An Ontology-based Question-answering Framework for Adaptive E-Learning Systems

By

Sudath Rohitha Heiyanthuduwage

A Thesis

Presented to

Macquarie University

in fulfilment of requirements

for the degree of

Doctor of Philosophy

in

Computer Science

Department of Computing

Faculty of Science and Engineering

Sydney, Australia

To my wife and two daughters

Abstract

Current e-learning systems are based on legacy database systems. Databases focus on logically organising data, but not on conceptual knowledge. Therefore, current e-learning systems have limitations in retrieving domain knowledge from a legacy database and benefiting from the full expressive capabilities which an ontology-based approach would be able to provide. In order to alleviate this problem, we propose to augment current e-learning systems with Semantic Web Technologies. Furthermore, different educational institutions have similarities, yet an analysis we conducted on subject descriptors and course handbooks from a set of institutions elicits the terminological differences they use. Such differences hinder deploying an e-learning system in multiple institutions.

We propose a twofold intuitive solution. Firstly, we propose an *ontology-based plug and play architecture* for developing an e-learning system's framework, instances of which can be deployed at different institutions by plugging in institution-specific learning ontologies. Secondly, we propose institution-specific learning ontologies for representing the knowledge and terminology specific to each institution.

Developing institution-specific ontologies duplicates the effort of an ontology engineer. We analyse a corpus of learning ontologies developed for the learning domain. The results of our analysis show that the language constructors used in the corpus of the learning ontologies belong to a sublanguage of the ontology language OWL 2, that we name as OWL 2 Learn profile. We demonstrate how we use the OWL 2 Learn profile as a guide to developing an institution-specific ontology and populating it by mapping the data available in a learning database into instances and properties in an ontology.

We develop an ontology benchmark query suite for evaluating learning ontologies for their query-answering and inference capabilities. We use the benchmark query suite to evaluate the sample learning ontologies that validate the expressivity of OWL 2 Learn profile as well. The adaptability of the e-learning system's framework is evaluated using a proof of concept prototype. We demonstrate how we can use two instances of the prototype with institution-specific ontologies for query-answering that generates similar answers, yet with domain specific terminology.

This thesis shows how we can assist the development of ontology-based e-learning systems using Semantic Web technologies. We anticipate that this research would give some directions in developing context specific ontology-based e-learning systems.

Acknowledgements

Many people have been a great help during the journey of my doctoral studies and I take the opportunity to thank them all. I am extremely grateful to my principal supervisors, Professor Mehmet A. Orgun and Dr. Rolf Schwitter for their continuous guidance and support during my doctoral studies. Without their support this work would have not been a reality.

I would like to thank Macquarie University for funding my doctoral studies. The financial support from the University has been an immense help. I also would like to thank the academic and administrative staff at the Department of Computing for helping me in numerous ways.

I extend my gratitude to my employer, Study Group Australia (Charles Sturt University Study Centre, Sydney) and the management of the CSU Study Centre, Sydney, for supporting my doctoral studies and considering it as a part of my professional development.

My deepest gratitude goes to my wife, Jeewanthi, for her continuous patience, love and encouragement in difficult times during my studies. Also, I extend my sincere gratitude to my two loving daughters, Dulya and Deana, for their sacrifices and for the inspiration they have been to me.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed) Date: _____07/06/2018_____

Sudath Rohitha Heiyanthuduwage

List of publications

My thesis research has resulted in one journal paper and several refereed conference papers. My contribution to the papers listed below is 80%.

Journal Paper:

Heiyanthuduwage, S. R., Schwitter, R., & Orgun, M. A. (2016a). OWL 2 learn profile: an ontology sublanguage for the learning domain. *SpringerPlus, 5(1)*, 291.

Conference Papers:

Heiyanthuduwage, S. R., Schwitter, R., & Orgun, M. A. (2013). An adaptive learning system using plug and play ontologies. *IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)* (pp. 429-434). IEEE.

Heiyanthuduwage, S. R., Schwitter, R., & Orgun, M. A. (2014). Towards an OWL 2 profile for defining learning ontologies. *IEEE 14th International Conference on Advanced Learning Technologies (ICALT)* (pp. 553-555). IEEE.

Heiyanthuduwage, S. R., Schwitter, R., & Orgun, M. A. (2016b). A learning ontology with metadata and user profiles for enhancing accessibility of resources. *IEEE Conference on e-Learning, e-Management and e-Services (IC3e)* (pp. 85-90). IEEE.

Table of Contents

List of Figures

List of Tables

Acronyms

ABox	Assertional Box
AI	Artificial Intelligence
AL	Attributive Language
ALC	Attributive Language with Complement
AO	Activity Ontology
AOMS	Advanced Ontology Management Systems
BK	Knowledgebase
CB	Content Based
CF	Collaborative Filtering
CPO	Class Profile Ontology
CSU	Charles Sturt University
CWA	Closed World Assumption
DBMS	Database Management System
DL	Description Logic
DPE	Data Property Expression
EIS	Educational Information System
ELO	E-Learning Objects
FL	Frame Language
FOAF	Friend of a Friend
FOL	First Order Logic
FSLSM	Felder-Silverman Learning Style Model
HERO	Higher Education Reference Ontology
IEEE	Institute of Electrical and Electronics Engineers
IMSLD	IMS Learning Design
IPO	Instructor Profile Ontology
IWT	Intelligent Web Teacher
KBS	Knowledgebase System
KR	Knowledge Representation
LCMS	Learning Content Management Systems
LD	Learning Domain
LOM	Learning Object Metadata
LPO	Learner Profile Ontology
MCQ	Multiple Choice Questions

MO	Material Ontology
MQ	Macquarie University
NGEE	Next Generation Education Environments
nRQL	new Racer Query Language
OBAA	Agent Based Learning Objects
OBDA	Ontology-based Data Access
OPE	Object Property Expression
OWA	Open World Assumption
OWL	Web Ontology Language
R2RML	RDB to RDF Mapping Language
RDB	Relational Database
RDBMS	Relational Database Management System
RDCEO	Reusable Definition of Competency or Educational Objective
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDOTE	Relational Databases to Ontologies Transformation Engine
RLO	Reusable Learning Objects
RS	Recommendation Systems
SCORM	Sharable Content Object Reference Model
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SW	Semantic Web
TBox	Terminological Box
TMQL	Topic Map Query Language
UICO	User Interaction Context Ontology
UML	Unified Modelling Language
URL	Unified Resource Locator
W3C	World Wide Web Consortium
WBES	Web-based Educational Systems
WBIS	Web-based Educational Systems
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Mark-up Language

Each new chapter begins on an odd page.

Chapter 1: Introduction

E-learning, simply electronic learning, has been a popular research area in over last two decades. As the name itself implies, it involves the use of electronic media to support the learning and teaching tasks. E-learning has been defined as:

> '*…all forms of electronic supported learning and teaching, which are procedural in character and aim to effect the construction of knowledge with reference to individual experience, practice and knowledge of the learner. Information and communication systems, whether networked or not, serve as specific media (specific in the sense elaborated previously) to implement the learning process*' (Tavangarian, Leypold, Nölting, Röser, & Voigt, 2004).

Among the other e-learning technologies, relational database management systems (RDBMS) have been used over many years to facilitate storage and retrieval of data related to the learning and teaching tasks. Distributed database systems have been suggested for institutions which operate in multiple locations to provide fast access and collection of data (Magdalena, 2011). In most case, the motivation behind the use of databases has been simple. Database systems store larger amounts of data than the primary memory size of a personal computer can accommodate. Database systems offer controlled access, security, recovery, integration and sharing of data (Sir, Bradac, & Fiedler, 2015).

Internet and World Wide Web (WWW) are two main driving forces of e-learning. The WWW has achieved a massive growth from its invention by Tim Berners-Lee in 1989 and it has changed the way people do business, communicate, and study. The WWW is an interconnected set of web pages (different documents and resources) that are identified by Unified Resource Locators (URLs), interconnected by hyperlinks and accessed via the Internet (Berners-Lee & Fischetti, 2001). The Semantic Web (SW) is a further step forward of the WWW. The Semantic Web is considered as a web of actionable information (Shadbolt, Berners-Lee, & Hall, 2006). The information can be derived from data and included in the web pages. The semantic theory gets the meaning of the logical connections between the terms that would allow the interoperability between systems (Shadbolt et al., 2006). The Semantic Web shows the connections between the data and information, instead of web

pages. This integration of heterogeneous data in a system or different systems is achieved through common conceptualisations known as ‘ontologies’ (Shadbolt et al., 2006).

Ontologies that represent the domain knowledge have been used as a main part of the Semantic Web. Ontologies are required to augment legacy database systems. Ontologies help to prevent and resolve communication issues among heterogeneous systems, enhance fusion of information and sharing knowledge. Also, ontologies help to integrate information, increase the interoperability of heterogeneous information sources by providing a high level, abstract view of information (Sir et al., 2015). The Ontology languages, Web Ontology Language (OWL) and OWL 2 are based on the Description Logics (DLs). DLs are formal knowledge representation languages that have been elaborated in Baader, Calvanese, McGuinness, Nardi, and Patel-Schneider (2003). DLs enhance the accuracy of the domain knowledge and support reasoning on them.

E-learning has been benefited by the Semantic Web technologies in different ways. We notice a gradual growth in the e-learning applications which are based on the Semantic Web technologies. These applications have attempted to model the domain knowledge of learning and provide access to the users to assist their learning and teaching. Some applications integrate metadata into Semantic Web based e-learning applications. The use of multi layered Learning Object Metadata (LOM) as an approach for searching for learning objects has been proposed, in (Hsu, 2012). On the other hand, some applications focus on using user profiles and personalisation. For example, learning patterns of the learners are analysed based on user profiles, ontologies and reasoning, in Nafea, Maglaras, Siewe, Smith, and Janicke (2016). Some e-learning applications have been integrated with other technologies such as Cloud databases. For example, an ontology-based adaptive personalised e-learning system that is based on Semantic Web and Cloud technologies has been proposed in (Rani, Nayak, & Vyas, 2015).

Many e-learning systems that are based on Semantic Web technologies have focused on achieving an adaptivity of e-learning systems at the user level. For example, Rani et al. (2015) and Gascueña, Fernández-Caballero, and González (2006) have focused on ontology-based e-learning systems to adapt the learning contents at the user level based on user profiles and learning styles. However, they do not address the adaptivity of e-learning systems to different learning contexts or institutions. A communication ontology has been proposed in (Aroyo & Dicheva, 2004) to achieve interoperability of e-learning systems and institution

level adaptability of e-learning systems by overcoming the problem of not having a common reference architecture.

1.1 Motivation

Current e-learning systems use database systems primarily as storage of learning resources and granting secure access to those resources. However, current e-learning systems are poor in conceptualising and managing the domain knowledge. However, the SW technologies have been suggested to provide the high level conceptual view of the data and information in the web application. Hence, we expect SW technologies to be a perfect candidate for augmenting e-learning systems that are based on legacy database systems.

As a step forward in augmenting current e-learning systems, we conducted an initial analysis of the unit guides of over ten institutions in the state of New South Wales, Australia. This analysis unveiled that there are differences in the terminologies used in those institutions. For example, Table 1.1 lists a part of the terminology used at MQ and CSU.

Table 1.1 : Different terminology used at MQ and CSU

Some Terms used at MQ	**Some Terms used at CSU**
Unit	Subject
Unit description	Subject outline
Course	Major / Specialisation
Convener	Subject coordinator
Topic list	Schedule
Assessment task	Assessment item
Learning resource	Learning material
Department	School

We also noticed that currently each institution's e-learning systems have been developed considering their specific requirements. Hence, the terminology used in an institution has been applied in all parts of an e-learning system. This leads to duplicated effort in developing e-learning systems. Getting consensus on a common terminology for all institutions is a tedious task. Each institution likes to keep their uniformity to compete with other institutions. This hinders deploying an instance of the same e-learning system in different institutions.

There are attempts by some scholars to align the terminologies used in each institution by considering synonyms and antonyms. They specially focus on sharing the knowledge in different institutions. However, that does not solve the problem of duplicated effort the institutions are expending in developing individual e-learning systems. An e-learning systems

architecture that is commonly applicable to many institutions would help to alleviate this problem. An ideal solution would be to allow institutions to use instances of the same e-learning system while using their own terminology and institution-specific knowledge.

An institution's specific knowledge is distributed in many parts of an e-learning system. Domain knowledge is identified by terminology specific to an institution. This makes e-learning systems further specific to an institution and duplicates the effort required to develop them. Therefore, we need a way represent the domain knowledge of an e-learning system as a separate component of it. Also, this would help us to separate an institution's specific domain knowledge using a specific terminology of an e-learning system. That would help to augment e-learning systems and overcome the duplicated effort in developing them.

Furthermore, if we could separate the institution-specific application knowledge from the domain knowledge of learning that would help to deploy the domain knowledge at institutions with application knowledge relevant to an institution's specific terminology.

1.2 Research Problem

In our research, we address the research problem of how to use SW technologies to augment current e-learning systems while keeping the terminological differences between the institutions. As a way to solve this problem, we attempt to address the research in two ways:

1. How to use SW technologies to design an e-learning system, instances of which could be deployed in different institutions.
2. How to separate domain knowledge of the learning domain from an e-learning system and reuse it in institution-specific e-learning systems.

1.3 Ontology and Ontology Languages

Ontology has been defined in different ways over time. Most popular definition "*An ontology is an explicit specification of a conceptualization*", is given in (Gruber, 1993). Again, it is stated that *"ontologies provide a shared and common understanding of a domain that can be communicated between people and across application systems"*, in (Horrocks et al., 2000).

An ontology is formally specified using an ontology language. A set of essential requirements of an ontology language have been listed in (Fensel, Van Harmelen, Horrocks, McGuinness, & Patel-Schneider, 2001). They include: having a compact syntax, a well-defined formal semantics, the potential for building knowledge bases, a proper link with existing web

standards to ensure interoperability; and, being highly intuitive to humans, being able to represent human knowledge including reasoning properties.

Different ontology languages have been proposed over the last two decades. The results of a survey on some of the early ontology languages are in (Pulido et al., 2006). These include KIF (Genesereth & Fikes, 1992), F-Logic (Kifer, Lausen, & Wu, 1995), CycL, Dublin core (Baker, 2000). The ontology languages: XML, SHOE, DAML+OIL, RDF, RDFS, OWL and its sublanguages have also been introduced. However, the current standard ontology language recommended by the World Wide Web Consortium (W3C) is OWL 2 (Motik et al., 2012). In subsections below, we introduce recent ontology languages that are more relevant to our work: RDF/RDFS, OWL and OWL 2.

1.3.1 Recent Ontology Languages

RDF/RDFS – The Resource Description Framework (RDF) was accepted by the W3C in 2004 as a framework for describing resources on the Web (Cyganiak et al., 2014). It has been a great foundation for the current ontology languages. However, RDF does not include sufficient constructors to specify a comprehensive ontology. RDF Schema (RDFS), a schema language for RDF, provides a framework for describing application-specific classes and properties (Fensel et al., 2001). RDFS is an extension of RDF and it includes additional language constructors to specify relations between concepts.

OWL – superseded RDF/RDFS in 2004 as a Web ontology language and became a W3C recommendation for the Semantic Web. As OWL is based on a version of description logic, it allows the use of a DL-based reasoner to derive information that is not explicitly specified in an OWL ontology (Horrocks & Patel-Schneider, 2011). OWL included many new language constructors to specify domain knowledge. OWL also includes features of RDF, DLs and frames (Horrocks, Patel-Schneider, & Van Harmelen, 2003).

OWL 2 – has been used as the W3C recommended ontology language for the Semantic Web (Motik et al., 2012). OWL 2 is a new and more expressive version of OWL which mainly improves the relational and datatype expressivity of the language. OWL 2 includes additional language constructors than OWL and includes three (3) standard sublanguages or profiles.

1.3.2 OWL 2 Standard Profiles

Over the past two decades the research on DL has focused on increasing the expressivity of an ontology language. It has led to developing highly intractable DL languages like OWL

DL. As a way of dealing with this problem two sublanguages of OWL DL: OWL EL and OWL Lite have been introduced (Baader & Lutz, 2010). Later, these OWL sublanguages have been proposed as OWL 2 profiles (Motik et al., 2009).

The OWL 2 language includes the three standard profiles: OWL 2 EL, OWL 2 QL and OWL 2 RL. They have been proposed to specify different application domains with restricted expressivity. Each OWL profile includes a subset of OWL 2 constructors and suitable for a particular type of applications and reasoning tasks.

The OWL 2 EL profile – has been designed for ontologies with a large number of classes and/or properties for which the basic reasoning tasks require polynomial time in terms of the size of the ontology (Motik et al., 2009).

The OWL 2 QL profile – has been recommended for specific applications that work on large volumes of data where the query-answering tasks require logarithmic space in terms of the size of the data (Motik et al., 2009).

The OWL 2 RL profile – has been designed for domains that require scalable reasoning but do not require too much expressive power compared to full OWL 2. OWL 2 RL implementations can use rule-based reasoning engines and query-answering over OWL 2 RL ontologies require polynomial time in terms of the size of the ontology (Motik et al., 2009).

1.4 E-Learning Systems and Databases

Databases work as a central and a shared repository of the data related to entities of a domain. In e-learning systems databases have been used to store data related to courses, learners, and learning resources. They are an indispensable component of information systems, including e-learning systems. The contemporary e-learning systems are based on relational databases that are created and managed using a Relational Database Management System (RDBMS). Databases have been used in systems including e-learning for nearly four decades. Here, we see some approaches for e-learning systems that are based on databases.

1.4.1 Databases for E-Learning Systems

Early e-learning systems have been proposed to change the class-room-based and instructor-based learning to learner-based and Internet-based e-learning systems. A variety of Internet-based technologies that enable the design and implementation of e-learning systems that aim at on-demand, low cost learning have been proposed in Zhang (2003). Among the many technologies, the database technologies are used for storing and manipulating e-learning

materials and to provide scalability that is required to support the learners (Zhang, 2003). Zhang (2003) has suggested using distributed computing technologies to achieve personalisation of a plethora of learning materials stored in databases.

Issues such as the optimum size in generating reusable e-learning objects (ELOs) are identified and an approach to create and reuse optimum sized ELOs that are stored in a database is proposed in (Muzio, Heins, & Mundell, 2003). These authors have suggested using text, images and audio templates to create, store and manage reusable ELOs. They suggest storing the ELOs in databases so that others can search for them. A developer is allowed to transfer them to a creator's library and transfer them back as a new ELOs after any updates or modifications. The copyrights of the ELOs are held by the original creator. However, these authors do not talk about any search criteria or use of queries to search the ELOs. Muzio et al. (2003) propose user interfaces to present the ELOs to the learners. However, personalisation or how to apply a learner-centred approach, are not discussed.

The above approaches have paid little attention to addressing the issues related to personalisation that demands conceptualising the learning contents. Some scholars have suggested storing user profiles in databases and some have suggested using software agents.

The architecture proposed for an e-learning system in (Chen, Lee, & Chen, 2005) is assisted by three main databases: a user account database, a courses database, and a user profile database, and several agents (Figure 1.1). In that work, an interface agent uses the user account database to identify learner status and then transfer learner's queries and returns the suggested course materials to learner. A course recommendation agent uses the courses' database and user profile database to select course materials that are appropriate for the learners from the course database.

In comparison to what is proposed in Zhang (2003), in addition to a course database, this approach has used a user account and a user profile database. Hence, the work in (Chen et al., 2005) could be considered as an extension to work in (Zhang, 2003). On the other hand, both of these works have included databases for managing data on learning materials and the assessment data.

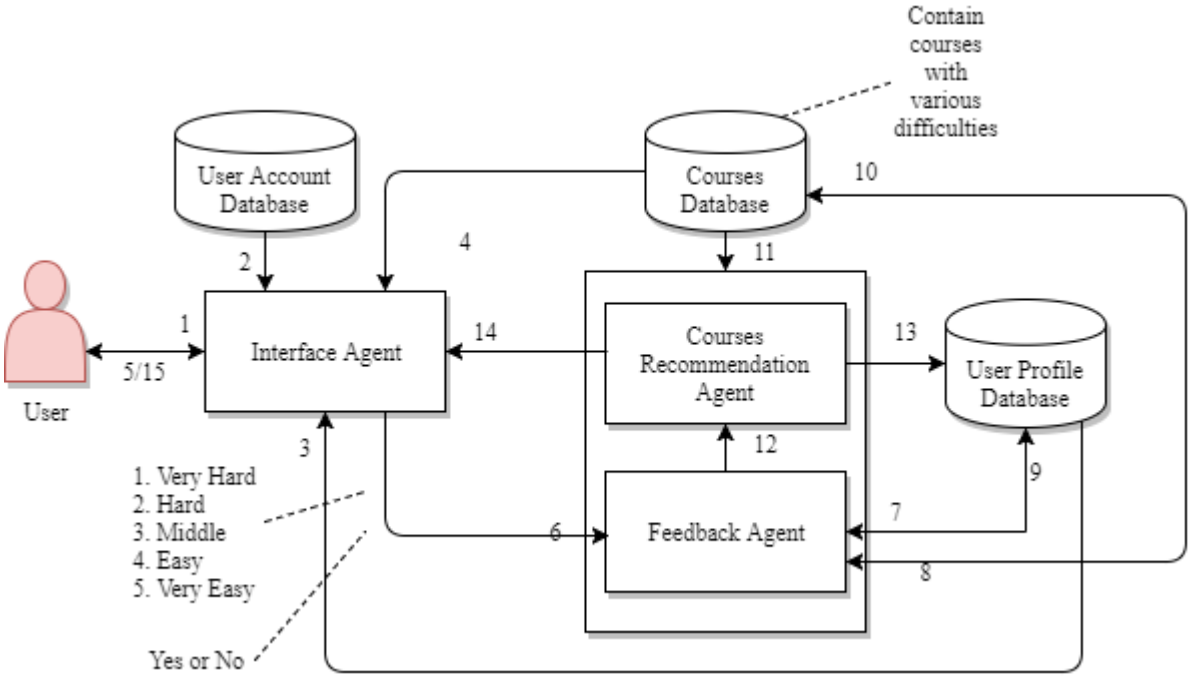


Figure 1.1: System Architecture – numbers 1 to 15 indicates the procedure of system operations (Chen et al., 2005)

A campus-wide Learning Content Management Systems (LCMS), adopted from the IBM Lotus Corporation's product *LearningSpace* has been proposed in (Cohen & Nycz, 2006), is a multi-database application that relies upon five databases. They have attempted to provide a broader context of using technology in e-learning to meet society's needs for learning especially for adults. Each database has been suggested to store various data. They include a schedule database, a media centre database, a virtual classroom, a profiles database and an assessment manager database. As their names imply they store different data related to different disciplines of a campus as required by that application. LCMS is not just a repository of learning objects, it includes systems and tools for authoring resources, and storing and retrieving the learning objects. This approach provides only an overview of the system and does not provide details of user profiles and criteria for retrieving the learning objects.

A Web-based diagnostic and dynamic assessment development and an assessment-centred e-learning system, named Graduated Prompting Assessment Module – Web-based Assessment and Test Analyses (GPAM-WATA), has been proposed in Wang (2014). This system provides personalised learning materials and assessments that is able to be adapted by the user. All learning contents of this system are stored in the e-learning material database. GPAM-WATA_EL e-learning system uses an instructional item and a prompt database for

dynamic assessment. The diagnostic item bank includes items developed by teachers based on their understanding of the misconceptions students have about learning concepts and are used to diagnose students' weaknesses in the learning process. For this diagnosis first, prior knowledge assessments and pre-tests have been given to the students. Then, remedial teaching and post-tests have been used.

The e-learning material database that is used in this system allows teachers to construct e-learning materials with multimedia content. In this approach, the instructional items in the personalised dynamic assessment are automatically retrieved from the instructional items database based on the learners' pre-test results of a two-tiered diagnostic assessment. This personalised dynamic assessment helps learners focus on interacting with the instructional items and instructional prompts that enhance their learning rather than on non-personalised dynamic assessments.

1.5 Enrichment of the E-Learning Systems with Semantic Web Technologies

The approaches introduced in the previous section show that systems for e-learning have traditionally relied on database technologies to manage learning objects. They have relied on agent-based technologies, and distributed computing and metadata to provide adaptive features to the learners. They are good at storing and managing learning objects and data, but not good at capturing domain knowledge. In database-driven information systems, domain knowledge is distributed in different parts of the system's application logic. It is only possible to store domain knowledge in 'stored procedures', 'triggers' or in an application program (Selfa, Carrillo, & Boone, 2006). In addition, database technologies have problems answering questions about intentional knowledge due to the fact that databases are intended for storing data rather than for representing the semantics of a domain. A repository of e-learning materials is considered as a knowledge base in (Zhang, 2003). Still, a repository of data or a repository of learning resources literally would not be considered to be a knowledge base. A knowledge base needs to store both terminological and assertional knowledge of a domain such as e-learning.

Even though conceptual schemas are used to design databases, operational databases do not provide a conceptual view to the users. Databases are more technical and based on Relational Algebra. Again, databases are based on the Closed World Assumption (CWA) and they work with complete data (Cortés-Calabuig, Denecker, Arieli, Van Nuffelen, & Bruynooghe, 2005). Query-answering is applied on databases to get results and happens by evaluating fixed

logical database structures. They do not provide direct reasoning capabilities based on a DL reasoning tool.

Four main differences between ontologies and database and other schema definitions have been identified in (Fensel, 2000). 1) The ontology languages currently used to define ontologies have richer syntax and semantics than the approach followed to define databases. 2) An ontology includes information which is described using a semistructured natural language, but a database includes information in a tabular format. 3) Terminology used in an ontology has to be commonly agreed upon and intuitive, as an ontology needs to be shared and/or exchanged. But, a database includes a terminology selected by the database designer that may not be intuitive to the others. 4) Also, an ontology represents domain knowledge, but not the structure of a data container (Fensel, 2000).

In contrast to databases, ontologies that are salient participants in the Semantic Web technologies have features which benefit and enrich the e-learning systems. Ontologies create an additional layer on top of the database layer in a learning system. Hence, a learning ontology is able to provide a simplified view of the learning contents by supressing the technical details and domain complexities.

Contemporary ontologies are designed using the ontology languages that are based on DL. Therefore, it is possible to do reasoning on e-learning ontologies using a DL-based reasoning tool. Reasoning helps to find the unknown facts based on the known facts rather than just retrieving the learning objects in comparison to the Structure Query Language (SQL) used in database systems. As the ontologies use the Open World Assumption (OWA) it is possible to work with complete or incomplete facts. If the facts are complete we are able to get complete results and if a learning ontology has incomplete facts we get incomplete facts or the system would say ‘not known’. When ontologies are used for reasoning they work as conceptual schema and the query-answering happens by evaluating an ontological structure that is navigated in multiple directions.

Many scholars have identified the Semantic Web technologies, primarily the ontology, to enrich the e-learning systems due to the benefits which ontologies offer. Detailed discussions on recent advances in e-learning and web-based technologies are provided in (Chrysafiadi & Virvou, 2015). In the next section we review the literature on the ontology-based e-learning systems in chronological order to identify the prominent features of the e-learning systems proposed in those approaches. We searched for what features have been thoroughly utilised

and discussed, and also what features have been given little attention. Finally, we attempt to identify areas of ontology-based e-learning systems that need further research.

1.6 Ontologies for the Learning Domain

The conceptualisation of the contents of e-learning system ontologies with their concepts and relationships makes it easy to understand them. There are several efforts in this line of research and we pay our attention to some of them in this section. They have attempted to conceptualise some aspects of the learning domain including the learning materials, users, instructors, and the devices used. Below we elaborate on the approaches that we found in the literature.

1.6.1 Modelling the Learning Domain Using Ontology

Different issues that contemporary higher educational institutes face due to a changed role have been highlighted in Dall'Alba and Barnacle (2007). They point out that universities focus on providing specialised knowledge rather than students' learning and several other problems. They suggest that conceptualising higher education programs as 'ontology' is a solution to the issues they have identified. Approaches to education are integral to learning or to taking actions. The authors believe that this dimension of learning needs to be revitalised through an ontological approach to higher education programs. They present a theoretical/conceptual account of the role of ontology in higher education that reasserts the ontological implications of learning. They stress that an ontological approach would help to provide specialist rather than general knowledge. However, no specific ontologies for institutions or use of ontologies in a system have been proposed in their work.

In line with the ideas in Dall'Alba and Barnacle (2007), a topic map ontology for e-learning has been proposed in (Kolås, 2006) that aims at actively involving students and teachers as producers of learning resources and sharing the student-made resources. In that work, topics in the learning domain and their associations to their occurrence have been identified from the learning resources. Those topics have been grouped under key areas: learning objectives, pedagogical methods and learning objects. Each key topic belongs to several topic types and has an association and occurrences (for example, Multiple Choice Questions (MCQs), short answers and blogs) (Kolås, 2006). Associations include, *is assessed through*, *is taught through* and *is produced through*. The topics are organised in a table that can be used as a source of candidates' terms for a learning domain ontology. The topic map proposed in this

approach shows the relationships between different learning resources. It is expected that the topic map will enhance teaching and learning.

