontológicas de una ontología de dominio, generando una ECC a partir del marco de la Teoría de Elaboración de manera que sea posible dar soporte al DI.

A continuación, para cada uno de los objetivos específicos, definidos también en la sección 1.4, se presentan los elementos que permiten verificar el cumplimiento de estos:

1. **Analizar la estructura y relaciones lógicas (*is_a*, *part_of*) de una ontología de dominio OWL:** En el Capitulo 2, sección 2.2.4 se presentó un marco teórico, sobre el cual se analizaron las relaciones taxonómicas de ontologías de dominio OWL, particularmente las relaciones de tipo y de partes, *is_a* y *part_of* respectivamente. En primera instancia, se analizó la relación de tipo enfatizando la relevancia de la jerarquía de clases que compone esta relación. En segunda instancia, se analizó las relaciones de composición y la particularidad que presenta esta en relación a cómo definir que un contenido(clase) sea parte de otro. Además, se destacaron las propiedades de transitividad, reflexividad y anti-simetría que precisan estas relaciones. Ambas relaciones son apoyadas por ilustraciones que plasman su estructura.

2. **Analizar la Teoría de Elaboración, dentro del contexto de Teorías de Diseño Instruccional:** Dentro del marco teórico del capítulo 2, la sección 2.4 define el contexto general para la aplicación sistemática de los principios y TDI que rigen el diseño de aprendizaje. Dentro de estas últimas, se enfatiza en la Teoría Elaborativa (Sección 2.4.1), bajo el alero de ser un apoyo a la selección de contenidos a aprender y proporcionando formas de secuencias para los tópicos de manera de fomentar logros instruccionales. El análisis de esta teoría es holgado, comprendiendo las diferentes secuenciaciones de tópicos y desatancándose la particularidad que precisa respecto al recorrido de tópicos que van desde lo más general a lo particular y jerárquicamente compuesto solo por relaciones de tipo y de composición. Esta cualidad se denomina Estructura de Conocimiento Conceptual y es precisamente la base que permite vislumbrar su modelación computacionalmente a partir de contenidos dispuestos en ontologías generándose partir del análisis taxonómico de estas.

3. **Estudiar la API Protégé OWL para el manejo de ontologías OWL sobre el lenguaje Java:** El manejo de lenguajes ontológicos se hace fundamental para realizar el propósito de esta investigación. En este sentido, se hace relevante estudiar cada lenguaje existente con la idea de poder seleccionar aquel que más se acerque al propósito de esta tesis. Es este sentido, basado en el número de ontologías utilizadas dentro del campo de E-learning, el lenguaje OWL, es el más indicado para esta. En virtud de lo anterior, la sección 2.2.3 del Capítulo 2, realiza

un análisis exhaustivo de las principales taxonomías del lenguaje, asociando estas a su representación modeladas a través de la API Protege OWL para el manejo de ontologías OWL sobre el lenguaje RCP Eclipse de Java

4. **Construir una aplicación que genere una ECC a partir de una ontología de dominio utilizando un framework de ontología y RCP de Eclipse:** El primer paso para la construcción del prototipo fue la propuesta de modelación de las relaciones tanto tipo, *is_a*, como la de composición, *part_of* (capitulo 3), vislumbrando su modelamiento mediante el lenguaje ontológico OWL. Un segundo paso, fue la implementación de este modelamiento con una semántica computacional (Capitulo 4), utilizando el framework para el tratamiento ontológico RCP de Eclípse. De manera similar a la modelación, en primera instancia se implementó la propuesta para la representación de la relación de tipo *is_a* y luego, basado en esta, la relación de composición, *part_of*. Esta aplicación, *OntoConcept*, permite generar a partir de una ontología de dominio cualquiera, una ECC tal cual precisa la Teoría Elaborativa.

   Este prototipo de software funciona como parte del entorno del editor Recourse, compatible con la especificación IMS-LD. Esta aplicación se proyecta como parte de la herramienta Terpsícore, para dar soporte a la Teoría de Elaboración y aplicar esta para la evaluación en conformidad con un LD.

5. **Integrar la herramienta a la aplicación Terpsícore, a través de un entorno grafico generado por el framework GEF, permitiendo la interacción del usuario para dar sustento a la Teoría de Elaboración por medio de la vinculación de un LD con una ontología de dominio tratada taxonómicamente como un ECC:** La integración con Terpsícore es la base para dar conformidad a la Teoría Elaborativa. En este sentido, es esencial comprender el proceso de integración de *OntoConcept* dentro del entorno de Terpsícore. Bajo esta premisa, en el capitulo 4 (4.5.1) se estudia en detalle dicha implementación, destacando la arquitectura técnica utilizada así como los pasos para generar la vinculación.

Finalmente, puesto que todos los objetivos específicos han sido logrados, puede considerarse que el objetivo general también ha sido alcanzado. De esta forma, la

Hipótesis de la tesis definida en la Sección 1.3, considerando los objetivos alcanzados, los resultados obtenidos y el método de investigación seguido, puede considerarse como verificada.

Por lo tanto, puede afirmarse que:

Es posible generar, una Estructura de Conocimiento Conceptual que precisa la Teoría de Elaboración mediante un lenguaje con semántica computacional por medio del análisis de relaciones taxonómicas de una ontología de dominio.

6.3 Conclusiones del trabajo

La Teoría de Elaboración propuesta por Charles Reigeluth, ofrece un soporte explicito de cómo ayudar a las personas durante un proceso de aprendizaje, facilitando la selección de contenidos a aprender a través de la particularidad que precisa en su Estructura de Conocimiento Conceptual, definiendo la jerarquización de los tópicos de un tema cualquiera, desde lo más general a lo particular(proceso intuitivamente más utilizado en el proceso de formación) y compuestos solo por relaciones de tipos y de partes. Sin embargo esta teoría está expresada en un lenguaje natural, por lo que disponer de esta en un lenguaje con semántica computacional, permitiría construir herramientas que bajo el alero de esta teoría, apoyen el proceso de Diseño Instruccional. Esta investigación marca un precedente al modelar una ECC por medio de un lenguaje ontológico formal, el cual está plasmado en la herramienta: *OntoConcept* y su extensión grafica *OntoConcept_graph*, las cuales permiten el tratamiento de ontologías de dominios y transformarlas por medio del análisis de las relaciones taxonomías en una que cumpla con los criterios de una ECC. Sin embargo esta representación no está exenta de contrariedades, puesto que la teoría además exige que sea necesario mantener la atención del alumno, cualidad que no es posible representar actualmente, debido a la ambigedad que comprende. No obstante, a futuro podría ser necesaria la utilización de calificadores provenientes de la Lógica Difusa que soporten la ambigedad de conceptos utilizados por la Teoría de Elaboración expresada en lenguaje natural. Por otro lado, en relación análisis taxonómico que precisa la aplicación, el tratamiento de ontologías presenta problemas debido a la diversidad de lenguajes utilizados en su confección, careciendo de una estandarización en su construcción que limita a la aplicación a dar

sustento a aquellas que estén solo construidas en lenguaje OWL y con una densidad de clases computables.

A pesar de las contrariedades, la aplicación *OntoConcept* cumple fielmente con su objetivo, sugiriendo además la vinculación con la herramienta Terpsícore. Dicha integración, permite obtener una representación formal de la Teoría Elaborativa.

Resulta relevante mencionar que los profesores optan por diversas teorías, siendo la elaborativa la más recurrente pero no la única, puesto que algunos sugieren otras modalidades de formación que se adaptan más fácilmente a otras teorías que apoyan más al logro instruccional.

En conclusión, se cree que la aceptación de la representación de una ECC se cumple a cabalidad, más aún en su integración con Terpsícore, cuyo resultado ha permitido el desarrollo de un artículo científico, el cual prontamente será enviado a una revista relacionada con la temática de ingeniería ontológica y representación del conocimiento, como parte del proyecto "Uso formales de ontologías en el Diseño Instruccional" a cargo del Dr. Christian Vidal Castro. Sin embargo, cabe resaltar que en relación a la presente investigación existen diversas aristas abiertas a futuras investigaciones, las cuales han sido detalladas y sugeridas de manera que puedan continuar siendo aportes para generar logros instruccionales.

6.4 Aportaciones

Esta investigación procura aportar al área del E-learning, en particular al Diseño Instruccional mediante el modelado de una ECC la que en conjunto con Terpsícore pueden ser utilizadas para asistir profesores y diseñadores instruccionales que no poseen experiencia en la aplicación de la Teoría Elaborativa o que precisan su uso correctamente. En esta sección se explican las aportaciones que realiza a este trabajo, dividida en dos grupos: aportaciones teóricas y prácticas.