This ontology in (Kolås, 2006) has been proposed for specifying a learning domain with learning objectives, learning methods and learning objects. It also identifies many resources related to each category, and associations of them, to provide a detailed specification of a learning domain. In Kolås (2006), a learning ontology has been presented in a tabular format to conceptualise the domain knowledge. However, the domain knowledge captured in this table could be depicted graphically using a diagram or in some other pictorial way that could enable visualisation of the domain knowledge.

A domain ontology to describe learning material that builds an adaptive course has been proposed in Gascueña et al. (2006). This domain ontology specifies a course that is capable of providing adaptive e-learning environments and reusable educational resources. This ontology consists of descriptions of resources, devices that are required to use the resources, and learning styles of the learners. Each resource in this ontology has been described using two characteristics, the most appropriate learning style and the most satisfactory hardware and software features of the used device. In Gascueña et al. (2006), learning styles are determined based on the Felder-Silverman Learning Style Model (FSLSM) (Felder & Silverman, 1998). Appropriate resources are related to the learning styles, and to the user's hardware and the software environment.

The approach proposed in (Kougias, Seremeti, & Kalogeras, 2011) works towards an ontology-based approach for enhancing educational activities to reduce the conceptual gap between the learners and instructors in Next Generation Education Environments (NGEEs). This aim is realised by semantically relating the cognitive profiles of learners and instructors by using ontologies. A set of ontologies is proposed to manage cognitive profiles of learners and instructors.

In the initial attempts to build cognitive profiles for learners and instructors, each individual described their profiles using different terms and in different ways (Kougias et al., 2011). Hence, a problem of lexical heterogeneity has appeared and occurs when the same term has different meanings, or the same meaning can be described using different terms (Kougias et al., 2011). Besides, structural heterogeneity occurs when the same concept is described using different relations and properties or when it has a different position in a hierarchy. The solution of those problems is to apply ontology merging and alignment approaches.

The users of the learning system are the learners and instructors. Hence, each learner's information such as age, gender, skills, learning preferences, special abilities, and cognitive background is considered to be the learner's profile. These profiles are represented by a Learner Profile Ontology (LPO) (Kougias et al., 2011). The instructor's information such as 'what to teach'—courses, content, learning goals, and 'how to teach'—lectures, teaching methods and conditions, depending on particularities, belong to instructor profiles. Instructor profiles are represented by Instructor Profile Ontology (IPO) (Kougias et al., 2011). When a class is considered the learners who attend a class have heterogeneous profile contents, as learners get different interpretations of their domain. Hence, class information is taken into class profile that is represented by a Class Profile Ontology (CPO) (Kougias et al., 2011).

The ontology-based cognitive profile management approach proposed in (Kougias et al., 2011) attempts to utilise a set of ontologies to alleviate the semantic gap between the cognitive profiles of learners and instructors. Further, this approach applies ontology alignment in creating semantic correspondence between the ontologies, and a set of ontology tools are manipulated to develop, merge, reason, and align ontologies. This approach has provided a broader view of learning with three main views: learner, instructor and a class.

In addition to focusing on modelling, the resources as in (Kolås, 2006) and the approach in (Gascueña et al., 2006) identifies the devices required to use learning resources and the learning styles to be included in the ontologies. The device descriptions include the subclasses; software and hardware. This approach has turned attention towards an ontology for learners and their profiles with their learning styles. The learning styles are based on the FSLSM (Felder & Silverman, 1998). They have provided a general layout of a domain ontology for learning and a hierarchy of learning resources with four layers of concepts. In this hierarchy, they have included assessment items as well. In comparison to the other approaches, this approach has considered several new areas for conceptualisation of a learning domain.

However, these approaches do not discuss the use of the ontologies in an e-learning system or within ontology-based systems architecture. We discuss such approaches separately in Section 2.4.

1.6.2 Description Logics (DLs) and Ontologies

The ontology languages OWL, OWL 2 and the OWL 2 profiles are based on the DLs (Baader et al., 2003). DLs is the name given to a set of knowledge representation (KR) formalisms

(Baader & Nutt, 2003). In DLs, first the terminology or the concepts in a domain are specified. Then roles and individuals of that domain are specified. They are the basic building blocks of a knowledge base (KB). The expressive power of a DL language depends on the constructors used in it. The basic DL language *ALC* includes the following DL constructors (Table 1.2). Other DL languages are derived by adding or removing the constructors from *ALC*.

Table 1.2: Constructors of ALC (Baader & Nutt, 2003)

#	DL Constructor	Name
1	C, D → A|	Atomic concepts
2	T|	Universal concept
3	⊥|	Bottom concept
4	¬A|	Atomic negation
5	C⊓D|	Intersection
6	∀R.C|	Value restriction
7	∃R.T	Limited existential quantification

A KR system consists of a terminology box (TBox), an assertions box (ABox), a DL language and a DL reasoner (Baader & Nutt, 2003). A KB consists of a TBox and an ABox: a TBox holds the vocabulary of an application and an ABox holds the assertions about the named individuals. Domain rules are applied to them and the knowledge can be retrieved using application programs. The knowledge that is not explicitly expressed in a KB is inferred automatically by using an inference procedure (Baader & Nutt, 2003). DLs and OWL use different terms in their language and a comparison is given below (Table 1.3). Axioms are statements that are true about a domain. A terminological axiom shows how the concepts and roles are related to each other.

Table 1.3: Terms used in OWL and DL (Baader & Nutt, 2003)

OWL	DL
Class	Concept
Object Property	Role
Ontology	Knowledge Base
Axiom	Axiom
Vocabulary	Vocabulary

1.7 Thesis Focus and Key Contributions

The main focus of this thesis is to propose an architecture for an e-learning system, instances of which could be deployed at different institutions. We provide a description of different SW

components of this architecture and technologies that could be used in the proposed architecture. One main focus of the proposed architecture is to represent the domain knowledge as an ontology. This thesis also identifies the common language constructors of learning ontologies that makes a sublanguage for developing learning ontologies.

This thesis provides the answers to the research questions and proves or disproves whether the following experiments provide feasible solutions:

1. Use of instances of a common e-learning systems architecture in multiple institutions.
2. Identify a sublanguage to represent domain knowledge of the learning domain based on an analysis of a corpus of learning ontologies developed for different institutions.
3. Use of the proposed sublanguage to represent the domain knowledge of different institutions.
4. Use of existing methods to map the domain knowledge in databases into a knowledge representation format so that they could be used in a knowledge base system.
5. Use of proposed architecture in at least two institutions using two instances of the proposed system, and,
6. Use of the domain knowledge in at least two institutions using a benchmark query suite.

The major contributions of this research are:

1. Proposing the design of an *ontology-based plug and play architecture* for an adaptive e-learning system's framework, instances of which could be deployed at different institutions.
2. Designing an OWL 2 sublanguage for the learning domain based on an analysis of a corpus of learning ontologies.
3. Proposing an ontology benchmark query suite to evaluate the query-answering and inference capabilities of learning ontologies.

The auxiliary contributions include:

1. Identification of the terminological differences between the institutions based on an analysis of unit guides and course handbooks.
2. Identification of different types of architectures proposed for e-learning systems, learning ontologies and their features based on a literature survey.
3. Writing mapping rules to map a relational database of an e-learning system to a learning ontology.

4. Identifying relevant metadata and user profile attributes to retrieve user specific learning contents.
5. Developing a proof of concept e-learning system with a query interface that hides the technical details from the user for querying a learning ontology.

1.8 Thesis Outline

The rest of this thesis is organised as follows:

Chapter 2 provides a review of literature related to e-learning and learning ontologies. We provide a critique on related literature in the last two decades and a summarised comparison. The purpose of this exercise is to gain an overview of the current work on learning ontologies and ontology-based e-learning systems architectures to determine directions for our research.

Chapter 3 explains the *plug and play architecture* for ontology-based e-learning. We propose this as a part of the solution to the research problem in line with the motivation explained in Chapter 1 and as an outcome of the literature survey.

Chapter 4 provides an analysis of the OWL 2 constructors used in a corpus of the learning ontologies and OWL 2 RL profile. Based on this analysis, a sublanguage of OWL 2 is proposed for the learning domain ontologies. This is the second part of the solution that restricts the use of OWL 2 constructors in developing a learning ontology that is pluggable to an instance of the proposed architecture.

Chapter 5 describes developing a learning ontology that uses the OWL 2 Learn constructors and an ontology editing tool. We demonstrate the creation of different elements of a learning ontology that will eventually be plugged to an instance of the plug and play architecture.

Chapter 6 explains mapping the schema of a learning database to the TBox of a learning ontology and populating a learning ontology with the instances based on the data in a relational database. This database to ontology mapping is done using a current mapping tool which is available as an open source.

Chapter 7 introduces ontology evaluation techniques and explores an approach to evaluate OWL 2 Learn ontologies. Based on an analysis of the current benchmarks for evaluating Knowledge Bases Systems (KBS) a benchmark query suite is proposed to evaluate the OWL 2 Learn ontologies. We show the use of the query suite to evaluate the query-answering and inference capabilities of two sample ontologies which we developed for two institutions. This

chapter also discusses the evaluation of the adaptability of the e-learning system that is based on *plug and play architecture* for two institutions.

Chapter 8 provides the conclusion of this thesis. This chapter summarises our findings and reviews how far we have proved or disproved the research questions. It also discusses limitations of this research and suggests further work that could complement or extend this research.

Chapter 2: Related Work on Ontology-based E-Learning Systems

In our research, we have reviewed a collection of contributions on Ontology-based E-Learning Systems that were returned as search results from the contemporary library databases such as Google Scholar, IEEE Xplore, ScienceDirect and SpringerLink. In our study, we intended to use our search results to identify and address the problems, study the approaches, and learn from the distinct characteristics of those approaches as related to ontology-based e-learning.

E-Learning has been a popular research area over the last two decades. E-learning became a way of facilitating distance learning or to complement face-to-face learning and blended learning. Most of the early e-learning systems used databases as repositories of the educational programs, their units of study, the learning materials and the details of the learners. However, the databases are at the backend of a system and use technical language which makes it difficult to understand, especially when the learning domain becomes complex. Hence, it requires the contents of a database to be conceptualised for a learner to get a clear picture of a course to gain knowledge. Some attempts have been made to simplify and get a better understanding of a learning domain in general, by specifying it as an ontology. Later e-learning systems have been introduced with ontology as an element of many e-learning systems that led to development of ontology-based e-learning systems.

Due to the benefits of ontologies in providing a conceptual view of the contents, ontology-based e-learning systems has also been a popular research area in the last two decades. The literature survey conducted in (Al-Yahya, George, & Alfaries, 2015) shows a substantial growth in research on approaches to ontology-based e-learning. Recently published papers report on different scholarly work on ontology-based e-learning systems and other Semantic Web technologies. The incorporation of e-learning systems in the changing Semantic Web environment, and achieving adaptive personalisation based on the learner's progressive behaviour, have been observed as two challenges in e-learning (Rani et al., 2015).

In this chapter, we review the literature on e-learning systems that are based on ontologies and the Semantic Web technologies. We specifically review the literature, the scholarly works, from the year 2000 to the year 2017 to gain insight into how the research on this area

has changed and what significant contributions have been made to this research area. We also see how research has led to evolution of the ontology-based e-learning systems that are backed up by the Semantic Web technologies.

Here we critically analyse literature which has been published over the last 17 years to identify the prominent perspectives of ontology-based e-learning systems. The intention of our literature survey is to study the literature on ontology-based e-learning systems; in particular, to identify their specific features, what work has been done and what further research needs to be done. Based on our study and analysis we identify a research area for us to make a substantial contribution to the research on the ontology-based e-learning systems.

The rest of this chapter is organised as follows. In Section 2.2, we provide an introduction to the ontologies proposed for the learning domain. Section 2.3 introduces e-learning systems that are based on databases. Then, in Section 2.4 we present, in chronological order, ontology-based e-learning systems' approaches and their significant features that are found in the literature. Section 2.5 provides a summary of the ontology-based e-learning systems that lead to identify some research problems for further research.

2.1 Ontology-based E-Learning Systems

In Section 2.2, we examined the ontologies proposed for the learning domain in several scholarly works. Even though those proposals conceptualise the learning domain, they do not discuss the application of ontologies in e-learning systems. Ontologies have been used as a critical component of many of the recent e-learning systems. In the rest of this section, we review the works and the approaches reported in the literature for the design, development and the use of ontologies in e-learning systems. In particular, we attempt to analyse the efforts in this area in order to gain insights into the critical components and features of ontology-based e-learning systems. We determine the relevance of the literature to ontology-based e-learning and evaluate it based on these criteria:

- Use of the ontology and types of ontology used
- What ontology language(s) are used
- Ontology design and population
- Use of ontology in systems architecture
- Personalisation methods and use of ontology

- Reasoning and querying technique(s)
- Technologies and tools used for ontology design and reasoning.

At the end of this section we summarise our findings and identify the knowledge gaps in this research domain. Then we discuss potential for further research which may help shape and enhance ontology-based e-learning systems.

2.1.1 E-Learning Based on the Semantic Web

The Semantic Web-based e-learning system proposed in (Stojanovic, Staab, & Studer, 2001) includes a course ontology as the backbone for specifying and accessing the learning materials. The course ontology consists of content, context and structure ontology to describe the content, context and structure of the learning materials. The authors have seen that these three ontologies have the benefit of providing greater access to the learning materials. A context ontology is used for providing a shared understanding about meaning of the context vocabulary (e.g. introduction). The context ontology is based on the pedagogical model and it includes concepts such as introduction, explanation, and examples that are used to describe the context of the learning materials. The purpose of the content ontology is the description of domain terms like *protocol*, *service*, *topology*, etc. The content ontology also includes the relations like *hasTopic*. The structure ontology given in the work of Stojanovic et al. (2001) describes the structure of learning materials. The critical part of the structure ontology is the relationship between learning materials and corresponding rules. The learning materials in the structure ontology are organised in a tree structure and include relationships to describe a sequence of the documents (Stojanovic et al., 2001).

The educators provide annotations on learning materials based on context, contents and structure and store those annotations in a document storage. They could use different terms in annotations which makes it difficult to provide a shared understanding. The facts about the context, contents and the structure have been identified as the main criteria that a student may use for searching learning materials.

All the ontologies and metadata have been identified as main components of a system's architecture (Figure 2.1) in (Stojanovic et al., 2001). Metadata on learning documents work as a bridge between the ontology and the learning documents. The system provides user access to the documents via a personalised interface. The user requests that include context, content and structure information are passed into queries. Then, the queries pass that

information to the middleware (an inference server) that utilises the ontology to identify the relevant learning documents and to retrieve them.

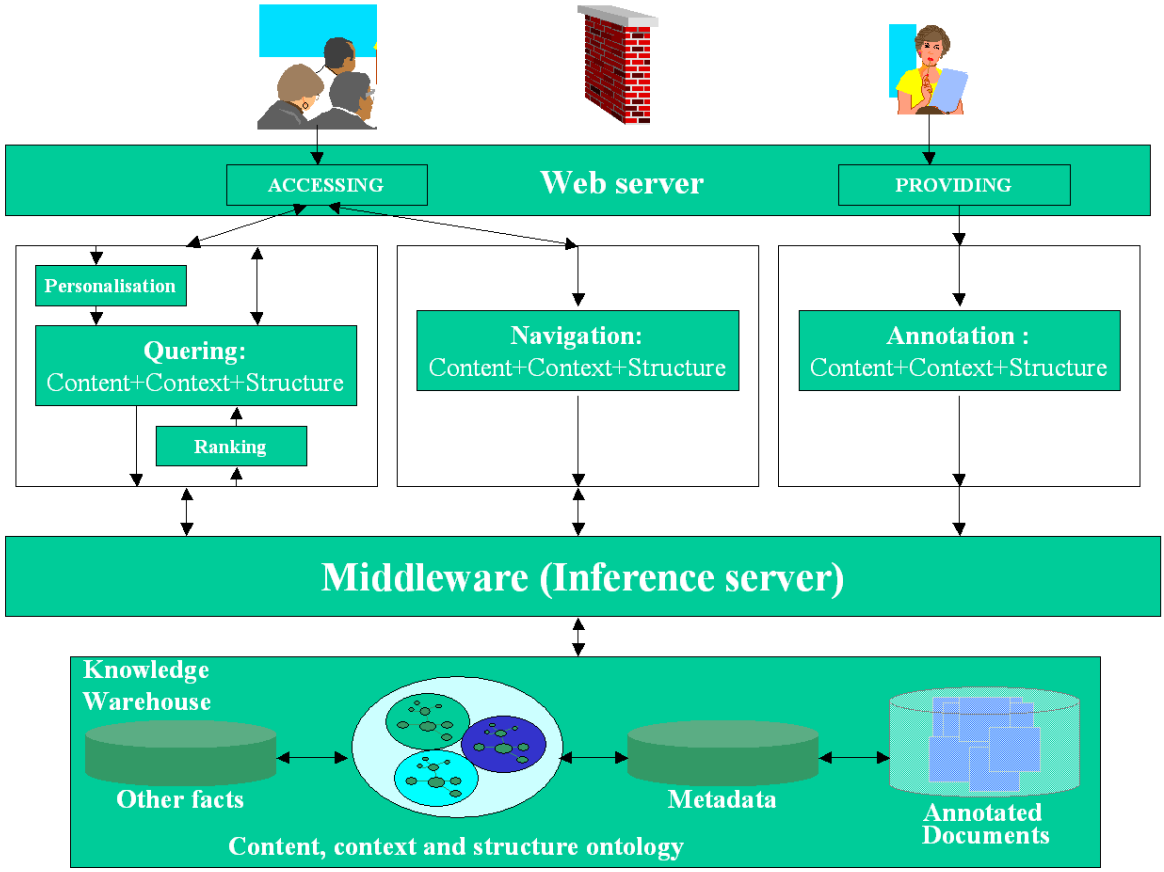


Figure 2.1: Architecture of an e-learning portal (Stojanovic et al., 2001)

This approach includes features which draw our attention and are worth highlighting. It elaborates on three aspects of learning materials (context, contents and structure) that are specified as ontologies. The authors also define some derivation rules as well in the ontology to infer new knowledge. The proposed ontologies are used in a system with associated IEEE LOM to retrieve the learning materials. Users use the context, contents and structure details in a query interface and the system maps them with the associated learning materials through the ontologies in query-answering. It talks about using rules on the ontology. However, this approach does not provide details of query language or the use of any reasoners or the expressivity of their ontologies that are in RDF.

2.1.2 Reasoning and Ontologies for Personalised E-Learning in the Semantic Web

Providing distributed information with a clear and well-defined meaning that is understandable for different parties has been identified as challenging (Henze, Dolog, & Nejdl, 2004). So it is important to provide optimised access to the information. As a solution

to that, a framework for personalised e-learning in the Semantic Web has been proposed. The Semantic Web resource description formats are utilised to generate hypertext structures from distributed metadata by reasoning mechanisms. A demonstrator is used to generate a conceptual context of learning resources for each learner by using adaptation rules.

The work presented in (Henze et al., 2004) uses three main ontologies to support the above process; a domain ontology, a user ontology and an observation ontology. A learning documents ontology model, that is a logic-based approach applied to investigate educational hypermedia, and proposes ontologies and metadata for three types of resources (domain, user, observation). These ontologies are aimed at achieving adaptive functionality of educational hypermedia systems. The domain ontologies model the documents and the relationships between them within the domain, user performance and observations about learner interactions (Henze et al., 2004). This approach proposes three ontology models on documents: an *ontology of documents*, an *ontology of document types*, and a *concept ontology*. The *user ontology* describes the learner characteristics and the *observation ontology* specifies the different possible interactions of the learner with the hypertext resources. Finally, the generated hypermedia structures are represented in a presentation ontology. Ontologies are given in RDF files.

In comparison to the other works, the work presented in (Henze et al., 2004) provides several additional ontologies to elaborate on the domain. It also utilises a user ontology, a user observation ontology and a presentation ontology to provide personalised learning contents. Another important aspect of their work is the use of reasoning rules that are encoded in the TRIPLE rule language and the same language is used for querying RDF. Reasoning in their work is based on First Order Logic (FOL).

2.1.3 Sharing Learner Profile through an Ontology and Web Services

A web services-based solution has been proposed for easy exchange of learners' information among different e-learning systems (Musa, Muñoz, & de Oliveira, 2004). The term *Web Services* is described as "a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone". The goal of this is to let different e-learning systems work harmoniously with each other to provide richer learner information than information found in the standard e-learning systems. The architecture shows how the collaboration between e-learning systems enriches the learner model. The learner model repository is the central element of this architecture. This

stores data gathered from different e-learning systems and could have a complex distributed structure. The access to the central learner model repository is through the Web Service. The communication and exchange of data between the learner model repository and the systems is done through the Web Service. The data is exchanged between applications using a learner model that has been defined as an ontology. Once access is granted details of operations available in the Web Service are provided in Web Services Description Language (WSDL) format. Learner information available in the learner model repository is exchanged in XML format and through SOAP protocol (Musa et al., 2004). It represents the learner model ontology in OilEd and it is implemented in DAML+OIL. Validation of the ontology is done using the reasoner Racer.

The notable features of Musa and colleagues' (2004) work are the use of different web technologies to share learner ontologies in different e-learning systems to build a collaboration among them. IEEE LOM are used to describe the learning objects and to achieve personalisation. This approach also has used many Semantic Web technologies including the Racer DL reasoner.

2.1.4 Communication Ontology and Integration of Learning Ontologies

The authors (Aroyo & Dicheva, 2004) attempt to give a solution for the next step in the evolution of e-learning. They suggest moving from a scattered intelligence to a coherent and collaborative intelligence. An approach towards a common reference architecture for adaptive concept-based, web-based educational systems (WBES) has been proposed in (Aroyo & Dicheva, 2004). This architecture is component-based and it allows sharing knowledge and components of the system. In comparison to the architecture proposed in (Navigli & Paula, 2004), this architecture (Aroyo & Dicheva, 2004) concentrates more on the communication between learning systems using an ontology to specify the terms used by those learning systems.

The authors consider user modelling, course sequencing, ontology visualisation and keyword search as the main components of the architecture. They also consider the domain ontologies, learning resources, course models, and user models as shared knowledge of the system. The authors also suggest that this architecture addresses four main research questions:

For the level of granularity of information exchange—what information should be included in a one communication transaction; for request semantics—what type of questions the requesting system can ask; for request syntax—what the format of questions is; and, for

domain or user model awareness—whether the requesting system should pass on any information it already knows. According to this architecture each educational information system (EIS) utilises a domain ontology and for the understanding between EISs, each EIS is expected to know the terms (concepts, relationships and roles) in the learning repositories. That is, each EIS must know how to map its domain knowledge into common learning concepts. This mapping process is a key feature of this approach. The author also proposes to use a common user model that is possible through a concept-based representation of the subject domain.

The architecture proposed in this paper includes four main components: WBES that use their private subject domain ontologies; information brokerage bureau to register applications; services that support communication between WBES (e.g. for ontology mapping); and, communication bridges between WBES that include transport mechanisms and interaction protocols. Besides those components, a communication ontology is proposed to define the semantics of the messages, message contents and content layers. To interpret the requests and responses standard domain ontologies, user modelling ontologies and upper ontologies are proposed.

Even though (Aroyo & Dicheva, 2004) talk about a domain ontology, they don't elaborate on how it is constructed. However, their ontology-based architecture for WBES provides directions to build interoperable learning systems. They aim at sharing a communication ontology which defines the terms used by individual learning repositories and the relationships between them. Use of multiple ontologies in e-learning systems' architecture and architectural features and functionalities of the system components are discussed.

2.1.5 OntoEdu: A Case Study of Ontology-based Education Grid System for E-Learning

It is questioned whether we are moving from a scattered intelligence to a collaborative intelligence in the WBESs. A flexible educational platform that is based on several new technologies including ubiquitous computing, grid computing, ontology engineering and Semantic Web has been proposed by (Guangzuo, Fei, Hu, & Shufang, 2004) for e-learning. It aims at overcoming the two challenges: interoperability among numerous educational systems and providing structured and unified authoring support. The architecture is based on the grid-based system design and ontology. It consists of five main components: *user adaptation*, *automatic composition*, *education ontology*, *service module* and *content module*.

A grid computing system is a geographically distributed environment with autonomous domains that share resources amongst themselves (Azzedin & Maheswaran, 2002). Grid computing is a service-oriented approach (Guangzuo et al., 2004) that can help educational institutions to aggregate distributed educational resources and build a unified system to address workflow requirements.

Each component of the ontology-based grid computing architecture proposed, OntoEdu (Guangzuo et al., 2004) has specific objectives.

1. *User adaptation* module accepts user (context/preference) requests, translates physical inputs from input devices into logical inputs and translates logical outputs into physical outputs.
2. *Automatic composition* that consists of system description and system composer creates tasks for each user request. System description creates function descriptions in OWL-S after reasoning on the education ontology. System composer translates the function descriptions into an implementation format.
3. *Education ontology* describes the educational content, educational activities, operations and relationships among them. When education ontology is considered separately, it consists of two main parts; A. *Activity ontology* (AO) and B. *Material ontology* (MO).
4. *Service model* tries to achieve a higher level dynamic model satisfying the requirements: open architecture and interface, high interoperability for information exchange, flexibility, accessibility, durability and reusability, and compatibility with other systems.
5. *Content model* is same as the service model.

The main features of this system are: 1. the integration of multiple technologies; 2. Aggregation of the distributed educational resources using an educational ontology; and, 3. User adaptation with ontology support. A lightweight mapping process is used to map the concepts in different systems instead of using an expensive reasoning process. However, it gets the reasoning support of the authoring tools.

2.1.6 Ontology Alignment for IT Lesson Planning

An ontology-based system has been introduced in (Kasai, Yamagushi, Nagano, & Mizoguchi, 2005) for teachers of IT education, with useful web resources. This system attempts to align multiple ontologies with the purpose of reusing the results of other research. This aims at

teacher education as a solution to the lack of IT teachers as Kasai et al. (2005) have mentioned. The proposed architecture includes two main groups of metadata. One group of metadata is based on the ontologies (the goal of IT education and the ontology of the fundamental academic ability) proposed by the aforementioned authors and the other group of metadata is based on the Goal List of IT Education (kayoo.org, 2001). The goal list is not high quality as it is not based on ontology theory. However, it has been used by Japanese IT teachers (Kasai et al., 2005). The Goal List has already been used by teachers, like an ontology, to annotate IT educational resources. Hence the Goal List is considered as an ontology (Kasai et al., 2005). The system is capable of doing semantic integration between the ontologies and the Goal List to reconstruct tagged lesson plans, integrate lesson plans, and effectively retrieve IT educational resources which are useful for teachers.

The Semantic Web application was built, based on architecture to align ontologies with the Goal List, and is structured into four layers: 1. web layer; 2. RDF model layer; 3. RDF-Schema layer; and, 4. ontology layer. The ontology layer, at the bottom defines all the concepts related to the two ontologies and the Goal List. The concepts related to IT education have been organised into an ‘is-a’ hierarchy (see Figure 3 p. 13 in Kasai et al., [2005]). The RDF-Schema layer includes the vocabularies of the classes and properties that are used in the RDF model layer. The RDF model layer defines the metadata (to align the ontologies and the Goal List) to create relationships between the two ontologies and the Goal List. These metadata items are RDF vocabularies. These relationships are defined using the vocabularies included in the RDF-Schema layer. The web layer retrieves the digital IT educational resources (digital recipes) from the Web (provided by Okayama Prefecture Information Education Center [OPIEC]) that are based on the Goal List. The web layer then annotates the digital recipes based on the two ontologies and the Goal List (ontology alignment).

The approach proposed in Kasai et al. (2005) attempts to align two ontologies with IT education resources and the Goal List of OPIEC to annotate IT education resources using RDF metadata to increase the reusability of IT education resources (digital recipes) that are used by the Japanese IT teachers. This approach proposes a layered architecture for ontology alignment application and defines the IT education concepts (IT education goals) in an ‘is-a’ hierarchy.

2.1.7 Learning Ontology from Relational Databases

The approach presented in Li, DU, and Wang (2005) automatically builds a learning ontology from a relational database. It exploits a group of learning rules in the ontology acquisition process (Li et al., 2005). Based on these rules the learning domain ontology is built in Web Ontology Language (OWL). This approach also proposes an ontology learning framework, where the ontology is a part of the system's architecture.

In acquiring the learning ontology from the relational database, the database is expected to be in third normal form (3NF). This approach proposes five groups of rules to be applied to the database to identify the concepts and relations that will become parts of the ontology. They are: rules for learning classes; rules for learning properties; rules for learning hierarchies; rules for learning cardinality; and, rules for learning instances.