6.4.1 Aportaciones teóricas

Las aportaciones teóricas se relacionan con la construcción de un modelo para representar una ECC en lenguaje computable. lo que hasta el momento no existe en la literatura. Esta representación se caracteriza por:

- Esta expresada en un lenguaje con semántica computacional. La utilización del lenguaje ontológico formal OWL, permite asegurar la computabilidad de la representación por parte de aplicaciones basadas en la representación del conocimiento.
- La vinculación con Terpsícore permite su relación con la especificación IMS-LD, permitiendo la interoperabilidad y reúso con otras aplicaciones que utilicen dicha especificación y que requieran del conocimiento relacionado con la Teoría Elaborativa y su ECC.

6.4.2 Aportaciones prácticas

Dentro del marco de las aportaciones prácticas, estas se relacionan con la implementación de la ECC y los beneficios que proporciona su utilización por sistemas que apoyen actividades relacionada con la Teoría Elaborativa. Por otro lado, otro aporte práctico se refiere a la construcción de una aplicación denominada *OntoConcept*, que permite generar una ECC a partir del análisis taxonómico de una ontología robusta de dominio. *OntoConcept* funciona integrado al entorno del editor ReCourse, conocido y utilizado por la comunidad de E-learning para el diseño de LD. Esta aplicación, que ha sido evaluada en términos de su utilidad y usabilidad por diseñadores instruccionales, dispone de importantes proyecciones para ser considerada en versiones futuras, como una herramienta de apoyo a diseñadores instruccionales y profesores.

Además, la vinculación de *OntoConcept* con la herramienta Terpsícore, la cual también se encuentra integrada a ReCourse, permite en conjunto dar soporte a la Teoría Elaborativa. Esta vinculación ha sido evaluada en términos de utilidad y usabilidad por expertos en DI y por el mismo diseñador de Terpsícore. El hecho que ambas aplicaciones hayan sido construidas bajo licencia LGPL, permiten su libre distribución, modificación y uso, posibilitando su desarrollo como herramienta de E-learning y ser un aporte concreto a dicha área así como a la representación del conocimiento en general.

Referencias

[1] GRUBER, T.(1993). *A translation approach to portable ontology specification.* Knowledge Acquisition, 5(1993), 199-220. 10

[2] BORST, W. N. (1997). *Construction of Engineering Ontologies, PhD Thesis.* University of Twente, Enschede. 10

[3] SOWA, J. F. (1999). *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Pacific Grove, CA, Brooks Cole Publishing Co. 10

[4] VAN HEIJST, G., SCHEREIBER, A.T., WIERINGA,B.J. (1996). *Using Explicit Ontologies in KBS.* Development. Internacional Journal of Human and Computer Studies. 11

[5] DEVEDZIZ, V. (2006). *Semantic and education.* Springer's Integrated Series in Information Systems. 11

[6] *Dictionary of XML Technologies and the Semantic Web.* Springer professional computing. 12

[7] HORROCKS, I., PATEL-SCHNEIDER, P., McGUINNESS, D., & Welty, C. (2007). *OWL: a Description Logic Based Ontology Language for the Semantic Web. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi & P. Patel-Schneider (Eds.).* The Description Logic Handbook: Theory, Implementation, and Applications (2nd Edition): Cambridge University Press. 17

[8] GROSSO, E., ERIKSSON, H., FERGERSON, R. W., TU, S. W., & MUSEN, M. M. (1999). *Knowledge modeling at the millennium - the design and evolution Protégé-2000.* The 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99), Banff, Canada. 17

[9] *An Introduction to RDF(s) and a Quick Tour of OWL.* The Univesity of Manchester, `http://www.co-ode.org/resources/tutorials/intro/slides/OWLFoundationsSlides.pdf` [consulta: 09 enero 2013] 20, 21, 22, 23, 24

[10] CAVERO BARCA, J.M., VELA SANCHEZ, B., MARCOS MARTINEZ, E. (2005). *Aspectos filosóficos, psicológicos y metodológicos de la informática.* Universidad de Rey Juan Carlos. 27

[11] HERNANDEZ, H., & SAIZ, M. (2007). *Ontologías mixtas para la representación conceptual de objetos de aprendizaje.* Procesamiento del Lenguaje Natural, Revista n 38 (Abril, 2007). 29

[12] MARENGO, A., ALBANESE, D., CONVERTINI, N., MARENGO, V., SCALERA, M., & SERRA, A. (2006). *Ontological support for the creation of learning objects.* Paper presented at the 28th International Conference on Information Technology Interfaces, 2006. 29