According to the rules for learning classes, potential concepts are identified. Based on the rules, related data available in multiple relational tables may be used to make one concept. The rules for learning properties are used to identify the roles (relations) between ontological concepts such as *is-part-of* or *has-part*. According to these rules, when two relational tables are related by a property/role (e.g. 'part-of'), those two relational tables become concepts in the ontology and they become the range and domain of the role as well. The rules for learning hierarchies are used to create concept hierarchies and role hierarchies, based on inheritance relations (supertype to subtype relations) between the relational tables. The rules for learning cardinality lead to identifying *minCardinality* and *maxCardinality* in the OWL ontology. For example, if an attribute in a relational table is defined as *NOT NULL*, then the OWL *minCardinality* is defined as 1 and if an attribute is defined as *UNIQUE*, then the OWL *maxCardinality* is defined as 1. According to the rules for learning instances, tuples or rows in a relational table or in relational tables related by 'foreign keys' become instances of a concept in the ontology.

These rules are stored in the rules library within the proposed ontology learning framework. The rules library is used by the ontology generator to build the learning ontology. The ontology generator gets the source information (information about relational database tables, the relations between them, their attributes, etc.) through the database analyser. The ontology is then fed into the parser application. This framework also includes an ontology reasoner and an ontology editor that have access to the learning ontology.

This approach is a step forward in generating ontologies compared to the approach proposed in (Navigli & Paula, 2004) that builds ontologies from the details available in document warehouses and web sites.

2.1.8 A Learning Design Ontology based on the IMS Specification

Metadata is easy for humans to understand, but it would be difficult for a machine (a computer) to automatically process it. It is claimed that the XML-Schema language has limitations in representing the IMS Learning Design (IMS LD) metadata specification in (Amorim, Lama, Sánchez, Riera, & Vila, 2006). They have suggested ontology as a way of formally and explicitly structuring and giving meaning to the metadata. A learning design ontology that has been designed in OWL is proposed to solve those limitations. The concept taxonomy and the ontology axioms are described and an example is used to illustrate how the ontology could be used to overcome these limitations. The solution, ontology is also presented in an educational environment. The IMS LD ontology for the upper concepts has *Unit of Learning* as the main concept and the *Organisation* of the *Unit of Learning* and the *Resources* used in the *Unit of Learning* are shown with the relations between them.

The proposed ontology aims to solve two issues. Firstly, the semantics of the concepts are precisely defined. Therefore, the instances of the concepts are created and managed in runtime with no misinterpretations or errors. The new concepts, attributes/relations, and formal axioms of the domain have been identified and formally represented in the ontology (Amorim et al., 2006). Secondly, the semantics of the IMS LD specification are explicitly described. Therefore, it avoids the need of codifying the semantics in the process of developing the software program for the users to design and execute specific learning units. Hence, a general reasoner that follows the same ontology language used to write the ontology can be used to check the consistency of the ontology.

This architecture of the system is based on an intelligent agent technology which utilise the LD ontology to manage the information about the learning resources. The architecture of the system includes four different tiers: the resource tier—for control of hardware/software; the services tier—to allow educational activities; the mediator tier—a common channel to route the messages between services and clients; and the client tier—graphic interfaces for the adaptation/personalisation of the contents and services to the user's preferences.

This approach proposes to use a learning design ontology to overcome the weaknesses in metadata and XML in specifying a learning domain. In comparison to the other approaches it

uses agent technologies to process the ontology and to support learning. The architecture proposed in this approach includes four layers. As in other approaches, this one also uses OWL as the ontology and Protégé as the ontology editor. One drawback of the LD ontology is that it could be affected by the limitations of the expressiveness and reasoning capabilities of the ontology languages (Amorim et al., 2006).

2.1.9 An Ontology and a Software Framework for Competency Modeling and Management

The management of competencies that is the central goal of any education or knowledge management processes. A competency ontology for learning and knowledge management has been proposed in (Paquette, 2007). It is required to embed acquiring new competencies, in any software framework, as an instructional engineering tool (Paquette, 2007). In developing the top-level competency ontology, the author has referred to different fields: mathematical logic, science methodology, educational technology, and artificial engineering (Paquette, 2007). Paquette has analysed different definitions of competency in developing the competency ontology. "Competencies are statements that link together skills and attitudes to knowledge required from a group of persons and, more generally, from resources" (Paquette, 2007). An example competency is given below (Table 2.1).

Table 2.1: An example competency statement (Paquette, 2007)

Source	**Competency Statement**	**General Skill**	**Knowledge Entity**
Ministry of Education Quebec (MEQ) – Student Competencies	Analysis and synthesis capability	Analyse Synthesise	All subject matter in the curricula

The top-level competency ontology includes the high-level concepts associated with competency and specifies those associations. Competencies annotate resources (human or media) (Paquette, 2007) and competency can be a prerequisite competency (prerequisite) or a target competency (objective). Each competency is made from a competency statement. Competency is based on generic skills that are measured by performance indicators. A generic skill is applied to a knowledge entity. A knowledge entity is also a part of competency and belongs to a domain ontology. A generic skill also belongs to a generic skills ontology.

The generic skills ontology is a subontology that extends the competency ontology. It is based on the generic skills taxonomy that is applicable to different domains and includes knowledge processing activities. In addition to the generic skills ontology, a performance indicators ontology has been developed and is an extension, or a subontology, to the competency ontology. Another notable feature of this approach is that the figures have been drawn using MOT[1] editor according to the MOT + OWL graphic syntax that covers OWL-Description Logic (OWL-DL). According to that syntax, rectangles show classes, hexagons represent properties and 'S' refers to subclass.

This approach has provided a comprehensive discussion on the competency and skills of the learners that are specified in the ontologies. Even though ontologies have been proposed for the learning domain to give a refined direction for higher education and learning, many systems are based on database technologies. Here in the next section we give an overview of some e-learning systems which are based on database technologies.

2.1.10 Metadata Application in Development, Exchange and Delivery of Digital Reusable Learning Content

A need for learning contents with higher adaptivity and flexibility has been identified based on a survey for a Bulgarian University in Yordanova (2007). A metadata and ontology-based approach to satisfy the needs of adaptive and flexible learning contents has been proposed in Yordanova (2007). The proposed model is applicable on existing Learning Content Management Systems (LCMS). Metadata are used in learning systems to describe Reusable Learning Objects (RLOs) and to store in digital repositories for sharing and reusing. Metadata defines the title, language, description, keyword, format, learning resource type, interactivity level, and difficulty of RLOs. These metadata are searched and compared with learner preferences to deliver the RLOs in an adaptive and flexible way.

A model for integrating metadata with RLOs in the delivery of adaptive and flexible learning contents is also presented (Yordanova, 2007). These metadata items defined on RLOs facilitate search of RLOs by the learners. RLOs are stored in a database (RLO DB) and organised in a hierarchy with the relations that is specified in an ontology. The RLOs are accessed through the ontology and reused with different learning contents. According to this model learners/students and teachers/instructors can search for RLOs using different search

[1] http://www.cogigraph.com/Produits/OWLDLOntologyEditors/tabid/1100/language/en-US/Default.aspx

criteria (e.g. language, keyword, title and format). The metadata manipulated in this approach are IEEE LOM (IEEE 2002). LOM are organised into eight main groups (general, life cycle, technical, educational, rights, relation, annotation and classification). The implementation of metadata has been done using a Protégé tool that allows different metadata formats such as RDF(S), OWL and XML Schema.

This approach depends more on metadata than on ontology. It uses different types of metadata to allow users to search for RLOs that adapts to learning content based on an ontology to support flexible learning.

2.1.11 An Ontology-based Planning System for E-Course Generation

Instead of heavily depending on metadata, a learning system named PASER that is based on Artificial Intelligence (AI), Planning and Semantic Web technologies for automatically synthesising curricula has been presented in (Kontopoulos, Vrakas, Kokkoras, Bassiliades, & Vlahavas, 2008). Classical planning techniques are used to dynamically construct learning paths even from disjointed learning objects. That is done by being based on the learner's profile, preferences, needs and abilities (Kontopoulos et al., 2008). Three metadata repositories to feed its modules with certain educational metadata are deployed in this approach: 1. the LOM repository stores metadata about the available learning objects; 2. a repository of LIP compliant metadata describes the learners who have access to the system; and, 3. the Reusable Definition of Competency or Educational Objective (RDCEO) metadata repository provides competency definitions which are referenced by the other two metadata repositories (Kontopoulos et al., 2008). The systems architecture presented in their work mainly focus on the core modules: the planning subsystem responsible for synthesizing the curricula, the repositories for ontology and metadata; the knowledge base module that is responsible for queries and reasons on learning metadata.

The system's (PASER) architecture includes five processing modules: a planner, an ontology and metadata server, the RDEVICE module and two data converters.

This approach uses the IEEE LOM standard and the ontology has been developed using RDF schema. The ontology consists of 310 AI-related competencies that are organised using *isPartOf* relation. The ontology includes the subcompetencies such as *Machine Learning, Planning,* and *Knowledge Representation* of the concept *Competency* and goes up to five levels of the hierarchy. Each competency is given a *preferred* and *alternative* label.

2.1.12 An Ontology-based Semantic Learning Layer Cake

This approach by Dutta, Maddali, and Prasad (2009) proposes a conceptual framework of semantic e-learning to overcome the problem of lack of shared understanding between terms of various metadata vocabularies (Stojanovic et al., 2001). These vocabularies? use ontologies as a conceptual backbone in an e-learning scenario. This architecture involves three main aspects of an education information system: content, context and structure introduced in (Stojanovic et al., 2001). However, (Dutta et al., 2009) are innovative in their approach to propose an architecture which logically organise its components into layers: learning objects layer, metadata + ontology (context and contents) layer, ontology (structure) layer, rules (learning design) layer.

The conceptual learning space Dutta and colleagues (2009) have proposed is called a *semantic learning layer cake.* While this framework discusses an architecture for a learning system, it covers the personalised aspects of learning systems by increasing their flexibility with the use of metadata. Again, they provide an extensive discussion of the learning ontology, parts of which belong to different layers of the framework specified in the Semantic OWL. The bottom layer of this framework includes the content objects that are made up of the content fragments (e.g. texts, images, sounds, and data sets). The related content objects are grouped to make learning objects. The semantic learning layer lies on top of the content objects. The semantic layer includes the *semantic content* of the learning objects formally represented as the *content ontology* or the *domain ontology*. The *context* is used to provide facts or circumstances of the learning objects. Context is represented in the following aspects: matching the educational level (of both the document and the learner); intended use of the learning object; and, the learning objectives. The structure which lies on top of the content and context ontologies defines the relations between the learning materials such as, *hasPart*, *isPartOf*, *hasPrerequisite* and *isPrerequisiteOf*. In addition to the content ontology, this framework includes a document ontology and a student ontology. The document ontology specifies the significant concepts; *context*, *learning objectives*, and *learning resource type*. Different *learning objects* are organised under the concept *entity*. The student ontology captures all student related concepts and their relations to the concept *student* (Dutta et al., 2009).

While the semantic learning layer cake is the main conspicuous feature of this approach, other features such as attempting to personalise contents using an ontology, and building the learning ontology to address *context*, *contents* and *structure* are also notable features of this

approach. Besides, this approach organises the three main aspects of context, content and structure of learning systems, while their approach (Stojanovic et al., 2001) stresses the importance of those aspects. Stojanovic and colleagues have tried to identify some issues of e-learning systems and their causes. Accordingly, they have proposed a personalised learning environment. Metadata standards IEEE LOM and Dublin Core have been used to describe the learning materials. The ontologies that provide a semantic backbone to metadata are in OWL-DL and have been developed using Protégé, and the logic rules have been developed using N3Logic, subset of the FOL. Rules have been written for two types of learner characteristics, *learning style* and *cognitive learning style of learning*.

2.1.13 Advanced Ontology Management System for Personalised E-Learning

Insufficient expertise in the area of knowledge engineering in e-learning has been stressed as a main problem in modelling educational domains in (Gaeta, Orciuoli, & Ritrovato, 2009). These authors highlight the lack of methodologies and techniques for effective management of ontologies, especially in the areas of ontology research: ontology versioning; ontology harmonization; and, collaborative ontology construction. As a solution to this problem, an integrated approach for managing the life-cycle of ontologies has been suggested by Gaeta and colleagues (2009). In their approach, they expect to overcome the use of any specific lack of expertise in knowledge engineering (Gaeta et al., 2009). Their focus is on ontologies that specify personalised e-learning experiences that support blended learning activities. This approach uses ontologies to model the educational domain. They also provide details of how to construct e-learning ontologies to optimise definition and implementation of personalised e-learning experiences. Highly efficient and effective e-learning activity is expected by such personalisation and their approach has been implemented in an e-learning platform named Intelligent Web Teacher (IWT).

The authors expect to exploit the ontologies to personalise e-learning experiences with the help of the information retrieved from learner profiles and the contents of the learning object repositories (Gaeta et al., 2009). The learning objects are annotated with semantic information using standard metadata schema. For simplicity, the authors consider that the e-learning experiences are represented by sequences of learning objects. This approach is also considered to be suitable for e-learning experiences with complex flows of learning activities.

IWT is supported by an Advanced Ontology Management System (AOMS) that improves the capabilities of these learning ontologies. The AOMS is used first to construct and validate the

ontology it is annotated with; metadata, indexed and archived. The teachers search the AOMS repository to find the required ontology and import it in IWT. The ontologies built here are in the OWL format.

2.1.14 UICO: An Ontology-Based User Interaction Context Model for Automatic Task Detection on the Computer Desktop

A User Interaction Context Ontology (UICO) has been proposed in (Rath, Devaurs, & Lindstaedt, 2009). It is aimed at enhancing the performance of task detection on the user's computer desktop. Metadata is captured from the user's desktop. In this approach, a User Interaction Context Model is automatically populated by utilizing rule-based, information extraction and machine learning approaches. The relations between the model's entities are automatically derived and the user's tasks are automatically detected. UICO has been developed using Protégé based on OWL-DL and the top-level perspective of the ontology covers the action dimension, resource dimension, information need dimension, application dimension and user dimension.

The UICOs specify the user's interaction context that is originated from context sensors that automatically observe the user tasks on the computer desktop and contextual information is automatically derived. A bottom-up approach has been followed to build the UICO based on a conceptual model. Relations are incrementally added with the addition of the new sensor data or algorithms. The context information held in UICO has been sensed and relates the information that is automatically derived from it. It holds the concepts and the relations between concepts of the conceptual model, and the resource data and metadata that the context sensors capture.

2.1.15 Ontology Design for Creating an Adaptive Learning Path in an E-Learning Environment

Learners need to build adaptive learning paths that allow them to understand curriculum, syllabuses, and subjects of courses in depth (Chung & Kim, 2012). Those ontologies have been developed as a part of an ontology-based e-learning system. These multiple ontologies (Curriculum ontology, Syllabus ontology, and Subject ontology) on different layers have been created, integrated and interfaced in (Chung & Kim, 2012) to allow learners to build adaptive learning paths.

The curriculum ontology organises various semantic relationships between individual courses in Computer Science or Engineering fields. The concepts in the curriculum ontology includes

ProgramOfStudy, Course, KeyConcept, AttainmentGoal, AttainmentLevel, etc. The semantic relationships include *hasSubtype, prerequisiteOf, basicOf, advancedOf, combinedOf*. The curriculum ontology also establishes direct connections with the syllabuses' ontologies.

The syllabus ontology specifies the internal and external structures of courses that are mentioned in the curriculum ontology. It defines a unified vocabulary of syllabuses that helps to compromise the different vocabularies used by different instructors. A syllabus class, which is the core concept of syllabus ontology, has nine data type properties and 12 object type properties to describe the semantic knowledge that has been extracted from the syllabuses.

The subject ontology includes a hierarchical structure of concepts related to a subject. It is composed of teacher-based ontologies, several learner-based ontologies and learning materials. The teacher-based ontologies consist of the learning concepts and knowledge structure that the teachers use in class. Learner-based ontologies consist of the concepts and knowledge structures created by the students. A subject ontology has been proposed to make student learning effective and enhanced.

An architecture that consists of those ontologies, among the other components, has been proposed to support different functionalities (Chung & Kim, 2012). They include:

- Understand the user requests and invoke the appropriate handlers.
- Translate the keywords in the user queries into SPARQL Protocol and RDF Query Language (SPARQL) or Topic Map Query Language (TMQL) queries.
- Collect, parse, and classify the resources (such as syllabus webpages, course description webpages) to create instances of the ontologies.
- Identify and manage the learning outcomes of individual courses and academic areas.
- Use the user requirements to create adaptive learning paths automatically.

This approach attempts to enhance student learning by ontologies. Still, it does not discuss the use of metadata or other techniques for personalisation, ontology language, ontology design tools used, reasoning and query-answering.

2.1.16 User Profiles and Learning Objects for Reasoning and Interoperability in Recommender Systems

A recommendation system is expected to allow reasoning and improved personalised results for users. To achieve this objective, a recommender system infrastructure for educational material which are described with metadata has been proposed in (Primo, Vicari, & Bernardi, 2012). This approach utilises the Agent-Based Learning Objects (OBAA) metadata standard, a Brazilian proposal for agent-based learning objects. Besides, it uses the Friend of a Friend (FOAF) ontology that describes people to find user profiles. OWL is used to describe domain features while a web service is used to connect to the contents repository. Recommendation is based on a Collaborative Recommendation Algorithm.

There are two stages of this approach (Primo et al., 2012): 1. The Recommendation System (RS) algorithm can be Content Based (CB), Collaborative Filtering (CF) or Hybrid. 2. Post-processing of the recommendations. Recommendations are required for any user, with two sets of metadata. 1. User profile metadata. 2. Learning object metadata. The final set of recommendations is made by applying reasoning on user profile metadata and learning object metadata (Primo et al., 2012). This approach uses educational contents such as books, articles, presentations, videos, or physical objects. Metadata that describe the educational contents make them functional. Interoperability between hardware platforms is obtained by the OBAA standard that is technology independent and flexibly used in this approach (Primo et al., 2012). The user model stores information such as usage history of the users (including evaluations on educational content), access logs, and interests. In addition to them, this approach stores and semantically describes information acquired from the users. Post-processing (regarded as reasoning) is done through production rules, case-based reasoning and statistical methods. The reasoning process is the last function before the recommendations are made (Primo et al., 2012).

This system selects the educational materials for a given user profile based on a recommendation algorithm. For example, if the user uses a mobile device the educational materials selected will be compatible to be displayed in the mobile device's screen. The system uses a number of application profiles (APs) to support the reasoning process. The APs are of two types; Metadata Profiles (MPs) and Technical Profiles (TPs) (Primo et al., 2012). MPs help the reasoning process to identify the educational content that matches the device (digital television, Internet, mobile devices) used by people with audiovisual disabilities. The TPs help the reasoner to generate relevant results considering technical aspects of the devices.

For example, file formats, colour and size formats suitable for a TV. These metadata are described in an OWL ontology (Primo et al., 2012).

2.1.17 A Personalised Adaptive E-Learning Approach Based on Semantic Web Technology

An adaptive e-learning system that has the ability to support personalisation based on learners' abilities, learning styles, preferences and levels of knowledge has been introduced in (Yarandi, Jahankhani, & Tawil, 2013). This e-learning system is based on the design of semantic content, learner and domain models to tailor the teaching process for individual learners' needs. The learners' characteristics are captured during their registration. In this approach, the ontological user profile is updated based on achieved learners' abilities. The system is able to recognise changes in the learners' levels of knowledge as they progress. The system is also able to track learners' activities and tests during the learning process and update the learner based on them. The system later uses the updated learner model to generate different learning paths for the learner. Learners' responses to test items are analysed by the system to calculate learners' abilities. Four ontology models are included as part of the system: domain model, user model, content model and test model (Yarandi et al., 2013).

The system uses three ontologies: user model – that describe learners' profiles; domain model - a logical taxonomy for the knowledge domain; and, a content model – that defines the structure of the learning content. The proposed adaptive e-learning system takes into account learners' abilities, learning styles, preferences and levels of knowledge and the ability to support personalisation. The users' learning styles, that is based on the FSLSM (Felder & Silverman, 1998), are recorded as a part of the user model. The achieved learners' abilities are reviewed to update the ontological user profiles. The learning contents are recommended based on the learners' abilities. The ontologies are in the OWL format and some reasoning has been applied in this approach.

The architecture of the ontology-based personalised e-learning system consists of a user interface, adaptive engine, user model mediator and content mediator. The user interface passes the learners' characteristics to the user ontology and presents the learning contents tailored to the learner. The adaptive engine generates learning contents based on the user model. It also evaluates the users' performance, abilities and knowledge to update the users' profile. The user model mediator is responsible for accessing and updating the user model.

The content mediator searches for the instructional objects (IOs) and populates the lesson with IOs and automatically annotates them.

2.1.18 An Ontology based Approach for Modeling E-Learning in Healthcare Human Resource Management

The ontology-based approach for modeling e-learning proposed in (Bajenaru & Smeureanu, 2015) has focused on learning path personalisation related to human resource management in healthcare. In this approach, students are able to receive the learning materials according to their level of knowledge, preferences and interests (Bajenaru & Smeureanu, 2015) based on a contents ontology and a user profile ontology. In their work, a student model, a domain model, and an ontology-based personalised learning path are proposed, targeting the healthcare HR system members.

The personalised model includes three main domains: 1) learning modelling process; 2) student modelling; and, 3) digital content modelling.

This adaptive system uses a student modelling process to create and maintain a student model based on the data collected from various sources. The students' preferences, cognitive style and level of experience are used to provide them with personalised training content.

The domain knowledge in this system is represented at three levels of abstraction. The lowest and the first level is the learning objects (LO). Learning objects are indexed to allow the system to identify LOs and how a specific subject can be used in learning. This information is obtained from the second level of abstraction that is represented by metadata. The third level is the domain ontology that is built with the concepts and relationships between them. The concepts are learning objects that refer to student progress on the concepts and skills specified. Each item describes the concepts and skills specific to a course that the students have to acquire when they complete the course.

A highlight of our work is the use of an ontology that maps students' knowledge in course concepts to provide better access to their progress as well as to customise content and navigation structure of the learning content for individual students.

2.1.19 Ontology-based Smart Learning Environment for Teaching Word Problems in Mathematics

MONTO – a machine-readable ontology for teaching word problems in mathematics, is demonstrated in (Lalingkar, Ramanathan, & Ramani, 2015) and has been proposed to

improve mathematical thinking and problem-solving skills. It includes various ontologies and the key functionalities of this system are derived from the ontologies. MONTO uses four main ontologies: system/pedagogy ontology, strategy/task ontology, user model/student ontology, and domain ontology (Lalingkar et al., 2015).

System/pedagogy ontology is used for solving teaching problems together with domain and task ontologies. The domain ontology means all the conceptual connections between the resources needed by students. The strategy/task ontology in MONTO includes a model of the strategies and tasks that students need in solving a problem. Various interactions between the student and the system are stored in the student model ontology in MONTO. The student model ontology captures information about the student and that is used for displaying the student's learning profile (Lalingkar et al., 2015). These ontologies have been developed using Protégé and are in OWL format. Hence, the reasoning could be done by using the reasoners available in Protégé as plugins.

MONTO has made several contributions. The proposed ontology has a perfect fit with a framework that has been developed for mathematical thinking, i.e., resources, heuristics and control, by Schoenfeld (1985). This ontology can capture the Semantic Knowledge of a word problem and show it to students for reinforcement. Such knowledge extracts show multiple knowledge representations to students, and that helps to reduce their cognitive load. The MONTO ontology uses problem models to represent strategies, and analytical questions, to capture missing concepts and misconceptions, and student models. This ontology helps to identify students' mistakes and provides constructive feedback. As MONTO has many general features other mathematical domains could deploy it.

2.1.20 An Ontology-based Adaptive Personalised E-Learning system, Assisted by Software Agents on Cloud Storage

E-learning has moved from a knowledge transfer model to interactive advanced decision-making abilities (Rani et al., 2015). An ontology driven system that implements the FSLSM has been proposed in their work. The system validates its integration with the Semantic Web environment using some learning content. Software agents play the role of monitoring the learning style of the learners and modifying the learning contents accordingly. Cloud storage is used to store and maintain the ontology, databases and other required server resources. Comparisons are made between the information presented in the system and adaptive learning styles of the learner, and how the agent reacts according to the learner's behaviour

(Rani et al., 2015). The proposed ontology presents the domain with more expressive relationships that are reusable in systems with a similar purpose. These ontologies have been written in OWL, and OWL DL is used as the query language.

A multi-agent architecture has been proposed for their e-learning system in (Rani et al., 2015). DL Query also has been used in their work for efficient extraction of the required information from the application's ontology. They also use the HermiT reasoner to determine the consistencies in *user.owl* and *course.owl* ontologies. Ontologies have been deployed on DigitalOcean's remote Cloud host due to its expanded and secure environment (Rani et al., 2015). These authors have conducted a survey to explore the features of the e-learning system. They have paid attention to adaptiveness and personalisation of the e-learning application with ontology. They have also explored the use of Semantic Web Cloud services, an incremental model and a multi-agent system in recognising adaptability of the learner's behaviour. Even though this approach synergises many contemporary technologies it does not discuss the population of the ontology.

2.1.21 Ontology-based Model for Learning Object Metadata

It is pointed out that most digital repositories implementations are based on distributed computing systems' architectures that deal with major technological and modelling issues (Kalogeraki, Troussas, Apostolou, Virvou, & Panayiotopoulos, 2016). However, such implementations hinder the data accessibility and reusability of heterogeneous databases and interoperability between them. The authors propose that learning Object metadata helps to overcome operational deficiencies if they were used with a content-structure and a systematic approach. They also expect to use semantics in learning materials and inference rules to solve interoperability issues and to support information retrieval.

An ontology-web model for learning Object Metadata has been proposed in (Kalogeraki et al., 2016). Their work detecting semantic relations in educational material, aims to assess, at an abstract level, its tutoring content in order to improve the learning procedure. The knowledge base of examination material has been extended by adding semantic syllabus content. This helps to provide instructors with a tutoring tool to make inferences on several aspects of the educational process. They include student learning, draw conclusions on their learning, and make suggest for further amendments. Question-answering material is integrated into syllabus content that allows extraction of inference results. There is a high possibility that the students are more likely to comprehend and estimate how representative

the assignment material vis-a-vis the syllabus content. It is aimed at enhancing Learning Object Metadata with semantic annotation to help improve tutoring skills of Learning Management Systems and support decision making (Kalogeraki et al., 2016). The EduSor ontology-based Model has been implemented in RDF/xml syntax and that facilitates the data interpretation process between heterogeneous repositories, provides unambiguous concepts and increases system interoperability.

2.1.22 Curriculum, Syllabus and Subject Ontologies

SKOS (a Semantic Web-based vocabulary for knowledge organization systems) has been proposed in (Miranda, Orciuoli, & Sampson, 2016) for modelling subject ontologies and as a standard way of representing domain knowledge. The focus of this attempt is to identify alternative strategies for storing and accessing ontologies related to learning scenarios. These strategies focus on supporting the knowledge sharing, knowledge reusing, planning, assessment, customisation and adaptation processes.

The main purpose of the subject ontologies with respect to other ontologies is that they bridge the different aspects of the educational systems. Subject ontologies are possibly used to semantically organise many elements of an e-learning system. They include: the learning objects that are described by using metadata; the learning activities; learning goals (competences, attitudes, skill or knowledge); assessment objectives; learning events and opportunities; and, people's social profiles (Miranda et al., 2016).

Experiments have been done on two subject ontologies, one big and one small, by executing SPARQL queries. These experiments aim to identify a better strategy that efficiently treats subject ontologies to support Semantic Web-based Educational Systems (SWBESs) and improve the user experiences in learning (Miranda et al., 2016). It is emphasized that the subject ontologies represent a scalable framework able to organise and retrieve learning material within a SWBES that allows the integration of different tools.

2.1.23 Towards Situation-driven Mobile Tutoring System for Learning Languages and Communication Skills: Application to Users with Specific Needs

Based on a literature survey, Khemaja and Taamallah (2016) have identified that in one-on-one learning systems, the content and the learning approach may be easily adapted. Still, the learning/supporting services requiring new development or reconfiguration of the system are not readily available. This has resulted due to the lack of standards promoting unified ways to develop services and hence to reuse them. A new systems' architecture for an adaptable and

reconfigurable mobile Intelligent Tutoring Systems (ITS) architecture has been proposed in (Khemaja & Taamallah, 2016) to overcome the above problem. Their approach allows the users to acquire relevant communication skills in different situations based on their specific needs.

It uses a services ontology to semantically describe bundles and services capabilities:

1) The Context ontology for reasoning on context.

2) The Goals ontology for expressing users' objectives independently from provided services. Goals instances are derived mainly from the context ontology.

3) The Domain ontologies, for knowledge representation and data exchange between the systems components (Khemaja & Taamallah, 2016).

The mobile application architecture of ITS includes two categories of Android components - Android activities and Android services. ITS also allows several services: automatic data collection about the system's behaviour, self-monitoring, reasoning that efficiently helps to draw conclusions and adapt the proposed solution.

In the work presented in (Khemaja & Taamallah, 2016) ontology development has been done using Protégé and reasoning among and querying the ontologies has been done using AndroJena plug-in. This approach focuses on a mobile application instead of a web application.

2.1.24 Personalised Students' Profiles Based on Ontology and Rule-based Reasoning

It is reiterated that gathering information on learning styles by using questionnaires leads to some problems. With their usage, learners are reluctant to answer questions, or they guess the answer which takes some time. Hence, an attempt to build adaptive student profiles by analysing learning patterns, based on rule-based techniques, has been proposed in (Nafea et al., 2016). This approach analyses learning patterns through a learning management system, according to the Myers-Briggs Type Indicator (MBTI) theory, besides the FSLSM.

The learning domain ontology used in the work presented in Nafea et al., (2016) describes learning in a general manner as a domain classification. It identifies the subdomains of learning and elaborates on the learning styles of the learners. Different data about individual learners, including personal, progress and performance data, are captured in a learner profile that is represented as a reference ontology. The reference ontology includes static contents

and dynamic contents. The ontology tool Protégé has been used to develop ontologies in OWL.

The ontologies are used in that work to give perspectives of the learner's learning style based on the way the student uses the system. Personalisation is achieved by comparing the users' profile with the courses offered in the institution Nafea et al., (2016). The users are subsequently provided with suggestions for courses based on data collected from learners' behaviours. This helps to avoid the generation of inappropriate recommendations to the learners. A salient feature of this approach is building adaptive student profiles by analysing learning patterns based on rule-based techniques. An adaptive engine module is a main part of the architecture and plays the adaptation role in this architecture.

2.1.25 A Proposed Paradigm for Smart Learning Environment Based on Semantic Web

A framework for a smart e-learning ecosystem that uses ontology and Semantic Web Rule Language (SWRL) has been proposed in and implemented by (Ouf, Ellatif, Salama, & Helmy, 2017). In this approach, instead of considering personalisation in isolation, an e-learning ecosystem is proposed that integrates the learner, educator and the content designers. A learner model with four separate ontologies—learner model ontology, learning object ontology, learning activities ontology and teaching methods ontology—has been proposed for the personalisation. The proposed model integrates the learner model with the learning process components such as learning activities and teaching methods. There are four layers in the proposed architecture of the e-learning ecosystem: interface layer, semantic reasoning mechanism layer, semantic layer and Semantic metadata layer. SWRL has been used to add a logic layer on the e-learning ontologies to represent the rules in the OWL ontologies in a consistent way.

This approach has several specific features. It attempts to build an e-learning ecosystem that involves the main stakeholders: learners, educators and instructional designers. In addition to personalisation, ontologies are integrated with the learning process and SWRL is used to represent the rules found in the knowledge base (Ouf et al., 2017). It proposes to use SWRL as a part of the e-learning contents personalisation process.

2.2 Discussion and Future Research

In this section, we first provide a summary of the literature survey that we conducted and highlight the important features and characteristics of those approaches. A snapshot of each approach is given in Appendix: Table A.1 to show the state-of-the-art ontology-based e-

learning systems. We also discuss the directions that these approaches have led to and the possible further research directions.

2.2.1 Summary and Discussion

Ontology-based e-learning has been a popular research area that has drawn the attention of many scholars. Limitations of XML-Schema language in representing the IMS Learning Design (IMS LD) specification has been identified as a problem in (Amorim et al., 2006). They have considered designing a learning design ontology as a way to solve those limitations.

Different research problems and issues in e-learning and ontology-based e-learning systems have been highlighted in different scholarly work. (Dutta et al., 2009) propose a conceptual framework of semantic e-learning to overcome the problem of lack of shared understanding between terms of various metadata vocabularies (Stojanovic et al., 2001).

Metadata elements are mainly useful in indexing the documents but they lack formal semantics (Dutta et al., 2009). Therefore, the lack of shared understanding between terms of various metadata vocabularies is expected to be avoided by using ontologies as a conceptual layer.

It is claimed that there is a lack of methodologies and techniques for effective ontology management; in particular, addressing the issues related to ontology versioning, ontology harmonization and collaborative ontology construction (Gaeta et al., 2009).

Two challenges in e-learning have been observed in (Rani et al., 2015) during their exploration of recent efforts. Firstly, incorporating the e-learning systems effectively in the Semantic Web environment. Secondly, achieving adaptive personalisation with the learners' changes in their behaviours.

A reasoner is used to check the consistence of an ontology. However, it could be affected by the limitations of the expressiveness and reasoning capabilities of the ontology languages (Amorim et al., 2006). OWL is used to represent instructional theories and their models that include rules in (Sicilia, Lytras, Sánchez-Alonso, García-Barriocanal, & Zapata-Ros, 2011). Then, the SWRL rule language is used for checking the compatibility of learning designs with instructional theories. Many approaches have been proposed to address different research problems and issues that have been presented here. Each approach includes

numerous common and distinct characteristics and features. In general, these research studies have taken clearly identifiable directions.

Personalisation of e-learning systems – dispels the notion of 'one-size fits all'. Personalisation of the e-learning systems and their contents has initially happened with the use of learning object metadata. The different metadata used in these approaches have been borrowed from the popular metadata standards, IEEE LOM, IMS and Dublin Core. Eventually, many approaches have given considerable attention to identifying learners' requirements for user profiles and adaptating e-learning systems based on these profiles. Whilst conducting this literature review, we discovered that this topic is probably the most popular one in e-learning research. Use of user profiles and the use of metadata have been used as an important way to personalise e-learning systems. We also noticed a recent trend of adaptation of the learning contents by analysis of the learning styles or learning patterns.

Ontologies for the learning domain – Ontologies have been proposed to overcome the weaknesses in the use of metadata for personalisation. Ontology has helped to add a conceptual layer that helps to describe the domain concepts (learning materials) using both human and machine-readable semantics. Ontology has been used as a key component in ontology-based e-learning systems. Over the past decades, different types of ontologies have been deployed in the e-learning systems. The type of the ontology has varied fundamentally based on the contents of the ontology. These types include user ontologies, contents ontologies, course ontologies, context ontologies, and task ontologies. Most researches have focused on creation of curriculum or syllabus ontology, organising learning objects based on ontologies, and retrieving the learning contents based on ontology. Besides the creation of ontologies, the aim of some research has been integration and interfacing multiple ontologies that are in different layers of an e-learning system (Chung & Kim, 2012). Among the most salient references, adaptation modules receive attention. Some of the components that are used in the architectures of these approaches have already been developed and are available commercially or are open sourced, such as reasoners.

Architectures for ontology-based e-learning systems – Different ontology-based e-learning systems have been proposed with different architectures that include peculiar elements and elements that are common to many ontology-based e-learning systems. A typical architecture of an e-learning system includes user interfaces for the learners, different services or tasks that a user could perform, and storages. The type of the user interfaces, the services they provide and the storages of the architectures varies in many ways. These

architectures include modules or components to provide several typical services: learning contents management, personalisation of the learning contents, reasoning and query-answering being among them. The repositories include learning object or material repositories, metadata repositories and ontology repositories. They also have followed different standards of metadata, ontology languages, and query languages.

Reasoning and e-learning – When it comes to usage of the ontologies, reasoning on them helps to filter the required instances from an ontology, based on the learners' requirements. Reasoning on e-learning systems is achieved by an automated reasoner and using diverse reasoning techniques. The capabilities of a reasoner depend on the expressivity and the DL language it is based on. Different reasoning tools have been used. For example, Racer Pro, Hermit, Pellet. Protégé has been a popular tool among the ontology research community. It is mainly used for ontology design, development and editing. It also provides additional services with the help of plugins. For example, reasoners and query interfaces are used as plugins in Protégé.

Ontology languages – Another critical feature of ontology approaches is the use of metadata and ontology languages. Metadata are really useful to elaborate the learning resources and ontology languages are useful in implementing the whole ontology. Different metadata standards that are widely used include IEEE LOM, CanCore LOM, Dublin Core and ADL SCORM (Roy, Sarkar, & Ghose, 2010). The standard ontology languages include RDF, RDF Schema, ADML+OIL, and OWL.

Modelling techniques – Ontology models are developed to represent and visualise an ontology of a domain. Ontology models help to visualise the concepts and relations within a learning domain. Ontologies are mainly modelled using topic maps and UML class diagrams.

Information source for ontology learning and ontology population – To build an ontology, the concepts and relations of the interested domain need to be identified. Gathering details of concepts and relations can be done from different information sources. Once the ontology is built, it has to be filled with instances. If these are done manually from electronic documents, it becomes a time-consuming process. Hence, some ontology development approaches attempt to automate the process of creating an ontology. 'The process of defining and instantiating a knowledge base is referred to as *knowledge markup* or *ontology population*, whereas (semi-)automatic support in ontology development is usually referred to as *ontology learning'* (Buitelaar et al., 2003). During ontology learning, while some

approaches use text document collections as a source of information, others use the Web (web pages) and existing databases as sources of information. The table A.1 in Appendix summarises all the approaches we presented in the previous section.

2.2.2 Further Research

We hold a view that there is a considerable duplicated effort in the design and development of ontology-based e-learning systems for each institution. A lot of time and effort is spent by each institution to design and develop an ontology-based e-learning system. As the literature highlights, this demands expert knowledge which most institutions lack. Again, we see that effort in design and development of ontologies for the learning institutions is duplicated. Hence, if we acquire more insight into learning ontologies, we could simplify the design and development of ontologies for the learning domain.

Again, many system-specific e-learning systems have been proposed that focus on personalisation of learning contents. Yet, an architecture for ontology-based e-learning systems that could be adapted at different institutions would benefit educational knowledge acquisition. Such an architecture could also use the data in legacy databases and be able to adapt the learning contents effectively. In line with that, we identify the following research problems:

1. What system architecture is required to adapt an ontology-based e-learning system at different institutions?
2. What are the representation and reasoning capabilities in a formal ontology required for the e-learning domain?
3. How does an ontology-based e-learning system work with a legacy databases?
4. How can the effectiveness of e-learning ontologies be evaluated?

Chapter 3: A Plug and play Framework and Architecture for Ontology-based Adaptive E-Learning Systems

In our literature survey that is elaborated in Chapter 2, we observed numerous approaches in designing and developing ontology-based e-learning systems. However, we observe some duplicated efforts in those approaches. The reason behind that is different educational institutions have institution-specific requirements and due to their differences and the competition among them, each of them prefer to keep their e-learning systems unique. Again, in another study that we conducted on unit guides, we also observed terminological differences in the unit descriptors of computing departments of different Universities in New South Wales (NSW), Australia.

All the scholarly works unanimously agree on the fact that the learning contents need to be conceptualised and personalised for learners. For this reason they have proposed approaches for adaptating e-learning systems. There have been many attempts made, and different techniques used, to personalise e-learning systems. Suggested personalisation techniques use numerous underpinning technologies and metadata standards, distributed computing methodologies, case-based reasoning techniques, models on learner learning patterns and styles, etc. Each of these techniques has its own benefits and most approaches found in the literature use one or more of those techniques.

An adaptive engine with an assessment unit is used as the dominant component in the systems architecture proposed in (Yarandi et al., 2013). In that approach the adaptive engine interacts with a content mediator and a user profile mediator during the adaptation process and the learners view the adaptive contents on a user interface. The content mediator refers to the LO repository (domain ontology and the content ontology), whereas the user profile mediator refers to the user profile repository (domain ontology and user ontology). Again, in (Nafea et al., 2016) an adaptive engine is used as the dominant component for adaptation of one learning system for users. The adaptive engine uses the results of a questionnaire and the results of users' behavior that is obtained from a reference ontology, for personalisation. To enable this, e-learning systems include many components in their architecture which perform different tasks.

The ontology-based architecture proposed in (Chung & Kim, 2012) includes three layers: semantic services, components and repositories layer (Figure 3.1). The services layer includes the user functions: syllabus and subject search, and editing the learning paths that can be performed by a learner in a user interface. The components layer includes the ontologies and the software components to perform the system tasks. The systems' architecture proposed in (Rani et al., 2015) organises the components on three levels: human level, Internet level and system level. The human level includes the user and a workstation, while the Internet level includes a web portal. The system level includes three main services, three agents and Cloud storage with a user database and the domain ontologies. The web portal interacts with the services and the agents in the system level. The services in the system level also interact with the user database and the domain ontology (Rani et al., 2015).

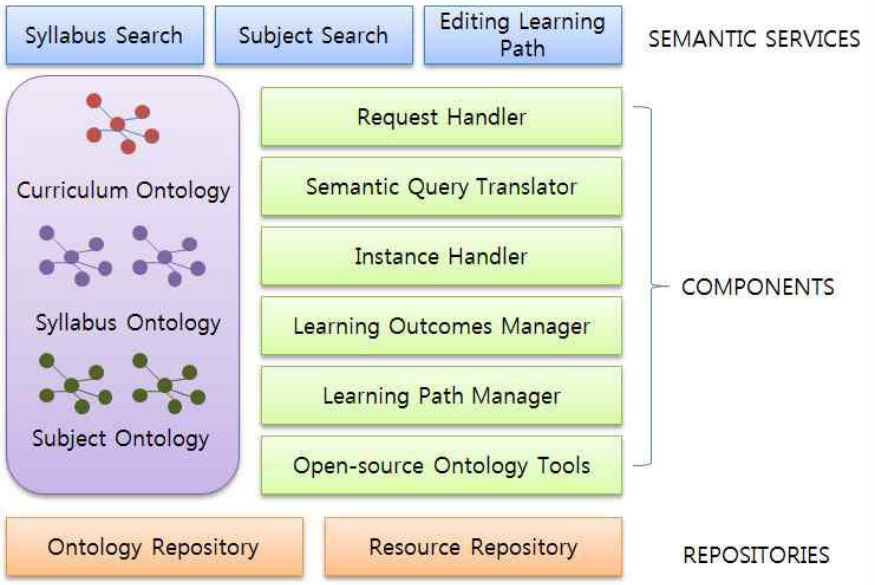


Figure 3.1: Ontology-based system architecture proposed in (Chung & Kim, 2012)

In this chapter, we propose *a plug and play architecture* that we visualise as a synergy of several techniques: metadata standards, learning styles and, especially, domain ontologies for the personalisation of learning contents. We also utilise many different components to perform different tasks within a system predominantly searching for learning resources. We also use storages of materials that are required for learning. We organise all these components of the system in three layers. We propose to make the top two layers of an e-learning system common to any institution and the bottom layer specific to an institution. However, if the system is to be deployed at an institution, all the layers should adapt to that

institution. We propose to achieve that by a plugin the ontologies (in the repositories layer) to the middle layer of the system.

The system architecture proposed in (Chung & Kim, 2012) has been aimed at supporting self-leading learning of students by building adaptive learning paths using the curriculum, syllabuses, and subjects' ontologies. However, it does not use the reasoning capabilities offered by DL reasoners. Chung & Kim (2012) focus on the adaptability of the system only at the user level, not at an institutional level. Also, they do not talk about using profiles for the users' adaptation of the system. In contrast, the *plug and play architecture* we propose has the benefits offered by the DL reasoners and the adaptability of the system. The adaptability the system for the user is achieved with the help of user profiles; and the adaptability by the institution is achieved based on institution-specific learning ontologies pluggable to specific system instances.

The expected outcomes of our *plug and play architecture* are:

- An adaptive e-learning system's framework that is deployable at many institutions will be developed.
- For the use of the adaptive system at institutions, institution-specific and pluggable ontologies that specify the learning domain of each institution will be deployed.
- Query-answering and reasoning will be allowed on the existing data in the learning databases and study materials, by using mapping and transformation of learning databases to learning ontologies.

Our approach is aimed at an ontology-based e-learning system's framework that is adaptable at different institutions. It also becomes a solution to the duplicated effort in designing and developing ontology-based e-learning systems and associated critical components. In this chapter, we provide a comprehensive description of the proposed architecture and its layers. In the next section we give an overview of the systems architecture and our approach to realise that. The approach we follow for the development of an ontology-based adaptive e-learning system has two main tasks: development of a learning ontology, and mapping a legacy database to learning ontology. Once they are done, a user is able to interact with the system. Literally, we can see the *plug and play architecture* in operational use only when the adaptable e-learning system is built based on to it. We describe the ontology development tasks and the systems' architecture in a sequence. In Section 3.1, we provide an overview of the ontology development and mapping system; and the *plug and play architecture* for an

adaptive e-learning system's framework. Section 3.2 introduces the ontology development task and Section 3.3 introduces a legacy database to an ontology mapping task. Section 3.4 introduces the end users' interaction and the plug and play architecture. Section 3.5 provides a discussion.

3.1 An Overview of the Plug and Play Architecture

In this section, we provide an overview of the ontology development and mapping system and the *plug and play architecture* of the proposed adaptive e-learning systems' framework that can be deployed at any institution with a domain-specific learning ontology (Figure 3.2).

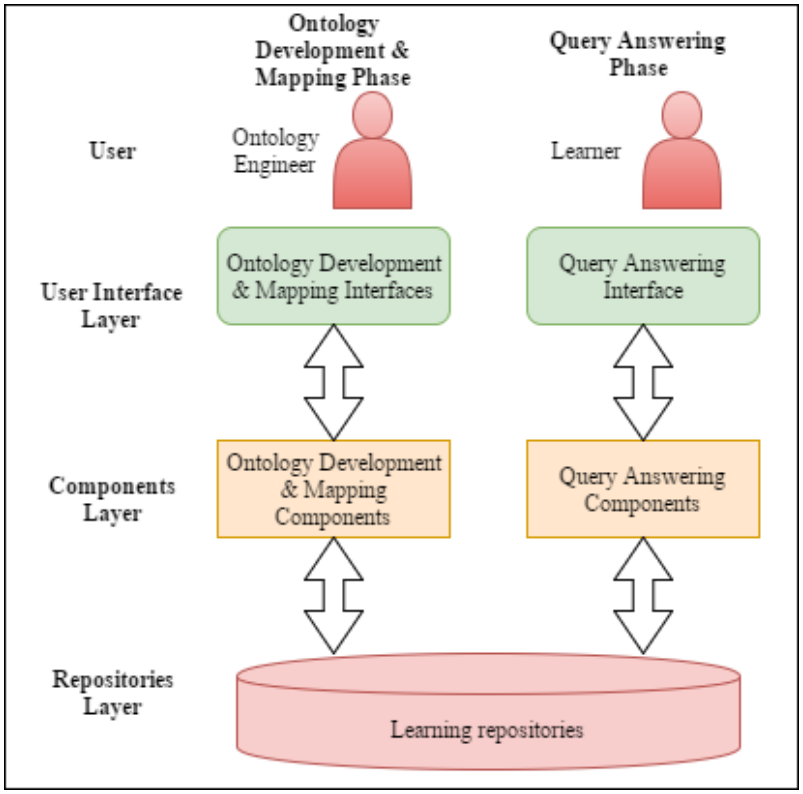


Figure 3.2: High level view of the plug and play architecture

At a higher level, the proposed architecture consists of three main layers: user interfaces layer, the components layer and the repositories layer. Also, there are interfaces between layers that pass information and controls between those three layers (Figure 3.2).

The main component that makes this systems architecture plug and play is the institution-specific learning ontologies that are a part of the repositories layer. The learning ontologies are developed for a specific institution to capture the domain knowledge of that institution. The learning ontologies are plugged in to an instance of this system built according to the

proposed architecture. The learning ontologies then adapts the system to that institution and personalises the system to the learners of that institution.

3.1.1 Ontology Development and Database to Ontology Mapping System

The development and mapping of the learning ontologies are done as two main tasks: 1) ontology development and 2) database to ontology mapping. Once these tasks are completed end users can interact with the adaptive learning system. Each system that is required to perform these tasks requires different interfaces, components and the repositories.

Learning Ontology Development

In order for the learners to use the adaptive e-learning system the required repositories of the system, the learning ontology and the learning materials should be available. This task includes primarily activities that are required to be carried out to develop the learning ontology before it is populated with the instances. The ontology engineer uses the ontology development interfaces to design the ontology. During that process the ontology engineer adds the concepts, subconcepts, roles, role constraints and data properties to the ontology and edits them. Different systems' components assist performance of the ontology development tasks, gradually building the ontology in the background and checking whether the ontology is consistent. Then, the resulting ontology is stored in the repository layer.

Legacy Database to Learning Ontology Mapping

The ontology engineer uses the database to ontology mapping interfaces to map the database schema of the legacy databases to the Terminological Box (TBox) of the learning ontology, and the data in databases to the Assertional Box (ABox) instances of the ontology (Baader & Nutt, 2003). To start this process the ontology engineer views the ontology and the database schema of a particular institution that are retrieved from the repositories layer. With the understanding of the database schema and the ontology, the ontology engineer writes the mapping rules. Mapping rules are used to specify what contents of the database should be mapped to what part of the ontology (Hazber, Li, Gu, & Xu, 2016). These tasks are supported by the ontology mapping components in the components layer. The mapping definitions are stored in the repositories layer that is used for mapping the database(s) to the ontologies, and the resulting ontologies with the instances are stored back in the repositories layer.

3.1.2 The End User Interaction and Plug and Play Architecture

Once the ontology is built and populated with the instances it should be available for learners' use. For that, an instance of the ontology-based adaptive learning system is adapted with the institution-specific learning ontologies that are plugged in to the system. That makes the learning ontologies ready for the learners to use for querying. For example, consider that an instance of the e-learning system is deployed at Macquarie University (MQ) with the ontologies (metadata, learner profiles and learning objects) specific to Macquarie University. When a learner logs on to the system his/her profile is loaded from the learner profile ontology. The learning concepts that are required to initiate searching for resources are loaded to the system from the learning ontology with the help of the metadata ontology.

The main task in end-users' interaction is to query the learning ontology or to search for the learning resources. A high-level view of the plug-and-play architecture for the end users' interaction is given in Figure 3.2. The learner uses the query interface in the query interface layer to search for the learning materials. The components in the components layer perform the required tasks to retrieve the details of the learning object from the ontology and then the actual learning objects.

For example, consider that a learner at Macquarie University wants to find 'what learning materials should be studied to answer the ISYS114 assignment 1?' To start this process, the learner composes the query on the query interface and submits it to the system. Then, the components in the components layer see what learning concepts, relations and attributes are involved in the query and include them in a query. The components in the components layer also see what preferences the learner has in his or her profile, what relevant metadata of learning objects should be checked and include them as well as in the query. Now, the relevant components process the query by requesting the relevant learning objects that satisfy the search criteria and that the metadata matches the learner preferences. The components in the components layer produces the query results and displays them on the query interface in the user interfaces layer.

In the next subsections we elaborate on the learning ontology development and database to ontology mapping tasks that are required to be completed before user interaction using the ontology-based adaptive e-learning system. We introduce the systems architecture for ontology development, database to ontology mapping and the *plug and play architecture* of the adaptive e-learning system with which the user interacts.

3.2 Learning Ontology Development

The ontology engineer who performs the ontology development and who has the required expert knowledge is the key user in this task. Primarily the ontology engineer directly interacts with the ontology development interface. However, components in the three layers of the system are involved in the ontology development process (Figure 3.3) and perform different tasks. In developing the learning ontology, the ontology engineer passes instructions to the ontology editor in the user interface layer, and views the ontology that is being developed. The components in the three layers interact with each other by passing controls and information.

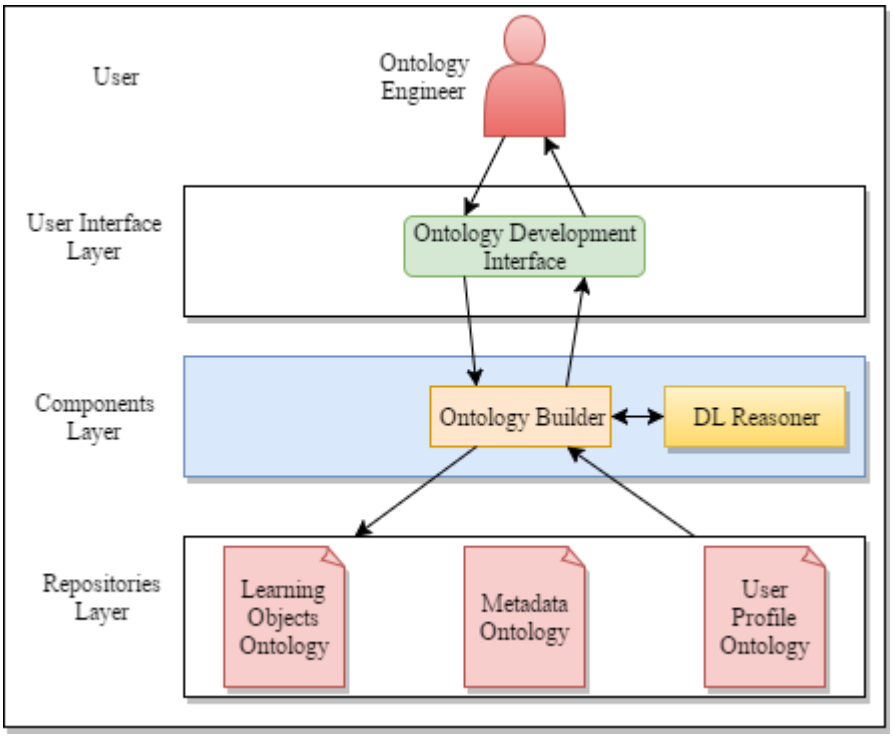


Figure 3.3: A simple systems architecture for the ontology development

3.2.1 User Interface Layer

This layer includes ontology development interfaces that are required to develop an ontology. These interfaces allow the ontology engineer to create the elements of the ontology and graphically present those elements on the interface. They also allow the ontology engineer to edit and delete elements of the ontology. The main tasks of these interfaces include creating and editing the learning concepts and concept hierarchies, relations or roles between the learning concepts, properties of learning concepts and instances of the concepts and roles.

3.2.2 Components Layer

The components in this layer receive the information and controls from the ontology development interfaces. Based on them, the components in this layer perform the due tasks. They include creating and editing the ontology components which reside in the components layer. Also, the components in this layer work as an intermediate layer and handle the communications between the interface and the repositories layers.

Ontology Builder – This component involves performing the tasks related to adding and editing the concepts, relations, attributes and constraints of the ontology. It also allows for inserting instances of the concepts into the ontology. Even though the ontology is represented graphically on the interface the ontology manager represents the ontology in an ontology language. Over time the ontology languages have evolved and we follow the contemporary ontology language recommended by W3C, OWL 2 (Motik et al., 2012). Once the ontology is validated by the DL reasoner, the ontology manager stores that in the repository.

The ontology builder is also responsible for storing the ontology and its elements in the learning ontology (learning objects ontology, metadata ontology and user profile ontology) in the repository layer. It also retrieves the learning ontology and its elements that would be required by the ontology editing interface.

DL Reasoner – The reasoning tasks on the ontologies are performed by a DL reasoner (Horrocks, Sattler, & Tobies, 1999). There are many DL reasoners available, such as RacerPro, HermiT, Pellet, and Quest Reasoner (Rodrıguez-Muro & Calvanese, 2012). According to this architecture the main reasoning task performed at this stage is checking the consistency of the ontology based on the DL language used by the reasoner. Protégé provides many reasoners as options that are made available as plugins.

3.2.3 Repositories Layer

The learning ontologies that are generated in the ontology development process are stored in the storages that belong to the repository layer. The learning ontology that we consider in our work is specific to an institution and specifies the domain knowledge specific to an institution. Therefore, when the system is deployed at a different institution a new learning ontology specific to that institution should be created and plugged to the learning system. A knowledge base (= an ontology) developed for the learning domain can include knowledge about the courses, students, learning strategy, learning objects, and metadata. A knowledge base consists of a TBox that specifies the domain concepts with their roles and an ABox that

specifies the instances. A *learning ontology* we consider here consists of three main ontologies: the *learning object ontology*, the *learning objects metadata ontology*, and the *users' profile ontology*. Each of these ontologies capture the knowledge specific to the learning domain and that includes the domain specific concepts, relations, attributes and constraints.

Learning Objects Ontology – *Learning objects ontology* holds concepts and relations that correspond to the learning objects and an institution's attributes. The learning objects are different learning resources (lecture slides, lecture notes, demonstrations, etc.) and assessment items. In our work, we identify the relationships between the concepts and constraints on them from the educational policies and procedures that are mentioned in the unit guides and the course hand book. They make the TBox of the knowledge base. The instances of the learning concepts and the roles among the instances make the ABox of the knowledge base. We get the instances from the legacy databases available at an institution.