[13] GUARINO, N. (1998). *Formal Ontology and Information Systems.* Amended version of a paper appeared in N. Guarino (ed.), Formal Ontology in Information Systems, Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998, Amsterdam, IOS Press, pp. 3-15. 11

[14] GOMEZ-PEREZ, A., FERNANDEZ-LOPEZ, M., CORCHO, O. (2003). *Ontological Engineering.* Springer Verlag. 2003. 11

[15] REIGELUTH, C. (1999). *Instructional-Design Theories and Models, Volume II:A New Paradigm of Instructional Theory.* Mahwah, NJ: Lawrence Erlbaum Assoc. 1, 2, 30, 31, 33, 40, 51

[16] BEAUVOIR, P., GRIFFITHS, D., & SHARPLES, P. (2009). *Learning Design Authoring Tools in the TENCompetence Project In R. Koper(Ed.).* Learning Network Services for Professional Development(pp. 379-387): Springer Berlin Heidelberg. 2

[17] BURGOS, D. (2008). *Extension of the IMS Learning Design Specification based on Adaptation and Integration of Units of Learning .* Tesis Doctoral en Ingeniería Informática, Escuela Politécnica Superior. 44

[18] GUERRERO, A. (2011). *L'especicació IMS-LD per a la descripció formal d'itineraris formatius adaptatius.Tesi Doctoral. Programa de Doctorat en Societat de la Informació i el Coneixement.* Universitat Oberta de Catalunya, Barcelona. 42

[19] SANCHEZ, E., & LAMA, M. (2007). *Técnicas de Inteligencia Artificial Aplicadas a la Educación.* Revista Iberoamericana de Inteligencia Artificial, 33, pp. 7-12. 42

[20] POLSANI, P. (2003). *Use and Abuse of Reusable Learning Objects.* Journal of Digital Information. 43

[21] LTSC (2002). *IEEE Learning Object Metadata (LOM), Draft Standard 1484.12.1, Learning Technology Standards Committee, (2002).* 43

[22] McGREAL, R. (2004). *Learning Objects: A practical Definition.* International Journal of Instructional Technology and Distance Learning., Vol. 1, pp 21 - 32. 43

[23] MILLAR, G. (2002). *Learning objects 101: A primer for neophytes.* BCIT Learning and Teaching Centre. 43

[24] BRUNER, J (1960). *The Process of Education.* Cambridge, Mass.: Harvard University Press. 97 + xxvi pages. 34

[25] AUSUBEL, D.P. (1968). *Educational Psychology: A Cognitive View. New York: Holt.* Rinehart & Winston. 34

[26] VIDAL-CASTRO, C., SEGURA, A., SICILIA, M-A., PRIETO, M. (2012). *Terpsicore: an Instructional Design Tool.* SPDECE-2012. Multidisciplinary symposium on the design and evaluation of digital content for education, Universitat d' Alcant & Universidad de Alicante. 2, 6

[27] VIDAL-CASTRO, C. (2011). *Uso de ontologías formales para el soporte al diseño instruccional.* Tesis Doctoral, Universidad de Castilla la Mancha. 2, 44, 53, 63

[28] W3C. (2004). *OWL Web Ontology Language Reference.* World Wide Web Consortium. Retrieved from `http://www.w3.org/TR/owl-guide/`. 12

REFERENCIAS

[29] Brickley, D. & Guha, R.V.(1999). *Resource Description Framework (RDF) Schema Specification. Proposed Recommendation.* World Wide Web Consortium: http://www.w3.org/TR/PR-rdf-schema. 28

[30] Hendler, J. & McGuinness, D.L.(2000). *The DARPA Agent Markup Language.* IEEE Intelligent Systems 16(6): 67-73. 28

[31] Price, C. & Spackman, K.(2000). *SNOMED clinical terms.* BJHCIM-British Journal of Healthcare Computing & Information http://www.bj-hc.co.uk/. 28, 75

[32] Humphreys, B.L. & Lindberg, D.A.B.(1993). *The UMLS project: making the conceptual connection between users and the information they need.* Bulletin of the Medical Library Association 81(2): 170. 28, 75

[33] VIDAL-CASTRO, C., SICILIA, M-A., PRIETO, M.(2012). *Representing instructional design methods using ontologies and rules.* Knowl.-Based Syst. 33: 180-194 (2012). 6, 101