User Profiles Ontology – This holds several general and specific characteristics of the learners. The user profile includes users' credentials (id, name, password, etc.) and several user preference attributes: preferred resource type, preferred resource format, preferred level, preferred difficulty level, etc. The learning system uses them for the purpose of adaptation of the system according to the learner. The user profile ontology includes the concept *Student* and the concept *preference attribute*. The different types of user preference attributes: *preferred resource type, preferred resource format, preferred level, preferred difficulty level*, etc. are gathered from the learner when he/she logs on to the system.

Metadata Ontology – The *LOM ontology* holds metadata about the learning objects and constraints on them. The LOM also plays a role in the process of adaptation of the system for the users. The different metadata we use in our work is a subset of the IEEE LOM (Roy et al., 2010). They include language, format, difficulty level, intended end users, and type of learning objects.

In this work, we use the ontology editing tool Protégé which provides an advanced user interface for developing an ontology. Protégé also provides several example ontologies such as the pizza ontology that is elaborated in (Rector et al., 2004).

3.3 Legacy Database to Learning Ontology Mapping System

Research has been completed on the database to ontology mapping and transformation processes (Mogotlane & Fonou-Dombeu, 2016). As a result of that a number of mapping and

transformation tools have been proposed. Recent mapping and transformation tools include Relational Databases to Ontologies Transformation Engine (RDOTE) (Vavliakis, Grollios, & Mitkas, 2010), KARMA (Knoblock et al., 2012), D2RQ, Virtuoso, DBOWLizer (Villanueva-Rosales, 2011) and the Protégé plugin Ontop (Bagosi et al., 2014). These mapping tools use mapping rules to facilitate the mapping process. RDOTE generates an instantiated OWL ontology using a relational database and an ontology schema as the inputs (Vavliakis, Grollios, & Mitkas, 2010). Protégé Ontop allows us to write the mapping rules or build them using a mapping assistant (Calvanese et al., 2017). Karma provides graphical assistance to build the mapping rules (Knoblock et al., 2012). It shows how table names with primary keys match against concept names, foreign keys to object properties, and attributes to the data properties. A number of mapping rules that are used in these tools are based on the mapping rules that have been proposed by W3C.

In this proposed architecture, we use an existing mapping interface. We found that Karma serves our purpose and we use the mapping interface Karma to map a learning database to a learning ontology. It allows mapping and transforming of tables, their keys and data in them to ontology concepts, object properties and data properties using a graphical user interface. Also, the Protégé plugin Ontop (Calvanese et al., 2015) provides OBDA without populating the ontology with instances.

In this section, we elaborate on the architecture of a typical database to ontology mapping system. This architecture is also described based on the three layers: interface layer, components layer and the repositories layer as shown in Figure 3.4.

3.3.1 Interfaces Layer

The interfaces layer of the architecture in the database to ontology system is utilised by the ontology engineer. The mapping interface includes three sections; ontology viewing interface, database viewing interface, and mapping interface.

Mapping Interface – This is used for three main purposes. Firstly, a mapping interface is used to load and to view the schema of the learning database, the tables and the attributes. This helps the ontology engineer to examine the ontology and understand the elements in it. Secondly, this interface is used to load and view the learning ontology and its elements that are to be populated from the data in the learning database. This helps the ontology engineer to examine the database schema and understand the elements in the database.

Finally, the mapping interface helps the ontology engineer to create mapping rules. This is the main interface used to create the mapping rules. After viewing the elements in the ontology and the database schema, the ontology engineer maps each matching element in the ontology with the elements in the database schema. Firstly, an ontology concept is mapped to the database table and secondly, each data property is mapped to an attribute. Thirdly, each object property is mapped to a relationship. The resulting mapping rules become visible to the ontology engineer. In addition to that the mapping interface helps to populate the ontology and view the resulting instances by opening the ontology file.

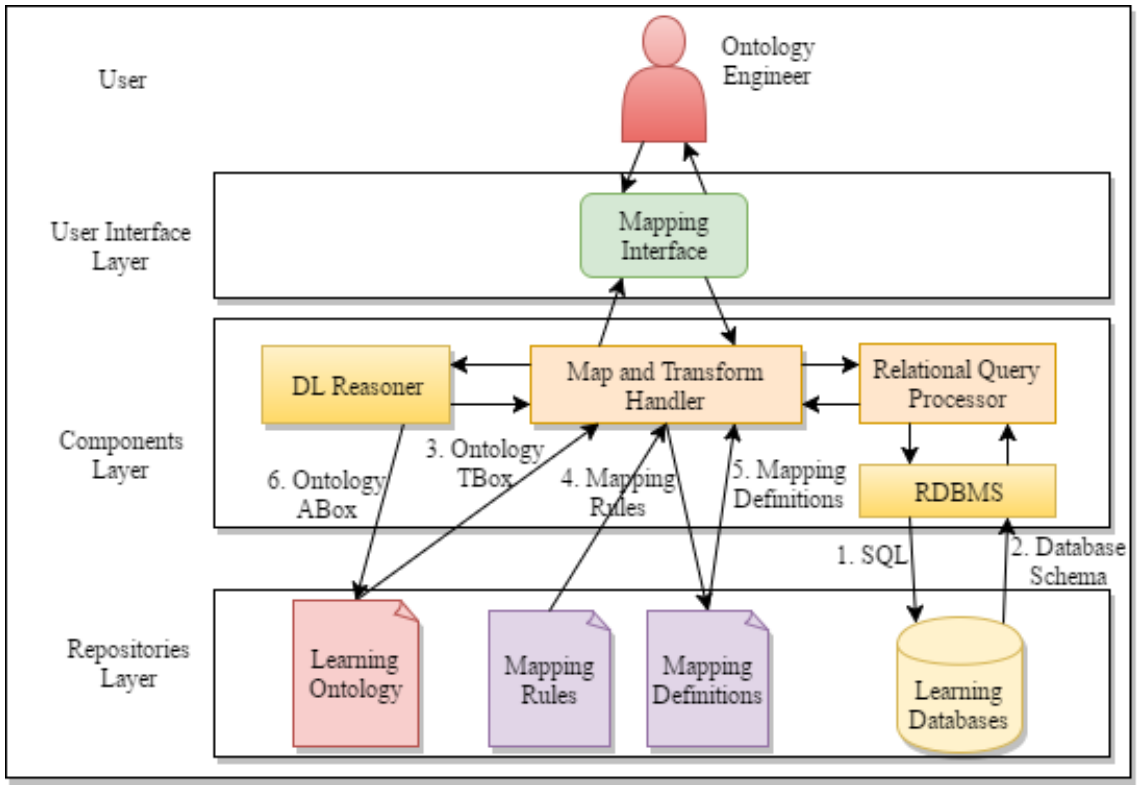


Figure 3.4: Systems architecture for the database to ontology mapping system

3.3.2 Components Layer

The Components in this layer are used to perform the tasks related to mapping the database and the learning ontology and populate the ontology by the data in the learning database into the instances, roles and constraints of the learning ontology. Figure 3.4 introduces the map and the transform handler as two subcomponents that perform two separate tasks.

Database to Ontology Mapping Handler – The *Database to Ontology Mapping Handler* first gets the instructions from the *mapping interface* to load and display the learning ontology. According to them it loads the ontology displays that are on the screen. The *Database to Ontology Mapping Handler* also gets the instructions from the *mapping interface*

to load and display the learning databases. According to them it loads the database schema and displays them on the screen. Importantly, it does database to ontology mapping that is based on different mapping rules. Mappings done by the ontology engineer are verified by the system based on these mapping rules. For example, a database table should be mapped to an ontology concept. The *Mapping Handler* gets the details of a database table and an ontology concept at the time of mapping from the mapping interface. The resulting mapping definitions are stored in an ontology definitions file.

Once the database to ontology mapping is completed, the ontology is populated with the instances from the data and keys in Relational Database (RDB) tables. Karma allows transforming relational data into ontology instances. In contrast to that, Protégé Ontop uses Ontology-Based Data Access (OBDA) (Calvanese et al., 2015) to allow querying the data in the database tables without storing them in an ontology ABox. The mapping and ontology population are elaborated in Chapter 5.

DL Reasoner – During the database to ontology mapping process, different reasoning tasks on the ontologies are performed by a DL reasoner to ensure the consistency of the resulting ontology. The reasoner checks whether the mapping rules defined by the ontology engineer are consistent with the ontology before it is populated with data. If there are any inconsistencies, the reasoner would notify those discrepancies that are noted to the user interface through the map and transform handler.

Relational Query Processor – The *Relational Query Processor* gets the name value pairs to form the queries from the *Map and Transform Handler*. The SQL queries are generated and then passed to the *RDBMS* and it returns the query results back to the *Map and Transform Handler*. The returning query results are of two types. Firstly, the database schema is returned to be used in the mapping process. Secondly, the data is returned to be used in the ontology population. The relational query processor obtains all these query results from the learning databases through *RDBMS*.

Relational DBMS – The relational query processor passes all the SQL queries to the RDBMS. The RDBMS retrieves the database schema for queries related to mapping and data for queries required in the ontology population process.

3.3.3 Repositories Layer

Four main components are involved in the mapping and transformation process. They are the legacy learning databases, learning ontologies and the mapping definitions. The critical features of each of them are briefly introduced here.

Legacy Learning Databases – Many different databases are used at an educational institution to store different details related to an institution. Among them *student databases and learning databases, course and unit databases, timetables and schedules databases* are prominent. Student databases hold personal information of students, their enrolments to study programs and information related to their learning progress. Learning databases hold learning resources of units of study and data about the learning resources. These databases include information that is specific to an institution. Hence, a learning system that is deployed at an institution holds data specific to that institution. In addition to them, user profiles and learning object metadata can also be found in the databases. All these learning databases are related to the other existing systems of an institution and those databases are populated and updated by those systems. In this work, we suggest obtaining access to them from the ontology-based adaptive e-learning system.

Learning Ontologies – Here we utilise the three learning ontologies (learning object ontology, metadata ontology and user profile ontology) that are generated in the ontology development task and populated in the database to ontology mapping and transformation task that is explained in Section 3.2 and Section 3.3.

Learning Objects Repository – The learning objects such as lecture slides, lecture notes, videos, sample exams, tutorials, etc. are stored in the learning objects repository. The learning materials are related to the course units of degree programs. They are generated by the academic staff of an institution before the delivery of lessons. The learning database holds the titles of them as a reference to them. The learning ontology is also a reference, the URLs of the resources that enable retrieving the specific learning resource from the repository.

Mapping Definitions – The mapping definitions that are created by the ontology engineer in the mapping interface are stored here. Once the mapping definitions are written they are saved as a separate file. The files with the mapping definitions are eventually used by the map and transform handler. Protégé Ontop allows creating, updating and viewing the mapping definitions on an interface and they are eventually saved as an OBDA file (Calvanese et al., 2017).

3.4 The Learning Ontology at Work

Once the ontology development and databases to ontology mapping and transformation are completed, the learning ontology becomes ready to use. For the usage of the learning ontology we suggest the *plug and play architecture* that is introduced in this section. Before the learners use the system, the system should be adapted to an institution with the learning ontology specific to that institution.

3.4.1 Adaptation of the System to an Institution

The *plug and play architecture* and the end user interaction it offers are common to any instance of the adaptive e-learning system deployed at an institution. The domain knowledge of the learning system varies according to the learning ontology plugged to an interface of the learning system. Hence, the plug and play ontology plays the sole role of adaptation of the learning system to a particular institution and its users. The adaptation of the system happens at a higher level according to the ontologies that are plugged in to the system. For example, if we plug in the ontologies of the <University 1> the system presents the learning resources and makes the system available for the <University 1> learners. Whereas, if we plug in the ontologies of the <University 2> the system presents the learning resources and makes the system available for the <University 2> learners. However, the advantage of our *plug and play architecture* compared to the other architectures is that we don't have to change or customise the interface layer and the components layer.

Once the adaptive e-learning system is adapted to an institution the learner is able to use it for querying the ontology to search for the learning resources. An overview of the *plug and play architecture* of the ontology-based adaptive e-learning system is given in Figure 3.5. As introduced in Section 3.1 it consists of three layers: interface layer, components layer and repositories layer. The query interface is the main component in the interface layer. The components layer includes the software components that perform the different tasks related to querying the learning resources backed by the learning ontology. In the following subsections we briefly discuss the components in each layer.

3.4.2 Interface Layer of the Plug and Play Architecture

This layer of the *plug and play architecture* consists of an interface for updating the user profiles and for querying the ontology. The learner is allowed to enter or update the user profile attributes on the user profile interfaces. Also, when the user logs on to the system, the user interfaces receive the user credentials and pass them on to the input/output handler.

The user query interfaces are used to first present the concepts and roles in the ontology to the learner and to compose a query. Then the learner is able to select the relevant learning concepts, roles and constraints of concepts in the ontology that are displayed to the learner. The interface allows the user to compose a query, avoiding the need for the learner to encode them in a formal notation.

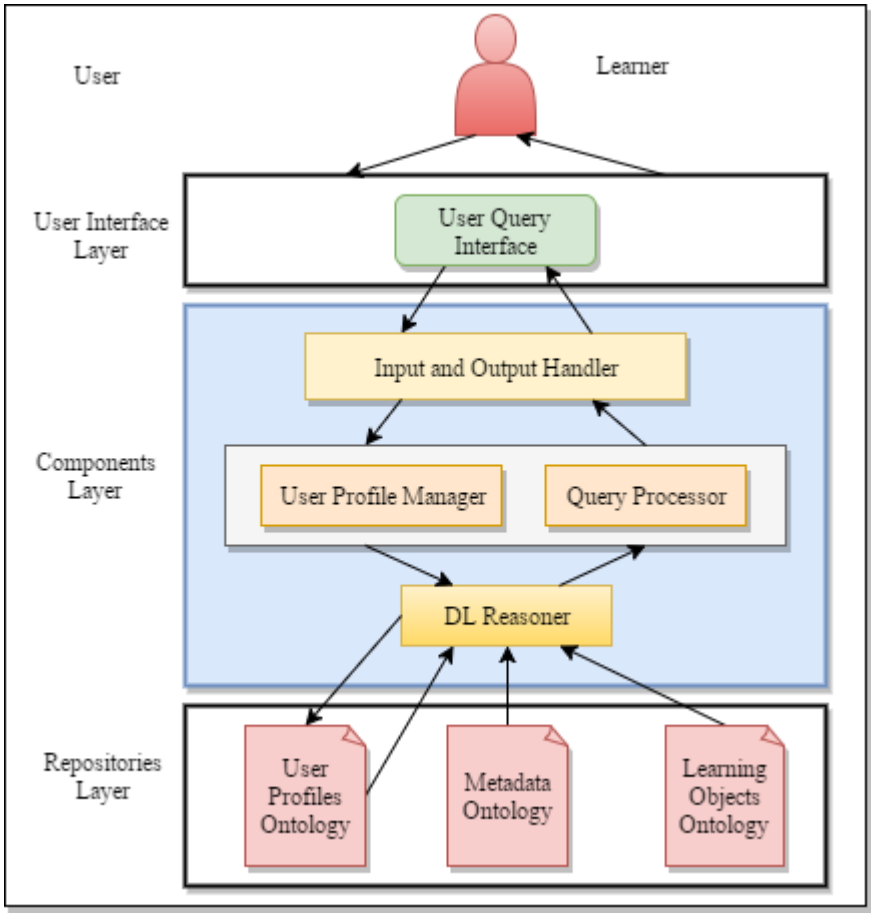


Figure 3.5: Ontology-based plug and play e-learning systems architecture

A learner composes a query by including multiple query atoms. A query atom consists of the triple: <subject> <role> <object>. The user selects a concept as a subject, then a role and another concept as the object. This is repeated if the learner wants to include several query atoms in a query. For example, a student at Macquarie University could compose a query to search for the lecture slides of the unit ISYS114 by including the query atoms (1) and (2).

ISYS114 isA Unit (1)

ISYS114 hasLectureSlides <LectureSlides> (2)

When a query is composed, the query atoms are then passed to the input output handler. The answer to a query is also displayed on the query interface. The answer could include a number of instances of a concept, possibly with several data properties. Below are two identifiers (3) of the instances of ISYS114 Lecture Slides that would be displayed on the query interface as the answer to the above query.

$$ISYS114LectureSlidesWk1, ISYS114LectureSlidesWk2 \quad (3)$$

Then, the user is able to retrieve the actual learning resources by clicking on the identifiers. An ideal query interface would allow the user to enter a query as free text. Hence, we have provided a textbox in the query interface as an option for future enhancements.

3.4.3 Components Layer of the Plug and Play Architecture

This layer includes components that support answering learner's queries. In following subsections we introduce each of them briefly.

Input/Output Handler – The *input/output handler* receives the users' queries as a collection of name-value pairs related to query atoms. For example, attribute name and value of it or concept name and value of it or instance name and value of it, or role name and value of it. The *input/output handler* then validates these values and passes them to the *user profiles manager* or *the query processor* as name-value pairs. The *input/output handler* also receives the query results, the instances and their attributes, from the *Query Processor* and displays them eventually on the query interface.

User Profile Manager – The users' profile manager is responsible for creating and updating the users' profiles. It receives the validated name-values pairs from the input/output handler that are related to the users' profiles. Then it composes the relevant queries according to the syntax rules of a query language such as new RacerPro Query Language (nRQL) or SPARQL, and passes the queries to the DL reasoner. It also receives the query results that are passed back to the input/output handler.

The *user profile manager* also receives the validated user credentials as name-value pairs from the *input/output handler*. Then, the *user profile manager* uses those credentials to identify the user and composes queries to retrieve and identify the user preferences. The user preferences are passed to the DL Reasoner to retrieve the initial learning concepts and accepts the facts about the learning resources returned by the DL reasoner. Then the query interface is populated with the possible concepts which the learner would query.

Query Processor – The query processor gets the input values related to the learner's queries as name-value pairs from the input/output handler. Then, it composes a complete query including the input values triples and passes the query on to the *DL Reasoner*. The query processor also accepts the query results from the *DL Reasoner* and passes them to the *input/output handler*.

DL Reasoner – The DL reasoner receives the learner's queries from the *user profile manager* and the *query processor* as a set of query atoms in the triple format, <subject role object> and logical constructors such as and, not, or, etc. The DL reasoner then performs the reasoning tasks and retrieves the relevant concepts and/or instances that are returned to the *user profile manager* or the *query processor.*

The queries that are composed are in a query language understandable to the DL reasoner. The query language to be used also depends on the DL reasoner. The most commonly used query language in the Semantic Web is SPARQL. The query language new RacerPro Query Language (nRQL) is used in the DL reasoner RacerPro.

3.4.4 Repositories Layer

The repositories layer includes the learning ontologies that resulted from the database to ontology mapping task. These ontologies include references to the learning object repositories as well.

Learning Ontology – The three ontologies; learning object ontology, metadata ontology and the user profiles ontology which constitute the learning ontology, are the key components in the repositories layer. They hold the domain knowledge as facts that are required for query-answering.

Learning Objects Ontology – The learning objects ontology has references to the actual learning objects in the learning objects repository. Based on the query results the identifiers or URLs of the learning resources are displayed on the user query interface. The learner is able to retrieve the actual learning resources using these identifies/URLs.

User Profile Ontology – The *user profile ontology* is generated in the ontology development task and it holds the general details of the learners and preferences of the learners. They are retrieved and updated by the *user profile manager* and used by the query processor during query-answering.

Metadata Ontology – This is also developed in the ontology development task to specify the metadata items of the learning objects. The metadata are used in query-answering to provide personalised contents to the learner. In the following section we elaborate on the personalisation process.

3.5 Personalisation with User Profiles and Learning Object Metadata

In this systems architecture, we use the user profiles and LOM for the purpose of personalising the system contents to the learner. We use a set of user preference attributes and LOM attributes. The preference attributes of the learner are gathered from the learner when the learner first logs on to the system and the learner is allowed to update them on the learner's request. The LOM attributes are created at the time of ontology development and populated at the time of database to ontology mapping and transformation.

Different metadata standards are used in e-learning systems and they include: IEEE LOM (IEEE, 2002), IMS consortium (IMS, 2001), Sharable Content Object Reference Model (SCORM) Metadata and Dublin Core (DCMI, 1990) Metadata standards. These different metadata standards include similar metadata attributes; although, some differences exist among them. In our work, we achieve some basic personalisation by using a subset of IEEE LOM. We also use a set of user preference attributes that matches with the selected metadata attributes.

3.5.1 User Profile Attributes

In a typical e-learning system the learning objects that are retrieved for a given query include both the learning objects required by the learner and these are not required by the learner. For example, a learner searches for learning resources of ISYS114. The query results for this could include presentations, lecture notes, demonstrations, etc. that could be in different formats such as pdf, power point, text, audio and video. However, the learner could have expected only the presentations in the power point format. To satisfy this learner requirement we utilise learner preferences. We store the preferences of the learner as user preferences in user profiles. Then, we use those user preferences in query-answering.

The learning objects to be retrieved for a given user depend on the preferences of the learner. In our context, additional information about the users (user profiles) are collected. User profiles consist of user preference attributes given in Table 3.1.

To answer the learner's queries we need to identify the learning resources that match the learner's preferences. For that reason, we have to keep details of the learning objects that match the learner's preferences. The LOM attributes that we use in this work describe the learning objects of a specific institution.

Table 3.1: Learner preference attributes

User Preference Attribute	Description
preferred language	the language of the content
preferred topics of interest	topics of the study area
preferred education type	lecture, presentation, assignment, etc.
preferred format	text, image, video, graphics, etc.
preferred education level	level 1, level 2, level 3
preferred difficulty level	very easy, easy, difficult, very difficult

The metadata that we use are listed in Table 3.2. We derived this short-listed subset of IEEE LOM based on the domain, context and how this knowledge could be combined with the information about the user profiles. For example, suppose that a user is interested only in presentations in a unit. The content types of the learning objects are represented by the LOM item content type. Hence, the search involves checking whether the learning objects have the content type *presentation*. During the ontology development task, these different user profile attributes and the LOM are represented as *OWL data properties* within the learning ontology.

Table 3.2: LOM attributes used in the plug and play architecture

Metadata Item	Description
language	The language the learning object is written in. Eg: English
subject area	The subject area for which the learning object was developed.
format	The format of the learning object. Eg: Text, image, audio, video, graphics
education type	The type of the learning object. Eg: lecture, presentation, assignment, etc.
difficulty Level	How difficult the learning object is. Eg: very easy, easy, difficult
intended user level	level 1, level 2, level 3

3.5.2 Personalisation and Querying the E-Learning System

As mentioned previously, personalisation of the e-learning system to the learner is done based on the user profiles and the LOM attributes. The steps in this personalisation process are depicted in the Figure 3.6. The personalisation of the e-learning system happens firstly by

personalisation of the initial interface settings. When a learner logs on to the e-learning system the user preference attributes are also loaded from the user profile ontology. Secondly, the system loads the concepts and learning objects that are relevant to the learner and displays them on the query interface.

During query-answering, the learner is able to select a learning concept and then the properties of the selected learning concepts. The learner is also allowed to select instances of the concepts that are displayed on the screen. For example, if the learner selects the concept *Learning Resource* to be queried, the roles of it are displayed to the learner. If the learner selects the role *learningResourceOf*, then the learner is able to see and select a *Unit* associated to the learning resources. Thirdly, if the learner selects the unit *ISYS114* and submits the query the system retrieves the LOM from the learning ontology. Fourthly, the system matches the user preferences with the LOM attributes returned from the ontology. If a match between the LOM attributes and the learner preferences is found, finally the system retrieves the learning objects that match the query and the user preferences.

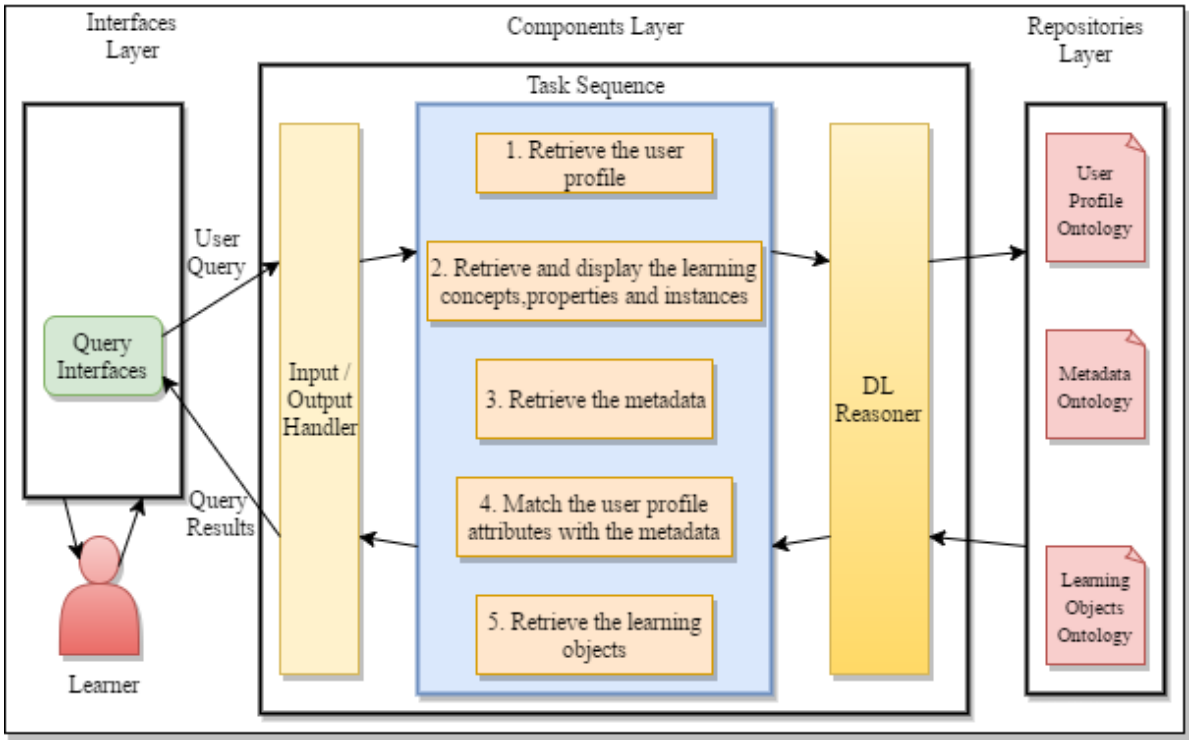


Figure 3.6: Personalisation of the e-learning system based on ontologies

3.6 Discussion

We observe a number of benefits of the proposed ontology-based systems architecture although, changes are required in implementing this architecture. Below we discuss those benefits and the changes so that we might define solutions to overcome those challenges.

We also see how practical and advantageous this solution is in comparison to the challenges that we have to overcome.

3.6.1 Benefits Offered by the Plug and Play Architecture

The primary advantage of the proposed architecture is that it avoids the duplicated effort in adapting an ontology-based e-learning system at different institutions. It allows replicating the system and implements an instance of the system at an institution that does not require an effort to build or to customise the learning system. It is capable of automatically adapting the system to an institution based on an institution-specific learning ontology.

Further, it has the advantage of personalisation of the learning contents to the user by using learner preferences and metadata. It also provides higher query-answering capabilities to search for the learning resources that is supported by the learning ontology and reasoning.

We propose to utilise the legacy e-learning databases in this architecture that avoids the need for populating the learning ontology from scratch. Due to its layered nature, this architecture could further be extended with additional components on a layer to provided additional service and enhanced features. For example, this architecture allows adding components to the three layers to manage the metadata or to enhance the personalisation with the study patterns of the learners.

We utilise the legacy data in the legacy databases by transferring them into instances in the learning ontology. Again, this needs time, money and expert knowledge.

Even though the ontologies used in this architecture uses formal structures, the query interfaces suppress the complexity of the formal ontology structures by presenting them as simple interface components on the query interface.

Due to the use of an automated database to ontology mapping tool, it becomes possible to update the learning ontology on a regular basis or on demand, or possibly this could be automated. This helps us to provide up-to-date information to the learners.

3.6.2 Challenges in Implementing the Plug and Play Architecture

An instance of the system is deployed at an institution. At the time of this system deployment an institution-specific ontology is required to be plugged in to the system and to support personalisation of learners' queries. The design and development of an institution-specific ontology demands expert knowledge and skills which are scarce. Again, three ontologies (learning object, metadata and user profile ontologies) should be designed by studying the learning domain of an institution that is a time-consuming process. We attempt to alleviate this problem by proposing an ontology sublanguage for the learning domain that is elaborated in Chapter 4.

This architecture has the advantage of using the data in legacy databases and learning objects that have already been developed for an institution. On the other hand, to utilise these data they should be mapped and transformed to the ontologies. Again, that needs time and expert knowledge. However, not only mapping rules should be defined; they should have to be modified in case of changes to the learning databases. To overcome this problem, we attempt to use an existing tool to transform the data in the learning databases to instances of the learning ontology. We elaborate this process in Chapter 5.

Also, when the data stored in the databases changes the ontology instances in the ABox need to be updated. This could be done automatically on a regular basis or that task can be avoided by using a tool that uses ontology-based data access (ODBA) (Calvanese et al., 2017). An ODBA tool allows access to the data in a database through an ontology without transferring them to the ABox of an ontology and building a fixed ABox.

For the learners to understand the domain, and to query it, we conceptualise the domain on the user interfaces. The user interface shows the concepts, relationships between the concepts and the constraints on those relationships. However, the ideal way of doing this is to first show the conceptual view of the domain knowledge to the users and then to allow them to write queries in text. For this we need a *text to semantic query converter*. We do not address this research problem at this time.

3.7 Conclusion

In this chapter, we provide an overview of the proposed *ontology-based plug and play architecture* for an adaptive e-learning system that uses a pluggable leaning ontology as the salient component. This architecture is a result of our attempt to answer the question: what systems architecture is required to adapt an ontology-based e-learning system at different

institutions? Instances of the system built based on the *plug and play architecture* are possible to be deployed at different institutions with a leaning ontology that is specific to an institution. An instance of the system adapts to an institution with the institution-specific learning ontology and personalises the learning contents to the learners in that institution.

We elaborate on our approach to the *plug and play architecture* and institution-specific learning ontologies in two main tasks: ontology development and legacy database to learning ontology mapping task and the end user interaction. We also point out the benefits and several challenges that have to be overcome in implementing the plug and play architecture. Overall, we suggest that this architecture offers many more benefits than the challenges that we have to overcome. In the next few chapters we elaborate on the solutions we propose to overcome those challenges and to gain further benefits from the proposed architecture.

Chapter 4: OWL 2 Learn Profile: An Ontology Sublanguage for the Learning Domain

Many experimental ontologies have been developed for the learning domain for use at different institutions. These ontologies include different OWL/OWL 2 (Web Ontology Language) constructors. However, it is not clear, what are the representation and reasoning capabilities in a formal ontology required for the e-learning domain? Also, which OWL 2 constructors are the most appropriate ones for designing ontologies for the learning domain? It is possible that the constructors used in these learning domain ontologies match one of the three standard OWL 2 profiles (sublanguages). To investigate whether this is the case, we have analysed a corpus of 14 ontologies designed for the learning domain. We have also compared the constructors used in these ontologies with those of the OWL 2 RL profile, one of the OWL 2 standard profiles. The results of our analysis suggest that the OWL 2 constructors used in these ontologies do not exactly match the standard OWL 2 RL profile, but form a separate subset of OWL 2 which we call OWL 2 Learn.

An ontology is a conceptual specification of a domain that represents concepts, relations and constraints of that domain. A well-designed learning ontology helps to clearly represent a learning domain and eventually could be used to search for learning resources. Moreover, the use of a learning ontology in an e-learning system could assist students who ask questions and search for learning resources by using the domain knowledge specified in the ontology. Different learning ontologies have already been designed for various purposes for some higher educational institutions.

Early attempts on learning ontologies focused more on conceptual modelling of learning ontologies. For example, a topic map ontology for e-learning has been proposed by Kolås (Kolås, 2006) to share learning resources. UML (Unified Modelling Language) diagrams have been used by Knight et al. (2006) who propose an ontology-based approach for adaptive and flexible learning. Some of the learning ontologies that have been proposed later have focused more on ontology design, using design tools and the Web Ontology Language (OWL) (Hitzler, Krötzsch, Parsia, Patel-Schneider, & Rudolph, 2012). OWL and OWL 2 have been used to specify various aspects of learning ontologies. For example, a learning

ontology has been used to measure the semantic relevance between a learning resource and the learning context of a learner (Yessad, Faron-Zuckerc, Dieng-Kuntzb, & Laskri, 2011). In their work, concept maps have been used initially to model an ontology and then OWL to build the ontology for measuring the semantic relatedness for relevance ranking of learning resources. OWL/OWL 2 based ontologies have also been used to recommend the contents in a tutoring system (Vesin, Ivanović, Klašnja-Milićević, & Budimac, 2013). The ontologies (ontologies for learning resources, tasks, learner model and teaching strategy) used in their work have been developed using the ontology design tool Protégé. An OWL-based ontology for teaching mathematical word problems has been proposed in (Lalingkar et al., 2015). These studies show the increasing popularity of OWL/OWL 2 for implementing learning ontologies.

In a recent study, the Higher Education Reference Ontology (HERO) developed in OWL (*HERO_ONTOLOGY_V 25.06.2013.owl*) has been proposed to overcome the problems in building application-specific ontologies in the higher educational domain (Zemmouchi-Ghomari & Ghomari, 2013). This study has found that the development and interoperability of application-specific ontologies are difficult. Therefore, we believe that the identification of a set of common features (OWL/OWL 2 constructors) in existing ontologies of the learning domain might help ontology designers in their work. However, we could not find any studies in the literature that attempt to identify a common set of OWL 2 constructors for the learning domain.

The W3C has recommended OWL 2 as a standard ontology language for the Semantic Web, which is based on a particular version of DL (Hitzler et al. 2012). The W3C has also recommended three standard profiles: OWL 2 EL, OWL 2 QL, and OWL 2 RL that are targeted at different application areas (Motik et al., 2012). Each standard OWL 2 profile includes a subset of OWL 2 constructors and has different computational properties (Motik et al., 2009).

To the best of our knowledge, so far no one has investigated how well learning ontologies are aligned with any of these three standard OWL 2 profiles. This chapter aims at answering this question and identifying a common subset of OWL 2 constructors (a sublanguage or a profile) for the learning domain. In our study, we have first collected and analysed a corpus of 14 learning ontologies that have been developed in OWL/OWL 2, including one in RDF/RDFS. When we consider the ontologies in our corpus, we can find a lot of object and/or data properties that require an expressive version of DL. Hence, one could presume

that the OWL 2 RL profile is a good starting point for modelling the learning domain. However, if it is the case that the learning ontologies in our corpus have different features, then it would make sense to propose a new OWL 2 profile for the learning domain.

We expect that modelling a learning domain requires different institution-specific ontologies whose expressive power depends on specific applications. For example, one of the ontologies in our corpus, the university ontology (*university.owl*[2]), is based on a highly expressive DL language. This ontology includes nominals (individual names) and cardinality restrictions (counting quantifiers) which increase the expressivity of the underlying DL language. On the other hand, another ontology in our corpus, the university benchmark ontology (*uni-bench.owl*[3]), does not include nominals or cardinality restrictions. That means that the university benchmark ontology is based on a less expressive DL language than the language of the university ontology. This is not surprising, since these two learning ontologies have been designed to satisfy different institutional requirements resulting in different ontology structures and features.

In our study, we have first collected a corpus of 14 ontologies designed for the learning domain, including several developed in OWL 2. We then identify and analyse the usage of OWL/OWL 2 constructors in our corpus and compare these identified constructors with those of the OWL 2 RL profile. We observe that not all the constructors in the OWL 2 RL profile are used in our corpus. Finally, we introduce the resulting new profile called OWL 2 Learn and investigate its expressivity.

The rest of this chapter is structured as follows. In Section 4.1, we introduce ontology languages and the three standard OWL 2 profiles. In Section 4.2, we discuss the corpus of 14 learning ontologies and provide an analysis of the corpus. In Section 4.3, we discuss the findings of the analysis of the ontology corpus. In Section 4.4, we present a comparison of the constructors found in the corpus and the constructors of the OWL 2 RL profile. In Section 4.5, we introduce the new OWL 2 Learn profile and discuss its expressive power. In Section 4.6, we summarise our contribution and discuss future work directions.

[2] http://rpc295.cs.man.ac.uk:8080/repository/browser

4.1 Ontology Languages and Ontology Language Profiles

RDF does not include sufficient constructors to specify a comprehensive ontology. RDFS, a schema language for RDF, provides a framework for describing application-specific classes and properties (Horrocks & Sattler, 2001). However, OWL superseded RDF/RDFS in 2004 as a Web ontology language and is now a W3C recommendation for the Semantic Web. As OWL is based on a version of description logic, it allows the use of a DL-based reasoner to derive information that is not explicitly specified in an OWL ontology (Horrocks and Patel-Schneider 2011). Since 2009, OWL 2 has been used as the W3C recommended ontology language for the Semantic Web (Motik et al., 2009). OWL 2 is a new and more expressive version of OWL, which mainly improves the relational and datatype expressivity of the language.

4.1.1 OWL, OWL 2 and Their Expressivity

The expressivity of the underlying DL language is a distinct feature of an ontology language and is determined by the type of constructors that are allowed in the language and how these constructors can be combined. Over the last two decades, the main focus of DL research was to increase the expressive power of DL languages and to understand their formal properties (Baader & Lutz, 2010). Highly expressive ontology languages include many types of different constructors. However, high expressivity comes at a price and query-answering over expressive ontologies can be computationally expensive.

The ontology languages RDF/RDFS, OWL and OWL 2 show a gradual increase in expressive power. OWL includes a range of constructors and axioms that provide a higher expressivity than RDF/RDFS. OWL 2 includes a number of extensions to OWL such as new constructors for expressing additional restrictions and characteristics of properties and property chains and keys (Motik et al., 2012). Self-restriction *ObjectHasSelf()* is one of them. OWL includes only three constructors for non-qualified cardinality restrictions as shown in (4) below whereas OWL 2 includes constructors for both non-qualified and qualified cardinality restrictions as shown in both (4) and (5) below.

$$(\mathit{ObjectMaxCardinality(n\ R)},\ \mathit{ObjectMinCardinality(n\ R)},\ \mathit{ObjectExactCardinality(n\ R)}) \tag{4}$$

$$(\mathit{ObjectMaxCardinality(n\ R\ D)},\ \mathit{ObjectMinCardinality(n\ R\ D)},\ \mathit{ObjectExactCardinality(n\ R\ D)}) \tag{5}$$

OWL 2 also includes different constructors for object properties and data properties. For example, OWL includes a single constructor *rdfs:domain()* to specify both the object property domain and the data property domain whereas OWL 2 includes two separate constructors *ObjectPropertyDomain()* and *DataPropertyDomain()* to specify the two domains.

OWL includes the constructor *DisjointClasses(C_1 C_2)* to specify disjoint classes. In addition to the above, OWL 2 introduces two constructors *DisjointObjectProperties()* and *DisjointDataProperties()* to specify disjoint object properties and disjoint data properties respectively. OWL 2 also introduces the constructor *ObjectPropertyChain()* to help specify property chains and the constructor *HasKey()* to define unique keys (Motik et al., 2012). OWL 2 includes extended datatypes; for example, *owl:real* and *owl:rational*. OWL 2 also provides additional features on data types that include datatype restrictions, range of datatypes, datatype definitions, new datatypes, and data range combinations. Data ranges can be combined by means of intersection, union and complement. Another new feature of OWL 2 is punning, that is, using the same name for a class and an individual or for properties and individuals or classes and object properties.

Even though OWL includes property assertions, it does not distinguish between object and data property assertions. For example, OWL uses the constructor *samePropertyAs(PN a_1 a_2/v)* for both object and data property assertions. On the other hand, OWL 2 includes two separate constructors *ObjectPropertyAssertion(PN a_1 a_2)* and *DataPropertyAssertion(R a v)* for object property assertion and data property assertion, respectively.

It has been shown that OWL has the expressivity of the DL language *SHOIN(D)* (Horrocks et al. 2003). OWL 2 is more expressive than OWL as it supports complex property inclusion axioms. It also includes new constructors to gain syntactic freedom; for example, it allows ontology designers to use *DisjointUnion* and *DisjointClasses* to express disjointedness in a more compact way (Golbreich, Wallace, & Patel-Schneider, 2009). Overall, OWL 2 has the expressivity of the DL language *SROIQ(D)* which is strictly more expressive than the DL language *SHOIN(D)* (Horrocks, Kutz, & Sattler, 2006).

4.1.2 OWL 2 Standard Profiles and Learning Ontologies

In recent years, research on DL based ontology languages has paid an increasing attention to identifying sublanguages to specify different types of application domains that require restricted expressivity. OWL profiles include subsets of OWL 2 constructors and are

designed for particular applications and reasoning tasks. As we can see, each OWL 2 standard profile has been recommended for specific types of applications. We could not find any works in the literature that discuss the applicability of OWL 2 standard profiles to the learning domain. Therefore, as a starting point, it is worth analysing the OWL 2 constructors used in the proposed learning ontologies and investigate the required expressivity of the DL language which can be used to model this domain.

4.2 Characteristics of the Corpus of Learning Domain Ontologies

The corpus of the learning ontologies that we collected for our analysis consists of 14 ontologies (Table 4.1). Twelve of these ontologies are publicly available and have been developed by researchers for use at different institutions. In addition to these 12 ontologies, the corpus includes two ontologies that we have developed for Charles Sturt University (CSU) and Macquarie University (MQ). This section describes the characteristics of this corpus.

Table 4.1: The corpus of learning ontologies

#	**Ontology File Name**	**Institution**
1	university.owl	Manchester University
2	univ-bench.owl	Lehigh University
3	AIISO schema-20080925.owl	Talis Information Ltd
4	swrc_v0.3.owl	University of Karlsruhe
5	TMDU.owl	Tokyo Institute of Technology
6	HU.owl	Tokyo Institute of Technology
7	TITech.owl	Tokyo Institute of Technology
8	ecs.owl	University of Southampton
9	AcademicInstitute.rdfs	University of Aberdeen
10	lom.owl	University of Alcala, Pontifical University of Salamanca
11	HERO_ONTOLOGY_V 25.06.2013.owl	M'hammed Bouguerra Boumerdès University
12	instOntology.owl	Indian Statistical Institute
13	CSU_Ontology.owl	Charles Sturt University
14	MQ_Ontology.owl	Macquarie University

4.2.1 RDF/RDFS and OWL/OWL 2 Ontologies

We identified a number of learning ontologies in OWL/OWL 2 format and in RDF/RDFS format in the open domain. Only one RDF/RDFS ontology (*AcademicInstitute.rdfs*) was included in the corpus. This might be because RDF/RDFS is not very expressive and has

been outdated by OWL/OWL 2 in recent years. We came across many other learning ontologies discussed in research papers; however, they were not included in the corpus, because we do not have access to the full ontologies.

4.2.2 Syntax of OWL 2 Ontologies

The ontologies in our corpus use three different syntaxes: OWL/XML syntax, OWL functional-style syntax and Turtle syntax. Although the choice of the syntax provides some flexibility for the ontology designer, it makes searching for OWL/OWL 2 constructors in the corpus difficult.

For our analysis, we searched for OWL/OWL 2 constructors in all these three different syntaxes. The most commonly used syntax in our corpus is based on OWL/XML. This makes the ontologies machine-readable but also very verbose. For example, the *university.owl* ontology uses the constructor *rdf:subClassOf(C_1, C_2)* to state that an artificial intelligence student (*AIStudent*) is a computer science student (*CS_Student*) as in (Figure 4.1).

```
<owl:Class rdf:about="http://www.mindswap.org/ontologies/debugging/university.owl#AIStudent">
<rdfs:subClassOf
rdf:resource="http://www.mindswap.org/ontologies/debugging/university.owl#CS_Student"/>
</owl:Class>
```

Figure 4.1: A class defined in OWL/XML syntax

The same statement can be expressed more concisely in the DL notation (6) or in the OWL 2 functional-style syntax (7):

$$AIStudent \sqsubseteq CS\_Student \tag{6}$$

$$SubClassOf(:AI\_Student\ :CS\_Student) \tag{7}$$

Similarly, the *uni-bench.owl* ontology uses the constructor *ObjectProperty(C_1, C_2)* to state that a person has got a degree from a University as in (Figure 4.2). This statement can also be written in functional-style syntax as shown in (8) below:

$$degreeFrom(:Person\ :University) \tag{8}$$

```
<owl:ObjectProperty rdf:ID="degreeFrom">
<rdfs:label>has a degree from</rdfs:label>
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdf:resource="#University"/>
</owl:ObjectProperty>
```

Figure 4.2: An object property defined in OWL/XML syntax

In the following discussion, we present our examples in the DL notation or in the OWL 2 functional-style syntax.

4.2.3 Use of RDF Constructors in OWL /OWL 2 Ontologies

All ontologies of the corpus include OWL/OWL 2 constructors. In addition, some ontologies include RDF/RDFS constructors as well. RDF/RDFS constructors are used to specify domain information that refers to RDF resources (Carroll, Herman, & Patel-Schneider, 2012). For example, some named classes are defined with the help of the OWL 2 constructor *owl:Class*. Some named classes that are RDF resources are defined with the help of the RDF/RDFS constructor *rdfs:Class*. In addition to *rdfs:Class*, a number of additional RDF/RDFS constructors have been used in some ontologies; for example, *rdf:DataType, rdfs:subClassOf, rdfs:subPropertyOf*, etc. The reason for this is that Semantic Web languages are layered (W3C, 2013) and those at a higher level borrow constructors from those at a lower level to avoid redundancy (Carroll et al., 2012).

4.3 Analysis of the Corpus and Findings

This section provides a detailed discussion of our analysis of the corpus. In order to identify the OWL/OWL 2 constructors in the corpus, we used a pattern matching program. In the following, we discuss the usage patterns of OWL/OWL 2 constructors in the corpus and the DL expressivity of the corpus.

4.3.1 The Usage Patterns of OWL/ OWL 2 Constructors in the Corpus

Different OWL/OWL 2 constructors have been used within the corpus to express different aspects of the learning domain. We first identified whether each OWL/OWL 2 constructor was used in the corpus or not. If a constructor was used, then we recorded its location and its frequency.

In this analysis, we used two main metrics to measure the usage of the constructors: a score and a frequency. Score (n) is the number of ontologies within the corpus where a given OWL 2 constructor is used. Frequency (f) is the percentage of the use of a given OWL 2 constructor

within the corpus. Frequency is calculated as (the total number (sum) of occurrences of a given constructor within the corpus)/(the total number of occurrences of all the constructors in the corpus)*100. Both metrics are required, because a particular constructor may have been used a lot in a large ontology whereas the same constructor may have not been used much in a small ontology (Power & Third, 2010). For example, the constructor *disjointClasses(C_1 C_n)* has a frequency of 8.85% (sum=544) whereas the constructor *dataProperty(P_1 P_n)* has a frequency of 16.79% (sum=1032) (Table 4.2). However, the score for both constructors is 9, since they occur in the same number of ontologies. We have observed that most of the constructors that have higher scores also have higher frequencies.

Table 4.2: Commonly used and infrequently used OWL/ OWL 2 constructors in the corpus

Commonly used OWL 2 constructors						
#	**OWL 2 Constructor/s**	**DL Syntax**	**DL Example**	**n**	**f**	**sum**
1	Class(C)	A, C, D	CS_Student	14	12.27	754
2	SubClassOf(C1 C2)	C ⊑ D	CS_Student ⊑ Student	14	12.41	763
3	ObjectProperty(P)	P	advisorOf	12	10.85	667
4	DisjointClasses(C1 C2) DisjointClasses(C1 Cn)	$C_1 \sqcup \ldots \sqcup C_n$	AI_Student ⊔ HCI_Student	9	8.85	544
5	rdfs:DataType(), DataProperty(D), DataTypeProperty()	D	hasTenure Boolean, NonNegativeInteger	9	16.79	1032
6	SubObjectPropertyOf(P1 P2), SubPropertyOf(P1 P2)	H - Role hierarchy	doctoralDegreeFrom ⊑ degreeFrom	8	5.60	344
Infrequently used OWL 2 constructors						
#	**OWL 2 Constructor/s**	**DL Syntax**	**DL Example**	**n**	**f**	**sum**
7	SomeValuesFrom(P C), ObjectSomeValuesFrom(P C)	∃P.C	∃hasAdvisor.PhDStudent	7	1.59	98
8	AllValuesFrom(P C), ObjectAllValuesFrom(P C)	∀P.C	∀takesCourse.CS_Course	6	1.40	86
9	owl:Thing	⊤	Class: Thing	7	1.09	56
10	ObjectIntersectionOf(C1…Cn), IntersectionOf(C1…Cn)	$C_1 \sqcap \ldots \sqcap C_n$	CS_Department ⊓ hasResearchArea.AI	6	1.27	78
11	MaxCardinality(n R D), MinCardinality(n R D), ExactCardinality(n R D)	(≥nR) (≤nR)	≥3 takesCourse.CS_Course ≤ 1 takesCourse.CS_Course	6	1.16	71
12	EquivalentClass(C1…Cn), EquivalentClasses(C1…Cn)	$C_1 \equiv C_2$	AI_Academic ≡ CS_Department ⊓ hasResearchArea.AI	6	0.86	53
13	ObjectUnionOf(C1…Cn), UnionOf(C1…Cn)	$C_1 \sqcup \ldots \sqcup C_n$	owl_SemanticLink ⊔ oc_SemanticLink	6	0.57	35
14	InverseObjectProperties(P1 P2), InverseOf(PN)	P^-	advisorOf $^-$ ≡ hasAdvisor	8	0.46	28
15	ObjectHasValue(P a), DataHasValue(R v), hasValue(), ObjectOneOf(a1...an), oneOf	∃R.{x}, $\{x_1,\ldots, x_n\}$	∃hasResearchArea.{AI}, {A26, A27}	4	0.23	14
16	TransitiveObjectProperty	P transitive role	SubOrganisation of is a transitive role	4	0.18	10

Each OWL/OWL 2 constructor used in an ontology corresponds to a specific DL constructor. The corpus also includes multiple OWL/OWL 2 constructors that correspond to the same DL

constructor with a specific DL expressivity. We list OWL/OWL 2 constructors found in the corpus with the corresponding DL syntax (Table 4.2). In our work, some counts (n) were taken on individual constructors and some on OWL 2 constructor groups, each of which corresponds to a specific DL constructor. For example, the three OWL 2 constructors for qualified cardinality restriction that are given in Section 4.4 correspond to a single DL constructor of qualified cardinality restriction (Q). Again, both RDF/RDFS constructor *rdfs:Class* and OWL/OWL 2 constructor *owl:Class* refer to the same DL constructor of Atomic Concept (A).

Based on the score and frequency of OWL 2 constructors or constructor groups, we found that there are three identifiable categories of constructors: 1) *commonly used constructors*, 2) *infrequently used constructors*, and 3) *unused constructors*. In the following subsections, we discuss each of these categories of constructors in more detail.

Commonly used OWL/OWL 2 Constructors

Our analysis shows that some OWL/OWL 2 constructors have been commonly used in a majority of ontologies. The OWL 2 constructors or constructor groups that have a score of greater than or equal (≥) to 8 and a frequency of greater than (>) 5% were included in this category (Table 4.2). The constructors *Class()* and *SubClassOf(C, D)* have the highest score of 14 and frequencies of 12.27% and 12.41%, respectively. The constructor *SubPropertyOf(P_1 P_2)* has the smallest score of 8 within this category and a frequency of 5.6%.

Infrequently used OWL/ OWL 2 Constructors

In our analysis, we have also identified a set of infrequently used OWL/OWL 2 constructors. These OWL/OWL 2 constructors or constructor groups have a score of less than (<) 8 and a frequency of less than (<) 5% (Table 4.2). They also have a low sum that is below 100. These measures show that infrequently used OWL/OWL 2 constructors are not required much within the corpus. Furthermore, all these OWL/OWL 2 constructors have a frequency below 2% and some of them have frequencies even below 1% (Table 4.2).

Unused OWL/OWL 2 Constructors

OWL 2 provides many constructors to specify concepts, object properties, and data type properties. However, we have found that some OWL/OWL 2 constructors are not used within the corpus; for example, object and data complement constructors: *ObjectComplementOf()*

and *DataComplementOf()*. None of the ontologies in the corpus specified individual values and reflexivity. Furthermore, the ontologies in the corpus did not use some of the data and object properties. More details of the unused OWL/OWL 2 constructors are provided in Section 5.2.

4.3.2 The Expressivity of Learning Domain Ontologies of the Corpus

We can measure the diversity of an ontology based on the number of different OWL/OWL 2 constructors used in a single ontology. A higher diversity means that a wide variety of OWL/OWL 2 constructors is used in that ontology. However, it is possible that some OWL/OWL 2 constructors or constructor groups are slight variations that refer to the same DL constructor. In such a situation, the DL expressivity of the ontology would not change. For example, all the OWL 2 constructors on qualified cardinality restrictions refer to the same DL expressivity (*Q*). Based on the different types of OWL/OWL 2 constructors used in each ontology of the corpus, we identified the DL expressivity of these ontologies (Table 4.3).

Table 4.3: An analysis of the expressivity of the corpus

#	**Ontology File Name**	**Diversity**	***FL*$^-$**	***R+***	***H***	***I***	***N/Q***	***O***	***D***	**Expressivity**
1	TMDU.owl	11	√						√	*FL$^-$(D)*
2	TITech.owl	12	√						√	*FL$^-$(D)*
3	HU.owl	11	√						√	*FL$^-$(D)*
4	AcademicInstitute.rdfs	9	√		√				√	*FL$^-$H(D)*
5	swrc_v0.3.owl	31	√			√			√	*FL$^-$I(D)*
6	AIISO schema-20080925.owl	21	√		√	√			√	*FL$^-$HI(D)*
7	univ-bench.owl	21	√	√	√	√			√	*FL$^-$R$^+$HI(D)*
8	MQ_Ontology.owl	27	√	√	√	√	*Q*		√	*FL$^-$R$^+$HIQ(D)*
9	CSU_Ontology.owl	23	√	√	√	√	*Q*		√	*FL$^-$R$^+$HIQ(D)*
10	HERO_ONTOLOGY_V 25.06.2013.owl	13	√			√	*Q*	√	√	*FL$^-$OIQ(D)*
11	ecs.owl	21	√		√		*N*	√	√	*FL$^-$HON(D)*
12	instOntology.owl	22	√			√	*N*	√	√	*FL$^-$OIN(D)*
13	lom.owl	32	√			√	*N*	√	√	*FL$^-$OIN(D)*
14	university.owl	32	√	√		√	*N*	√	√	*FL$^-$R$^+$OIN(D)*

The diversity of the ontologies in the corpus varies from nine to 32, whereas their expressivity varies from *FL$^-$(D)* to *FL$^-$R$^+$ION(D)*. Still, we further analysed the usage of

OWL/OWL 2 constructors used in the corpus to reveal a broader view of the OWL/OWL 2 constructors contained within it.

All the learning ontologies of the corpus include OWL/OWL 2 constructors that refer to the Frame Language (FL^-). FL^- includes intersection of concepts, value restrictions and simple existential quantification (Baader et al. 2003). FL^- is a sublanguage of the Attributive Language (*AL*) that is obtained by disallowing atomic negation from *AL* (Baader et al. 2003). *AL* includes the features of atomic concept, universal concept, bottom concept, intersection, value restriction, limited existential quantification and atomic negation (Baader and Nutt 2003). If full negation/complement (*C*) is included, then we derive the DL language *ALC*.

We have found that transitive roles (R^+) were included in four ontologies of the corpus (Table 4.3), which means that their expressivity corresponds to the DL language S ($ALCR^+$), provided that complement is included. Also, six of the learning ontologies of the corpus include role hierarchy (*H*). Inverse properties (*I*) are used in nine ontologies. Number restriction (*N*) or qualified number restriction (*Q*) is included in four ontologies. Constructors that refer to nominals (*O*) were found in five ontologies. Finally, all the ontologies of the corpus include data types (*D*). These language features result in higher expressivity. In short, our analysis shows that the corpus includes ontologies with a range of expressivities. Also, it is worth studying the expressivity of the corpus in comparison to the OWL 2 standard profiles.

4.4 A Comparison of Constructors in the Corpus and OWL 2 RL Profile

The OWL 2 RL profile includes a subset of OWL 2 constructors recommended by the W3C. In this section, we compare the constructors of the OWL 2 RL profile with those of the corpus. We check which OWL 2 RL constructors are used in the corpus and which are not.

4.4.1 OWL 2 RL Constructors Used in the Corpus

The corpus includes OWL 2 RL constructors that belong to different categories. However, in some situations, the corpus includes ‘old’ OWL constructors as well. These old OWL constructors are the predecessors of the OWL 2 RL constructors and each of these old OWL constructors and its corresponding OWL 2 constructor refer to the same DL constructor. Hence, in cases where an old OWL constructor was found, it was interpreted as an OWL 2 constructor.

Predefined Class Expressions

All predefined class expressions of OWL 2 RL, except the empty class *owl:Nothing*, were found in the corpus. The universal class (or top concept in DLs) *owl:Thing* is used as the superclass of all the other classes. The universal class *owl:Thing* includes all the individuals of an ontology. For example, in the *instOntology.owl* ontology, the class expression in (9) specifies that the class *Research_Interest* is a subclass of the universal class *owl:Thing*.

SubClassOf(:Research_Interest owl:Thing) (9)

Therefore, according to (6), instances (individuals) of the class *Research_Interest* become individuals of the universal class *owl:Thing* as well. The concepts in a domain are defined as named classes. In the above example, *Research_Interest* is a named class that is specific to the *instOntology.owl* learning ontology.

Boolean Connectives and Enumeration

The Boolean connectives intersection *IntersectionOf()* and enumeration *OneOf()* were found in both the OWL 2 RL profile and in the corpus. The connective *IntersectionOf()* has been used in the *university.owl* ontology. For example, the statement in (10) specifies that every artificial intelligence (AI) department is fully defined as a computer science (CS) department that has the research area AI.

EquivalentClasses(:AI_Dept ObjectIntersectionOf
(ObjectHasValue(:hasResearchArea :AI) :CS_Department)) (10)

Enumeration *OneOf()* has only been used in the *ecs.owl* ontology of the corpus with a single value for each enumeration: *OneOf ("A27" ^^xsd:string), OneOf ("A41" ^^xsd:string)* and *OneOf ("A47" ^^xsd:string)*.

Object and Data Property Restrictions

Both the OWL 2 RL profile and the corpus include data property restrictions. The OWL data property constructors *domain()* and *range()* used in the *university.owl* ontology correspond to the OWL 2 RL data property constructors *DataPropertyDomain()* and *DataPropertyRange()*, and they can be directly substituted by them in an OWL 2 ontology. For example, the domain and range restrictions of the data property *hasTenure* can be specified as shown in (11). The data property constructors *domain()* and *range()* have also been used in the *uni_bench.owl* ontology to specify the domain and the range of the data property *emailAddress*.

Declaration(DataProperty(:hasTenure)) DataPropertyDomain(:hasTenure (11)

:TeachingFaculty) DataPropertyRange(:hasTenure xsd:boolean)

All object property restrictions of the OWL 2 RL profile, except self-restriction *ObjectHasSelf()*, were found in the corpus. Universal quantification *ObjectAllValuesFrom()* was found in seven ontologies of the corpus and existential quantification *ObjectSomeValuesFrom()* in six ontologies of the corpus (Table 4.2).

Individual value restriction *ObjectHasValue()* has been used in a few ontologies of the corpus. This restriction appears in the *university.owl* ontology twice. It is used to specify that AI is a research area of the computer science department as shown in (12) below.

ObjectPropertyAssertion(:hasResearchArea :CS_Department :AI);

ObjectHasValue(:hasResearchArea :AI) (12)

Again, it has been used to specify that a teaching faculty has no tenure as shown in (13) below.

ObjectPropertyAssertion(:hasTenure :TeachingFaculty :False) ;

ObjectHasValue(:hasTenure :False) (13)

The HERO Ontology also includes individual value restriction to specify three different situations. Firstly, it has been used to specify that a dean, who is a technical staff member, has a doctoral degree (14).

SubClassOf(:Dean :TechnicalStaff);

SubClassOf(:Dean ObjectHasValue(:HasDegree :doctorate))) (14)

Secondly, in the statement (15) below, it has been used to specify that a degree that is a deliverable is obtained by a doctorate.

SubClassOf(:Degree :Deliverable);

SubClassOf(:Degree ObjectHasValue(:ObtainedBy :doctorate)) (15)

Finally, in the statement (16) below, it has also been used to specify that a registrar who is a department staff member works with a chair.

SubClassOf(:Registrar :DepartmentStaff);

SubClassOf(:Registrar ObjectHasValue(:WorksWith :Chair)) (16)

The individual value restriction *ObjectHasValue()* was also found in the *instOntology.owl* ontology. The statement (17) below specifies that a teacher who is a person has a PhD qualification.

SubClassOf(:Teacher :Person);
SubClassOf(:Teacher ObjectHasValue(:hasQualification :PhD)) (17)

Object and Data Property Expressions

OWL 2 properties are used to state property expressions (Motik et al., 2009). A named object property expression can be used to connect the individuals of a domain. For example, the statement (18) below from the *university.owl* ontology specifies that an AI student has a professor in HCI or AI as an advisor.

Declaration(ObjectProperty(:hasAdvisor));
SubClassOf(:AIStudent ObjectSomeValuesFrom(:hasAdvisor :ProfessorInHCIorAI)) (18)

The named object property expression *hasAdvisor* is used in (19) to specify that *John* has *Peter* as his advisor.

ObjectPropertyAssertion(:hasAdvisor :John :Peter) (19)

Similarly, a named data property expression can be used to connect an individual with a literal. For example, the statement (20) below from the *university.owl* ontology features a named data property expression *hasTenure* which is used to specify that Peter has tenure.

DataPropertyAssertion(hasTenure :Peter :True) (20)

Class Expressions

A number of class expression constructors that are included in the OWL 2 RL profile were found in the corpus as well. As shown in (21) below, the class expression constructor *SubClassOf()* is used in the *university.owl* ontology to specify that computer science students are a subclass of students.

SubClassOf(:CS_Student :Student) (21)

The equivalent class expression constructor *EquivalentClasses()* as shown in (22) has been used in the *university.owl* ontology to specify that the class *AI_Dept* is equivalent to the class *CS_Department* which has AI as a research area.

EquivalentClasses(:AI_Dept ObjectIntersectionOf
(ObjectHasValue(:hasResearchArea :AI) :CS_Department)) (22)

In the *university.owl* ontology, it has also been stated that the class *AssistantProfessor* and the class *Professor* are disjoint as shown in (23).

DisjointClasses(:AssistantProfessor :Professor) (23)

Object Properties

In order to specify object properties, OWL 2 RL includes the subobject property constructor *SubObjectPropertyOf()*, the object property domain constructor *ObjectPropertyDomain()* and the object property range constructor *ObjectPropertyRange()*. Similarly, the corpus includes the uses of the following OWL property constructors: *subPropertyOf (), PropertyDomain()* and *PropertyRange()* to specify object and data properties. For example, in the *uni_bench.owl* ontology, it is stated that the object property *doctoralDegreeFrom* is a subobject property of *degreeFrom*, as shown in (24).

SubObjectPropertyOf(:doctoralDegreeFrom :degreeFrom) (24)

Additional object property constructors were found in the corpus (Table 4.2). The OWL functional property constructor *FunctionalProperty()* that is similar to the OWL 2 RL functional object property constructor *FunctionalObjectProperty()* was found in both the *ecs.owl* ontology and the *HERO_ONTOLOGY_V 25.06.2013.owl* ontology. The statement in (25) specifies that each individual teacher is hired by at most one faculty.

FunctionalObjectProperty(:IsHiredBy));
ObjectPropertyDomain(:IsHiredBy :Teacher); (25)
ObjectPropertyRange(:IsHiredBy :Faculty)

The OWL inverse functional property constructor *InverseFunctionalProperty()* that is similar to the OWL 2 RL inverse object functional property constructor *InverseFunctionalObjectProperty()* was also found in the *HERO_ONTOLOGY_V 25.06.2013.owl* ontology. For example, (26) states that the inverse of the property *Teaches* is functional in this ontology.

InverseFunctionalObjectProperty(:Teaches) (26)

The OWL transitive property constructor was found in the *uni-bench.owl*. The transitive property constructor *TransitiveProperty()* is similar to the OWL 2 RL transitive object property constructor *TransitiveObjectProperty()*. Using the *TransitiveObjectProperty()* constructor, the property *subOrganizationOf* is defined to be transitive (27).

TransitiveObjectProperty(:subOrganizationOf()) (27)

Similarly, the *university.owl* ontology also includes the *TransitiveObjectProperty()* constructor as shown in (28). It is used to specify that a university A is affiliated with another university C whenever A is affiliated with a university B and B is affiliated with C.

TransitiveObjectProperty(:affiliatedWith()) (28)

Assertions

Assertions provide facts about individuals (Smith, McGuinness, Volz, & Welty, 2004). Both the OWL 2 RL profile and the corpus include the uses of some constructors to make assertions about individuals or instances of OWL classes. The class assertion constructor *ClassAssertion(C a)* has been used in the corpus to specify the individuals of each class. For example, in the *MQ_Ontology.owl* ontology, the class assertion shown in (29) below states that ISYS114 is a unit.

ClassAssertion(:Unit :ISYS114) (29)

The OWL 2 constructor *ObjectPropertyAssertion(PN* a_1 a_2*)* is used to create object property assertions. For example, in the *Macquarie.owl* ontology, the object property assertion shown in (30) states that the unit ISYS114 has particular lecture slides for Week 1.

ObjectPropertyAssertion(:hasLectureSlides :ISYS114
:ISYS114LectureSlidesWk1) (30)

Again, the data properties are asserted using the constructor *DataPropertyAssertion(R a v)*. For example, the following assertion (31) states that the unit ISYS114 is worth 3 credit points.

DataPropertyAssertion(:creditPoints :ISYS114 "3"^^xsd:integer) (31)

4.4.2 OWL 2 RL Constructors Not Used in the Corpus

Some of the constructors of OWL 2 RL were not found in the corpus. In the following, we discuss them in more detail and their potential use in learning ontologies.

Class Expressions

The constructor for pairwise disjoint classes *DisjointClasses()* is included in the OWL 2 RL profile, but it was not found in the corpus. Also, even though many predefined class expressions of OWL 2 RL were found in the corpus, the empty class constructor (or the bottom concept in DLs) *owl:Nothing* was not found. The empty class constructor *owl:Nothing* is used to define terminal classes in a class hierarchy.

Boolean Connectives and Enumeration

The OWL 2 RL profile includes Boolean connectives for union: *ObjectUnionOf()* and *DataUnionOf()* and for complement: *ObjectComplementOf()* and *DataComplementOf()*. However, the corpus does not include these connectives or variants thereof: *UnionOf()* and *ComplementOf()*. In spite of that, the W3C has proposed three situations where union can be used (Motik et al., 2012).

1. Union of data ranges can be used to create a new data range by combining two or more data types. For example, *xsd:string* and *xsd:integer* can be joined as shown in (32) below to create a new data range with both *xsd:string* and *xsd:integer*.

   DataUnionOf(xsd:string xsd:integer) (32)

2. A union of class expressions can be used to form a new class that contains all the individuals that are instances of at least one of those class expressions (Motik et al., 2012). For example, the class expression in (33) could be used to create a new class that consists of all the individuals that are instances of either an assignment or a quiz.

   ObjectUnionOf(:Assignment :Quiz) (33)

3. A disjoint union of class expressions states that a class is the pairwise disjoint union of one or many class expressions (Motik et al., 2012). For example, the assertion shown below in (34) states that each assessment is either an assignment or an exam. Also, each assignment is an assessment, each exam is an assessment, and nothing can be both an assignment and an exam.

DisjointUnion(:Assessment :Assignment :Exam) (34)

The W3C shows that complement can be used in two different ways: to specify the complement of class expressions using the constructor *ObjectComplementOf()* and to specify the complement of data ranges using the constructor *DataComplementOf()* (Motik et al., 2012). A complement of class expressions consists of all individuals that are not instances of that class expression. For example, the complement class expression in (35) specifies all those things that are not instances of the class *Assignment*.

ObjectComplementOf(:Assignment) (35)

A complement of a data range can be specified using the constructor *DataComplementOf()* and consists of all the tuples of literals that are not contained in the given data range (Motik et al., 2012). For example, the statement in (36) describes literals that are not positive integers.

DataComplementOf(xsd:positiveInteger) (36)

Object and Data Property Restrictions

The self-restriction *ObjectHasSelf()* is included in OWL 2 RL, but it was not found in the corpus. A self-restriction includes an object property expression (OPE). In addition, self-restriction includes all those individuals that are connected via an OPE to themselves (Motik et al., 2012). For example, the statements in (37) below specify that Mary loves herself.

ObjectHasSelf(:loves) ; ObjectPropertyAssertion(:loves :Mary :Mary) (37)

Even though OWL 2 RL includes all the OWL 2 RL data property restrictions, the corpus did not include any of them. OWL 2 RL includes data property restrictions that are similar to the object property restrictions except that there is no data property constructor for specifying reflexivity.

Object Properties

The OWL 2 RL profile allows for property chain inclusion; however, we did not find any examples in the corpus. For instance, in the statement shown in (38), the property chain inclusion constructor *ObjectPropertyChain(*OPE_1 *…* OPE_n*)* can be used to specify that the extension of one object property expression is included in the extension of another property expression (Motik et al., 2012).

SubPropertyOf(ObjectPropertyChain(:locatedIn :partOf) :locatedIn) (38)

The equivalent object property constructor *EquivalentObjectProperties(OPE$_1$ ... OPE$_n$)* can be used to specify that all of the OPEs from 1 to n are semantically equivalent to each other (Motik et al., 2012). Therefore, in specifying domain information in an ontology, one OPE can be replaced with another OPE. For example, in the learning domain, this constructor could be used to state that the properties *hasTeacher* and *hasLecturer* are equivalent properties as shown in (39).

EquivalentObjectProperties(:hasTeacher otherOnto:hasLecturer) (39)

Pairwise disjoint properties are also included in the OWL RL profile; however, they were not found in the corpus. The disjoint object property constructor *DisjointObjectProperties(OPE$_1$ ... OPE$_n$)* can be used to specify that all of the OPEs from 1 to n are pairwise disjoint (Motik et al., 2012). For example, in the learning domain, we could specify that the object properties *hasFinalExam* and *hasAssignment* are disjoint as shown in (40).

DisjointObjectProperties(:hasFinalExam :hasAssignment) (40)

Reflexivity is an important property in general; however, the reflexivity object property constructor *ReflexiveObjectProperty(OPE)* of OWL 2 is not included in OWL 2 RL and was not found in the corpus. This constructor says that the OPE is reflexive. Hence, each individual that is connected by OPE refers to itself (Motik et al., 2012). Similarly, the irreflexivity object property constructor *IrreflexiveObjectProperty(OPE)* says that the OPE is irreflexive, that is, no individual is connected by the OPE to itself (Motik et al., 2012). For example, in the learning domain, we could specify that the object property *prerequisiteOf()* is irreflexive as shown in (41).

IrreflexiveObjectProperty(:prerequisiteOf) (41)

The object property symmetry constructor *SymmetricObjectProperty(OPE)* states that the OPE is symmetric. That is, if x is connected to y by an OPE, then y is also connected to x by the same OPE. For example, in the learning domain, suppose that two particular subjects should be studied by a student in the same semester, then the OPE *corequisiteOf* could be used as shown in (42).

SymmetricObjectProperty(:corequisiteOf)) (42)

The object property asymmetry constructor *AsymmetricObjectProperty(OPE)* states that the OPE is asymmetric. That is, if x is connected by an OPE to y, then y cannot be connected to x by the same OPE. For example, in the learning domain if one unit is the prerequisite of another unit, then, the second unit cannot be the prerequisite of the first unit as shown in (43).

$$AsymmetricObjectProperty(:prerequisiteOf) \quad (43)$$

Assertions

Assertions used to compare individuals (equality or inequality) such as *SameIndividual(a_1 ... a_n)*, *DifferentIndividual(a_1 a_2)* and *DifferentIndividuals(a_1 ... a_n)* were not found in the corpus. The corpus did not include negative property assertions of the form *NegativeObjectPropertyAssertion(P a_1 a_2)* and *NegativeDataPropertyAssertion(R a v)* either. The reason for this could be that the corpus includes ontologies that are specific to each institution. However, the above assertions would be more useful to compare elements of different ontologies. For example, the statement in (44) implies that *John Miller* and the lecturer of *ISYS332Lecturer* are the same individual in the two different ontologies *Onto-A* and *Onto-B*.

$$SameIndividual(Onto{-}A{:}JohnMiller\ Onto{-}B{:}ISYS332Lecturer) \quad (44)$$

4.4.3 OWL 2 RL vs the Learning Domain

Based on the above comparison between the constructors of the OWL 2 RL profile and the constructors used in the corpus, we observe that the corpus has fewer constructors than the OWL 2 RL profile. In particular, the OWL 2 RL profile includes all the different OWL 2 constructors that are associated with nominals (*O*). In addition to many object property restrictions, data property restrictions and assertions can be used in an ontology based on the OWL 2 RL profile. However, the corpus did not include those OWL 2 RL constructors that relate to nominals as well as object property restrictions, data property restrictions and assertions. Also, the corpus did not include object properties such as reflexivity, irreflexivity and role disjointedness that contribute to the DL expressivity of *R*. Hence, we conclude that learning domain ontologies could be specified with a smaller set of OWL 2 constructors than those available in the OWL 2 RL profile. Such a subset of OWL 2 constructors may form an OWL 2 profile that is specific to the learning domain.

4.5 OWL 2 Learn Profile

The new OWL 2 profile that we derived from the results of our analysis is called the OWL 2 Learn profile. The OWL 2 Learn profile includes the commonly used constructors of the corpus and some others that are infrequently used. We also include some OWL 2 constructors that are not used in the corpus. The inclusion or exclusion of a constructor depends on four factors: 1) the count (n) and the frequency (f) of the constructor; 2) the relative importance of the constructors; 3) the possibility of representing a constructor in an alternative way; and, 4) the impact of the constructor on the computational complexity of the profile.

4.5.1 The Constructors Included in the OWL 2 Learn Profile

The constructors that are commonly used in the corpus have higher scores of count (n) and frequencies (f). This shows that the commonly used constructors are required in many situations of the learning domain. Many infrequently used constructors are also included in the profile. The rationale for this is that some infrequently used constructors have a higher relative importance than some other infrequently used constructors. The constructor *AllValuesFrom()* is required to specify various basic situations of the domain. For example, in the *university.owl* ontology, the constructor *AllValuesFrom()* is used to specify that all the courses taken by students in the computer science department are computer science courses (see Table 4.2).

The inclusion of *InverseObjectProperty()* constructor provides the flexibility of navigating through an ontology in either direction of the OPE. For example, the OPE *hasResource()* has the inverse property *isResourceOf()*. Therefore, we can find the learning resources of a given unit using the OPE *hasResource()* as well as the unit for which the learning resources are provided. The use of the inverse property makes query-answering on a learning ontology easier.

Qualified cardinality restrictions are also infrequently used. They are more specific than non-qualified cardinality restrictions. Qualified cardinality restrictions clearly qualify what objects or data the restrictions are imposed on. Hence, specific results can be generated in query-answering. For example, the qualified cardinality constructor *MaxCardinality()* of OWL has been used in the *university.owl* ontology. For example, the statement (45) specifies that a teaching faculty can take a maximum of 3 computer science courses. The OWL constructor *MaxCardinality()* corresponds to the OWL 2 RL constructor *ObjectMaxCardinality()*.

MaxCardinality (3 :takesCourse :CS_Course) (45)

The transitive object property constructor *TransitiveObjectProperty()* is infrequently used but it is included in the OWL 2 Learn profile. Transitivity cannot be naturally expressed using other constructors. For example, the OPE *subOrganizationOf()* as shown in (46) of the *uni-bench.owl* ontology defines a transitive relationship between organisations. We can specify that every department is a suborganisation of a faculty and every faculty is a suborganisation of a university in the following way.

TransitiveObjectProperty(:subOrganizationOf);

ObjectPropertyDomain(:subOrganizationOf :Organization);

ObjectPropertyRange(:subOrganizationOf :Organization); (46)

subOrganizationOf(:Department :Faculty);

subOrganizationOf(:Faculty :University)

Another relevant example would be the prerequisite relationships between units of study where enrolment at certain units may require the completion of some other units. The transitive object property *prerequisiteOf()* is used in the statements (47) below to specify that the unit *COMP115* is a prerequisite of the unit *COMP125* and the unit *COMP125* is a prerequisite of the unit *COMP225*.

TransitiveObjectProperty(:prerequisiteOf);

prerequisiteOf(:COMP115 :COMP125); (47)

prerequisiteOf(:COMP125 :COMP225)

4.5.2 Inclusion of Unused Constructors

As we discussed above, some OWL 2 constructors have not been used to express the domain knowledge in the corpus. However, some of them would seem to be required to specify specific domain knowledge. It could also be possible that some unused OWL 2 constructors may become useful to specify particular information of a future learning domain ontology. Therefore, we consider a few unused OWL/OWL2 constructors as candidates for inclusion in the OWL 2 Learn profile.

The OWL 2 constructor *ObjectComplementOf()* was not found in the corpus. However, the derived constructor *SubClassOf()* is found in the corpus and can be defined by means of

disjunction and negation. That means, the OWL 2 constructor *ObjectComplementOf()* is indirectly an element of the OWL 2 Learn profile. Also, we include some additional datatypes in the OWL 2 Learn profile which were not found in the corpus (*xsd:decimal, xsd:integer, xsd:long, xsd:int, xsd:float, xsd:double, xsd:string, xsd:Boolean, xsd:dateTime*). Inclusion of these datatypes does not increase the expressivity of the profile, but they offer more syntactic freedom and flexibility for the ontology engineers.

4.5.3 The Excluded Constructors

A number of OWL 2 constructors are excluded from the OWL 2 Learn profile. Many of them are not used in the corpus and some are infrequently used. Reflexive and irreflexive object properties, symmetric and asymmetric object properties are some of those excluded constructors. We think that those excluded constructors would rarely be required to specify an ontology in the learning domain.

Another main group of OWL 2 constructors that are excluded from the OWL 2 Learn profile are nominals. The OWL 2 constructors that refer to nominals are: *ObjectOneOf(), DataOneOf(), ObjectHasValue() and DataHasValue()*. They have been used only in three ontologies of the corpus. Even though the inclusion of nominals gives the ontology designers more syntactic freedom, it increases the computational complexity. While OWL 2 RL includes nominals, nominals have been excluded from several Semantic Web languages and are not used in some ontologies. For example, nominals have been excluded from the DL ontology proposed by Kepler et al. (2006). The DL reasoner *RacerPro* approximates nominals by atomic concepts (Haarslev et al., 2012).

With respect to the learning domain, we propose that domain information provided with the value constraint *hasValue()* can instead be specified using a primitive class. For example, in the *university.owl* ontology, the nominal *AI* could be replaced by a primitive class *ResearchArea*. Similarly, the *uni-bench.owl* and *lom.owl* ontologies also use nominals to specify domain information that can also be expressed using primitive classes.

4.5.4 The Expressivity of the OWL 2 Learn Profile

The expressivity of a DL language is determined by the DL constructors included in that language. Hence, to identify the expressivity of the OWL 2 Learn profile, we list each OWL 2 constructor or constructor group together with the corresponding DL constructor (Table 4.4).

Accordingly, the OWL 2 Learn profile consists of all the afore-mentioned *included* OWL 2 constructors and is more expressive than the DL language *ALC*. The profile includes transitive properties (*R+*) which lift the expressivity to the DL language *S (ALCR+)*, plus several other DL constructors that further increase the expressivity of the profile to *SHIQU(D)*. Since the symbol *C* for complement can be used in place of the symbols *UE* for union and existential quantification (Baader et al. 2003), the DL expressivity of OWL 2 Learn becomes *SHIQ(D)*.

We note that the OWL 2 Learn profile (*SHIQ(D)*) has a different expressivity, to OWL 2 RL (a fragment of *SROIQ(D)*). Still, it can be used to specify all the learning ontologies of the corpus. The three ontologies: *university.owl*, *uni-bench.owl* and *lom.owl* that include nominals can also be specified in OWL 2 Learn with minor changes by converting the nominals to instances of primitive concepts.

Table 4.4: The Constructors of the OWL 2 Learn profile

#	OWL 2 Constructor/s	DL Constructor/s	DL Language
1	Thing, Nothing	Top - ⊤, Bottom - ⊥	*FL-*
2	Class	Atomic Concept - A	
3	ObjectIntersectionOf	Conjunction - ⊓	
4	ObjectAllValuesFrom	Universal Restriction- ∀	
5	ObjectSomeValuesFrom	Limited/Full Existential Restriction - ∃	
6	ObjectProperty	Atomic Role – R	
7	ClassAssertion, ObjectPropertyAssertion	Assertions C(a), R(b, c)	
8	ObjectComplementOf	Negation - ¬	*AL* (with atomic negation), *ALC* (with full negation)
9	TransitiveObjectProperty	Transitive Role - Tr (R)	*R+* — *S / (ALCR+)*
10	SubObjectPropertyOf SubDataPropertyOf	Role Hierarchy - H	*H*
11	InverseObjectProperties	Inverse Role - I	*I*
12	{Object/Data}Max/Min/Exact Cardinality	Qualified Cardinality Restrictions - Q	*Q*
13	DisjointClasses	Disjunction - ⊔	*U*
14	DataProperty, DataPropertyAssertion, xsd:{integer, string, …}	Data {Types, Values} (D)	*D*
Expressivity of the OWL 2 Learn Profile			***SHIQ(D)***

OWL 2 corresponds to a more expressive DL language *SROIQ(D)* (Horrocks et al., 2006). The OWL 2 RL profile supports all axioms of OWL 2 except the two constructors: disjoint

unions of classes (*DisjointUnion*) and reflexive object property axioms (*ReflexiveObjectProperty*). The OWL 2 RL has additional syntactical constrains on axioms to support implementing the rule-based reasoning systems. These restrictions are made to avoid inferring the existence of individuals that are not explicitly present in the knowledge base and to avoid nondeterministic reasoning (Motik et al., 2012). OWL DL has a very high DL expressivity of *SHOIN(D)* which has a complexity that is undecidable (Hitzler & Parsia, 2009). All the standard OWL 2 profiles (EL, QL and RL) are strict subsets of OWL DL (Motik et al., 2012).

OWL 2 RL is considered as a tractable fragment of Horn Logic of the DL language *SROIQ* without existential quantification (Krötzsch, Rudolph, & Hitzler, 2013). The Horn Logic has the limitation of its inability to represent disjunctive information. However, it has the advantage of refutation (Hustadt, 2005). *Horn-SHIQ* that has a lower complexity, P-complete and that does not include disjunction, has been introduced in (Hustadt, 2005). On the other hand, *Horn-SHIQ* is not a subset OWL DL as it includes qualified cardinality restriction that is outside OWL DL. The DL *Horn-SHIN* that is obtained by excluding quantified cardinality restriction (Q) becomes a subset of OWL DL.

In terms of DL complexity *Horn-SHIN* becomes an ideal subset of OWL DL. The DL, *SHIQ* has higher complexity than *Horn-SHIQ* and it is data complete for NP and it can jump to a much higher complexity of EXPTIME-complete for KBs with large ABoxes.

According to the constructors used in the corpus of learning ontologies OWL 2 Learn profile has the expressivity of *SHIQ(D)*. When we compare the expressivity of OWL 2 Learn profile with *Horn-SHIN*, we identify the below differences and similarities listed in Table 4.5.

If a lower complexity is to be achieved with *Horn-SHIN* we need to exclude negation, disjunction, existential quantification and qualified cardinality restriction. Exclusion of negation does not create a problem as none of the learning ontologies in the corpus include negation. However, the exclusion of disjunction, existential quantification and qualified cardinality restriction is not possible as we are unable to restrict the ontology engineers at different institutions to exclude those constructors. Hence, we stick to the OWL 2 Learn profile with the expressivity of *SHIQ(D)*.

Table 4.5: A comparison of Horn-SHIN and OWL 2 Learn profile

Horn-SHIN	**OWL 2 Learn Profile (*SHIQ(D)*)**
Differences	
A subset of OWL DL	Not a subset of OWL DL
Excludes negation	Includes negation
Excludes disjunction	Includes disjunction
Excludes existential quantification	Includes existential quantification
Includes non-qualified cardinality restriction	Includes qualified cardinality restriction
Excludes data types	Includes data types
Has the complexity of P-complete	NP-complete / EXPTIME-complete (for large ABoxes)
Similarities	
Excludes Nominals	Excludes Nominals
Subset of OWL 2	Subset of OWL 2

4.5.5 The Usage of the OWL 2 Learn Profile

To demonstrate the usage of OWL 2 Learn profile, an excerpt of the MQ ontology is given in Appendix: Table A.2. The MQ ontology is compliant with the OWL 2 Learn profile. The excerpt includes different statements from the MQ ontology that use different OWL 2 Learn constructors from Table 4.4. Firstly, the example includes the declaration of a class Person and a class *AssessmentTask* using the OWL 2 Learn constructor *Class()* as shown in (48).

Declaration(Class(:Person)) (48)

Moreover, it includes the subclasses of the class Person and *AssessmentTask*. Secondly, it includes the declaration of some object properties such as *hasPrerequsite* as shown in (49) followed by other types of properties.

Declaration(ObjectProperty(:hasPrerequisite)) (49)

Thirdly, the data property *assignmentMark* is declared as a subproperty of the property *assessmentMark*. Finally, a class assertion and an object property assertion is given.

4.6 Discussion and Conclusion

Our analysis shows that a corpus of 14 learning ontologies includes a subset of OWL 2 constructors. This subset is different from the OWL 2 constructors in the OWL 2 RL profile. Predominantly, the OWL 2 RL profile includes all the nominal constructors whereas the corpus includes only a few occurrences of nominals. Those occurrences of nominals could

also be represented using primitive classes. Since the OWL 2 constructors of the corpus form a subset of the OWL 2 RL profile, we consider this subset as a new profile and call it OWL 2 Learn profile. The OWL 2 Learn profile includes the great majority of the OWL 2 constructors that are used in the corpus and is sufficient to build all the ontologies in it with small modifications.

Our analysis includes a comparison of the OWL/OWL 2 constructors in the corpus with those of the OWL 2 RL profile. This comparison gives the ontology designers an insight into the use of OWL 2 constructors in existing learning domain ontologies. The new OWL 2 Learn profile has the expressivity of *SHIQ(D)* that is different from the expressivity of the OWL 2 RL profile that is based on Description Logic Programs (DLP). Hence, the potential ontology designers may select a DL-based reasoner that supports OWL 2 RL, for reasoning and querying an OWL 2 Learn ontology. *RacerPro* has been optimised for reasoning in *SHIQ(D)* (Bock, Haase, Ji, & Volz, 2008) and would be an ideal candidate for a reasoning engine for the OWL 2 Learn profile as they have the same expressivity. However, any reasoner that has a higher DL expressivity than *SHIQ(D)* would suite OWL 2 Learn profile. Eleven reasoners that support OWL 2 DL have been listed in (Parsia, Matentzoglu, Gonçalves, Glimm, & Steigmiller, 2017) that could possibly be used as an OWL 2 Learn reasoner. In the next chapter we elaborate on how to develop an ontology for the learning domain and how an ontology-based e-learning system works with legacy databases?

Chapter 5: An Approach for Developing a Learning Ontology

In Chapter 3 we gave an overview of developing a learning ontology mapping and a *plug and play architecture* for an adaptive e-learning system's framework. In this chapter, we elaborate on developing a learning ontology. Firstly, we analyse the approaches and current tools for ontology development. Secondly, we introduce our approach to ontology development.

Different approaches, methods, languages and tools used to develop an ontology have evolved over the last two decades. In the early work on ontology development topic maps have been used and one such work is found in (Pepper, 2007). They propose a topic map for the Dublin Core metadata standard. Some of the learning ontologies have been modelled in UML class diagrams as explained in (Cranefield, Hausteiny, & Purvis, 2001). In parallel to these pictorial representations of the ontologies, formal methods such as FOL, DLs and Horn Logic have been utilised to model the domain ontology.

We also notice a recent trend in the use of graphical software tools to model the learning ontologies. Some of the early ontology development tools including Protégé 2000, OilEd, OntoLingua, Apollo, OntoEdit, RDFedt, WebODE, and WebOnto (Youn and McLeod, 2006) had limited functionality. Over time, ontology development tools have been updated by the addition of different functionalities. That has made them versatile and enriched with different capabilities to perform different related tasks. Recent graphical ontology development tools such as Protégé has gained popularity and maturity due to the fact that it offers a range of functionality with a lot of plugins. It supports ontology development query-answering, reasoning, and ontology mapping. Here, we elaborate on approaches to ontology development and the use of a graphical tool for developing a learning ontology.

The rest of this chapter is organised as follows. In Section 5.1, we analyse the current approaches to ontology development. Then, in Section 5.2, we analyse the current tools for ontology development. We introduce our approach to developing a learning ontology in Section 5.3 using a sample ontology for the learning domain.

5.1 Approaches to Ontology Development

Approaches for ontology development have been proposed in the literature. Here we study them with the intention of identifying their notable features and adopting a methodology and a tool for developing ontologies for the learning domain.

Early attempts in ontology development have been analysed in (Jones, Bench-Capon, & Visser, 1998). This analysis includes an early ontology development approach to develop the Toronto Virtual Enterprise (TOVE) ontology (Fox & Grüninger, 1997; Grüninger & Fox, 1995). The methodology followed in (Uschold, 1996) includes four steps: *identify purpose, identify scope, formalization* and *formal evaluation*. The methodology that is proposed in KBSI IDEF5 (KBSI, 1994) includes five specific steps that includes ontology refinement and validation as an additional step: *organising and scoping, data collection, data analysis, initial ontology development* and *ontology refinement and validation*. The seven steps proposed in their methodology in (Fernández-López, Gómez-Pérez, & Juristo, 1997) includes evaluation and documentation as the last step. They are: *specification, knowledge acquisition, conceptualization, integration, implementation, evaluation* and *documentation*. The methodology to develop an ontology followed in Noy and McGuinness (2001) includes seven steps and has considered reuse as a step. Still, they have not considered refinement and validation. The steps they propose are: *determine the domain and scope of the ontology, consider reusing existing ontologies, enumerate important terms in the ontology, define the classes and the class hierarchy, define the properties of classes - slots,* define the facets of the slots, and *create instances*. In this methodology, Protégé has been used as the central tool for developing an ontology.

The Product Family Ontology Development Methodology (PFODM) has been introduced in Nanda, Simpson, Kumara, and Shooter (2006). It includes four main steps: *1. Component lexicon analysis and pruning, 2. Formal contexts composition, 3. Concept hierarchy formulation using Formal Concept Analysis (FCA), 4. Product family ontology formation and enrichment using OWL*. The methodology proposed in Li, Raskin, and Ramani, (2008) includes six steps: *1. preprocessing, 2. knowledge source, 3. knowledge source acquisition and maintenance, 4. concept tagging, 5. concept indexing, and 6. query processing.*

We consider the recent ontology development methods have focused on collaborative ontology engineering and a survey on them is reported in (Simperl & Luczak-Rösch, 2014). Collaborative ontology engineering has focused on ontology development processes, methods

and tools for an ontology engineering community which has members in diverse geographical locations, with different skills, knowledge and interests. An early work on collaborative ontology engineering is described in (Holsapple & Joshi, 2002). This methodology includes four phases and the initial ontology is developed by integrating the existing ontologies. Then, it is revised based on feedback received from the community.

The requirements for supporting collaborative ontology development and the collaborative features of Protégé have been discussed in Tudorache, Noy, Tu, and Musen (2008). They have pointed out several collaborative features of Protégé. They have identified several positive collaborative features of Protégé and identified Protégé as a collaborative tool for ontology development. The users have found that no special training is required to use the collaborative features of Protégé. It is possible to link Protégé to an issue-tracker system to enable users to track the progress of the task assignments that have been made as part of a discussion. Protégé can easily be extended with new annotation types and users do not have to add annotations to changes, but annotate only ontology components where the rationale for changes is recorded. Discussions and questions on a class become easily visible and reaching a consensus is well facilitated (Tudorache, Nyulas, Noy, & Musen, 2013).

When we consider an approach to ontology development, an ontology development tool is an indispensable part of it. So, in the next section we provide an analysis of the current tools for ontology development.

5.2 Ontology Development Tools

We found several surveys were conducted on ontology development tools over last two decades. An evaluation of the four ontology development tools: Protégé-2000, Chimaera, DAG-Edit and OilEd - has been done in (Lambrix, Habbouche, & Perez, 2003) in relation to bioinformatics. This evaluation has been carried out based on the literature and the tests done using the Gene Ontology (GO). The GO has been proposed as a dynamic and controlled vocabulary of genes and the biological functions that are shared by all the microbes that belong to the category eukaryotes (Ashburner et al., 2000). Lambrix et al. (2003) have concluded that none of the four systems is preferred. Each system has different strengths and weaknesses; and each is situation specific. It has been noted that, in comparison to the other systems, Protégé-2000 has strengths in: user interface, the extendability of the system using plug-ins, vast functionality with the plug-ins, and the ability to import and export the ontologies (Lambrix et al., 2003).

They also have noted that Chimaera has the strengths in its functionality that include merging and diagnosis, use of different formats to import and export ontologies, help functions, shortcuts offered for expert users and the ability for multiple users to work on the same ontology. However, its user interface has been identified as a main weakness. The authors also have identified some advantages of OilEd; use of a description logic-based model and the use of the FaCT reasoner for performing the reasoning tasks (ex: classification and checking the consistency of ontologies). DAG-Edit has specifically been built for GO ontologies and has the advantage of being easy to use and to learn the user interface (Lambrix et al., 2003).

Followed by the above work, another survey on 14 ontology development tools in the area of ontology-based knowledge management has been done and is explained in Youn and McLeod (2006). They have claimed that the current tools have problems with interoperation and collaboration and have suggested the development of a next generation of tools, not only to overcome those problems, but also to support more capabilities. Their evaluation included the 16 ontology development tools: Protégé 2000, OilEd, Apollo, RDFedt, OntoLingua, OntoEdit (free version), WebODE, KAON, ICOM, DOE, WebOnto, Medius VOM, LinKFactory and K-Infinity. These tools have been evaluated on their support in import and export formats, interface, consistency checking, multi-user support, web support and support on ontology merge (Youn & McLeod, 2006). The authors point out that these ontology tools have the disadvantages of not being interoperable and not covering all the ontology life cycle activities. They also highlight a lack of interoperability between these tools. Hence, it becomes difficult or impossible to integrate ontologies developed using different tools into a single library or to merge two ontologies which have been built using different tools or languages (Youn & McLeod, 2006).

The survey done in (Corcho, Fernández-López, & Gómez-Pérez, 2003) provides a review on methodologies, tools and languages that are available for building ontologies. Based on their review they propose to create a workbench for ontology development by integrating the currently available technologies. The aim of such an integrated workbench is to facilitate the different activities related to ontology development: development, exchange, evaluation, evolution and management, export ontologies to different ontology languages and to provide methodological support.

An online survey has been used to evaluate the ontology development tools in (Khondoker & Mueller, 2010). This survey has aimed at finding the tools that are mostly used by the users

and the drawbacks of using those tools. This survey showed that Protégé has the highest popularity as an ontology development tool. According to this survey 24 out of 32 participants were using Protégé. Also, a majority of the participants of this survey could learn Protégé in less than one month. Six out of 32 participants were using the three tools (two were using each tool): SWOOP, Internet Business Logic, and Top Braid Composer. The two tools: Onto track and IHMC Cmap Ontology Editor were used by a single participant each.

In the literature we observed a recent trend in using the ontology tools for collaborative ontology development. Such collaborative ontology development is expected to support developing quality ontologies in distributed settings (Rospocher, Tudorache, & Musen, 2014). Different collaborative features of WebProtégé has been discussed in (Tudorache et al., 2013). A quantitative analysis of the two Web-based modeling tools, WebProtégé and MoKi has been presented in (Rospocher et al., 2014). They have investigated the collaborative processes in ontology development in these two tools. They also provide an analysis of the user activity logs from both tools as a proof of collaborative ontology development in online settings. In another scholarly work described in (Paschke & Schäfermeier, 2013), a new design artifact called OntoMaven, based on the Maven-based development methodology, has been proposed. It also adapts the Maven-based concepts for developing Maven-based ontology and managing ontology artifacts in distributed repositories.

The above analysis of literature shows that Protégé is the most popular tool used by the ontology research community. It offers user friendly interfaces and multiple features and capabilities using plugins. Plugins have been developed to enhance the capabilities of Protégé in ways including reasoning, query-answering, and modelling. Due to the above reasons, we selected Protégé to develop the learning ontologies in our work. In the following section we elaborate on our approach to developing a Learning Ontology.

5.3 An Approach to Developing a Learning Ontology

The approach we follow includes a process, some ontology development methods and tools that makes a methodology. Here, we use Protégé as the primary tool. Protégé includes many other tools such as reasoners, query interfaces and graphical modelling tools which are available as plugins.

Developing learning ontologies and most of the other ontologies is a tedious task. Following a structured approach or a methodology would help to alleviate these problems. Each of the

approaches that we discussed above include a set of steps to develop an ontology. The process we follow is based on them and our approach consists of these steps:

1. Analysis of the learning domain
2. Modelling the learning domain
3. Developing a learning ontology
4. Formalising the learning ontology
5. Evaluating the learning ontology

We also follow and apply different methods associated with each step including: ontology modelling methods; ontology development methods; methods for formalising the ontology; and, ontology evaluation methods.

We used domain ontology models (Lee & Mizoguchi, 1998) and a description logic (DL) (Baader et al., 2003) based ontology languages as the main techniques to present the domain knowledge. Domain ontology models are graphical representations which are semi-structured, easy to understand and helpful during the elicitation of concepts and relations. DLs are used to formalise the ontology. In the next sections we use the Computing Department of Macquarie University as the sample domain of study.

5.3.1 Analysis of the Learning Domain

The learning domain analysis is typically done based on the information gathered during the analysis of institution-specific materials. In our work, we analyse the course handbook and the unit guides of learning institutions.

Based on the domain analysis we identify the domain concepts, properties and constraints on the properties. In this work, we limit our study to the computing department of an institution. The concepts are primarily related to the programs of study, their units and the people working in computing departments. We also identified the concepts related to the lessons and the different learning resources related to the lessons that assists the learning process. We searched for the concepts which are associated to assessments of each unit. Besides that, we also used the sources to identify the institution-specific business rules that helped to identify the constraints on the properties. Some of the concepts that we found in the domain are in Table 5.1.

Table 5.1: Some domain concepts in MQ

#	Concept	#	Concept
1	Course (undergraduate, postgraduate, higher degree research)	8	Lesson Delivery (lecture, tutorial, laboratory / practical)
2	Unit (core, elective, people, capstone, etc.)	9	Topic list
3	Unit description	10	Assessment tasks
4	Unit guide		- assignment
5	Unit aims		- tutorial submission
6	Learning outcomes	11	Teaching staff
7	Teaching and learning strategy	12	Learning resources (lecture slides, iLecture, notes, etc.)

The concepts that we identified have their own attributes that describe the concepts. Some of the attributes that we found in MQ are in Table 5.2.

Table 5.2: Attributes of some concepts in source documents of MQ

Concept	Attributes	Concept	Attributes
Department	Department name, faculty, location, head of department	Teaching staff	staff number, first name, last name, address, telephone number, department code
Degree Program	Program code, program name, program level, duration,	Student	Student number, first name, last name, address, telephone number, program code
Semester	Semester code, year,	Learning resource	Resource id, resource title, type, unit code, author
Unit	Unit code, unit name, credit points,	Assessment item	Assessment number, type, unit code, due date, return date, weight
Unit Offering	Unit code, semester, availability	Lesson delivery	Unit code, lesson number, week number,

We also identified the user profile attributes, their references on the learning resources (see Table 3.1). The preferences are named as preference attributes. The user preference attributes are associated to the metadata items of the learning resources. Hence, LOM items that are required to specify the learning resources are also identified (see Table 3.2).

5.3.2 Modelling the Learning Domain

We model the learning domain of Macquarie University based on our findings. We identify that some concepts in the learning domain includes inclusion relations between a parent concept and its child concepts. The inclusion relations are common in the learning domain

and some of the inclusion relations that we found in MQ are in Table 5.3. Some child concepts can have their own child classes that makes a deep concept hierarchy. For example, the concept *People* makes a deep concept hierarchy.

Table 5.3: Some inclusion relations in MQ

#	**Parent Concept**	**Child Concepts**
1	Course	undergraduate, postgraduate, higher degree research
2	Unit	core, elective, people, capstone, People, etc.
3	Assessment	Assignment, final exam, etc.
4	Learning resource	Lecture slides, iLecture, note, reading material
5	People	Staff (teaching staff and admin staff), Students (undergraduate and postgraduate)

Concepts in a domain have relations between them as well. They show the roles the concepts can play on another concept. A sample list of relationships that we found in MQ are in Table 5.4.

Table 5.4: Some relations between domain concepts in MQ

#	**Subject**	**Role**	**Object**
1	Student	registerIn	DegreeProgram
2	Unit	isPartOf	DegreeProgram
3	Unit	hasResource	LearningResource
4	Unit	hasAssessment	Assessment
5	Unit	hasTeachingStaff	TeachingStaff
6	LearningResource	hasAuthor	TeachingStaff
7	Student	attends	LessonDelivery
8	Unit	includes	LessonDelivery
9	Semester	hasOffering	UnitOffering
10	UnitOffering	isOfferingOf	Unit

5.3.3 Developing a Learning Ontology

In this step, we elaborate on the development of the learning domain ontology for Macquarie University using the ontology editor Protégé[4]. Some classes include subclasses that makes a class hierarchy. In subsections below we elaborate on each step we followed to develop a sample learning ontology specific to MQ. Developing an institution-specific ontology has to be done in several steps. These include developing the concepts and concept hierarchies of

[4] http://protege.stanford.edu/

the domain, developing the object properties, data properties, property hierarchies and property constrains (Noy & McGuinness, 2001).

Developing the concepts and concept hierarchies

In Protégé learning concepts are represented as classes. These concepts and concept hierarchies in an e-learning system can be developed by adding the primitive classes and their subclasses.

In MQ, ontology different types of learning resources can be organised as a hierarchy of classes. For example, Figure 5.1 shows that the classes *ILecture, LectureSlides* and *Notes* are subclasses of the class *LearningResource* that makes a hierarchy of classes due to the inclusion (*kind-of* or *type-of*) relations between them.

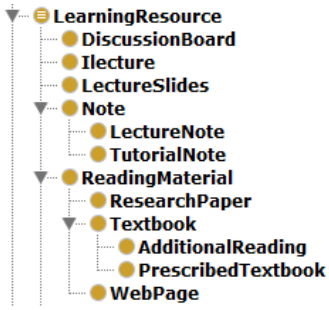


Figure 5.1: The inclusion relations between learning resources in unit guides

Again, in MQ ontology another example for inclusion relations exists between the classes *AssessmentTask* and *Assignment*; the class *Assignment* is a subclass of the *AssessmentTask* as shown in Figure 5.2.

It is essential for the concepts in a domain to interact with each other to perform a task or to provide a service to the domain users. In Protégé these interactions between the concepts are shown as object properties. An object property includes a domain and a range.

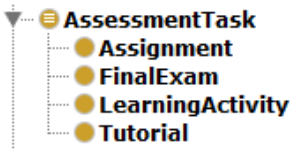


Figure 5.2: A class hierarchy of assessment tasks

Developing the object properties (Roles)

Protégé allows adding the object properties (roles) between the classes with a domain and a range. The class (subject) that initiates the role becomes the domain of the object property

and the other class (object) that the role is performed on becomes the range. For example, in the Figure 5.3 the object property *hasAssessmentMethod* has the class *Unit* as the domain and the class *AssessmentTask* as the range of it.

Figure 5.3: An object property in the MQ learning ontology

At the time of creating object properties we can create inverse properties of them as well. For example, when we create the object property *hasAssessmentMethod* we also create the inverse object property *isAssessmentMethodOf* as in Figure 5.4. The inverse properties allow navigation through the ontology in the opposite direction. For example, we can navigate from the *Assessment* to the *Unit* as *Assessment isAssessmentMethodOf Unit*.

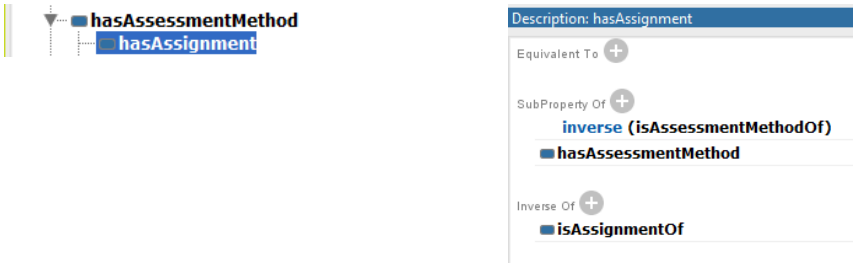


Figure 5.4: Inverse property: isAssessmentMethodOf

Developing the data properties

We create data properties that describe a class with a domain and a range. Domain of a data property is a class and range belonging to a data type. For example, in Figure 5.5 the data property *assessmentType* that describes an *AssessmentTask* is specified with its domain *AssessmentTask* and its range, *xsd:string*. Xsd:string and other data types that are used in OWL 2 have been borrowed from the XML Schema datatypes (Hitzler et al., 2012) and identified by the prefix xsd:.

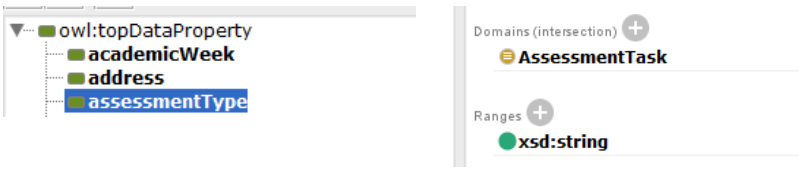


Figure 5.5: A data property in the MQ learning ontology

Creating the property hierarchies

Some properties in a learning ontology include subproperties of them. Subproperties can be created in Protégé by adding a new property under an existing property. For example, the object property *hasAssignment* is a subproperty of the object property *hasAssessmentItem*. *Unit* is the domain of *hasAssignment* and *Assignment* is the range of it. These are created in Protégé as in Figure 5.6.

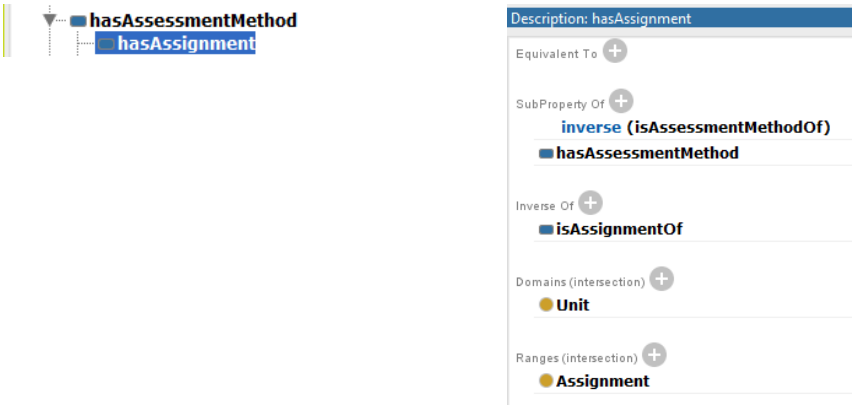


Figure 5.6: An object property in the MQ learning ontology

Adding Quantification

When the properties are added to an ontology we can quantify the domain or range of the property with the quantifiers: *some* or *all*. OWL 2 includes the constructors *ObjectSomeValuesFrom(), ObjectAllValuesFrom(), DataSomeValuesFrom()* and *DataAllValuesFrom()* for specifying the quantification in an ontology. Quantification can be done in Protégé using *class expression editor* for a selected class. For example, to specify that a unit has *some* unit aims first, we select the class *Unit.* Then, we can see the *class expression editor* under the pane *sub class of* where we can specify that *hasUnitAim(Unit some UnitAim)* as in the Figure 5.7.

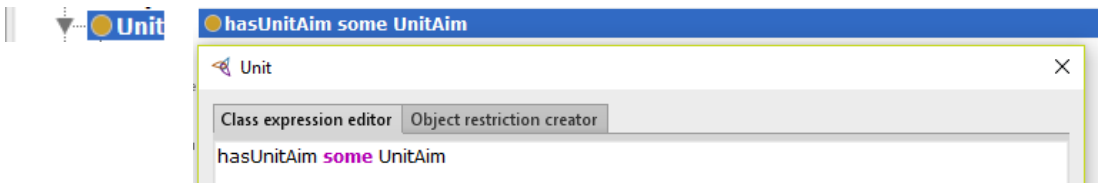


Figure 5.7: Specifying quantification on unit aim

Adding the property constrains

An institution includes different constraints or rules enforced to ensure the smooth operation of it. In terms of ontology these constraints are added on properties. Object properties can

include different constraints on them including the cardinality constraints (min, max, exactly) that specify the number of instances of a class that can participate in that *ObjectProperty*. The OWL 2 includes the cardinality constraints *maximumCardinality(), minimumCardinality(), exactCardinality()*. For example, we specify the exact cardinality constraint on the object property *isAssesmentMethodOf* as in Figure 5.8. It specifies that an *AssessmentTask* can be an assessment method of exactly one *Unit* as shown in Figure 5.8.

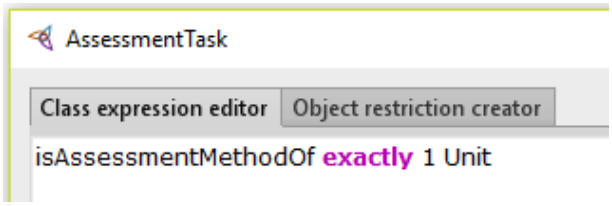


Figure 5.8: A cardinality constraint on an object property in the MQ learning ontology

Other object property characteristics include functional properties, inverse functional properties, transitive properties, symmetric properties, asymmetric properties, reflexive properties and irreflexive properties that can be specified in Protégé using property characteristics as in Figure 5.9.

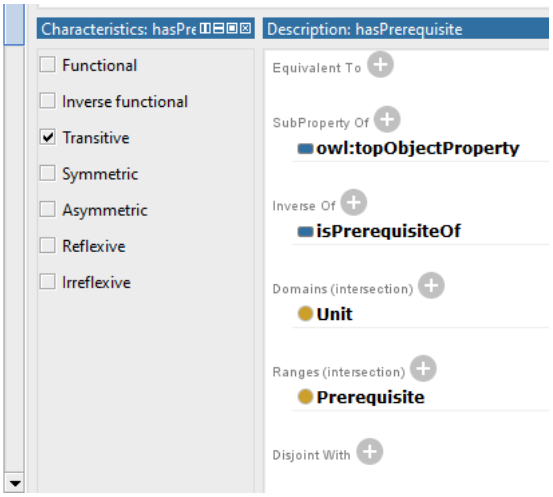


Figure 5.9: Specifying property characteristics in Protégé

However, according to OWL 2 Learn profile (see Chapter 4), we use only inverse property and transitive property in learning ontologies. Other property characteristics are not allowed in OWL 2 Learn profile.

5.3.4 Formalising the Ontology

Protégé provides a GUI for us to develop an ontology in a simple and easy to understand way. Even though an ontology developed on the GUI of Protégé is human readable it is not

machine readable. Protégé allows us to save an ontology in a specified ontology language. Ontology languages are based on formal knowledge representation methods such as DL that provides different constructs to specify the elements of an ontology. Formalising an ontology makes it both human readable and machine readable. We store the learning ontologies as OWL files.

Formalising the ontology in a DL-based ontology language has several advantages. A reasoner can perform reasoning tasks such as checking the consistency on a formal ontology easily. Moreover, formal ontologies provide the basis for automated query-answering (Donini, Lenzerini, Nardi, & Schaerf, 1996; Horrocks et al., 1999). Concepts and instances that occur in an ontology can be searched for easily and the ontology can be extended by adding more contents. Also, when an ontology is in a formal language, and when it is updated with the new contents, a DL reasoner can recheck it for its consistency automatically.

The sample ontology we use here is an OWL 2 Learn ontology. OWL 2 Learn profile offers certain formal constructs which can be used to establish the relationships between the concepts. Each OWL 2 Learn statement is either a part of the TBox of the ontology or part of the ABox of the ontology.

In the next section we elaborate on developing the TBox and ABox of a learning ontology gradually in Protégé. Protégé allows us the save an OWL ontology in different ontology formats (functional, turtle, xml, etc.). We show forming the MQ ontology using the OWL 2 functional-style syntax (Motik et al., 2012). The OWL 2 functional-style syntax provides an intuitive human-readable as well as machine-processable format to express the OWL 2 Learn constructs.

Building the TBox

Here we show how the TBox of an ontology is created in the backend of Protégé as a result of the tasks performed on its user interface in developing the MQ ontology.

Classes and Class Hierarchies—When we create classes and class hierarchies in Protégé and save as an OWL file, they are saved as OWL 2 statements. For example, if we would consider the following statement (50) found in MQ ontology, it declares that *LearningResource* is a class. The subsequent statement in (51) defines an equivalent class to the class *Unit*.

Declaration(Class(ont:LearningResource)) (50)

EquivalentClasses(ont:Unit ObjectSomeValuesFrom(ont:partOf ont:DegreeProgram)) (51)

Concept hierarchies are built using the OWL 2 Learn constructor *SubClassOf()*. For example, the following statement in (52) specifies that the class *Ilecture* is a subclass of the concept *LearningResource*:

SubClassOf(ont:Ilecture ont:LearningResource) (52)

***Object Properties*—**Once the concepts are defined, the object properties or roles of the TBox concepts are defined as in (53) using the OWL 2 constructor *ObjectProperty()*. In this example, the object property *hasPrerequisite* is defined as an inverse property of *isPrerequisiteOf* and as a transitive property. The object property *hasPrerequisite* has the class *Unit* as the domain and the class Prerequisite as the range.

Declaration(ObjectProperty(ont:hasPrerequisite));
InverseObjectProperties(ont:hasPrerequisite ont:isPrerequisiteOf);
TransitiveObjectProperty(ont:hasPrerequisite); (53)
ObjectPropertyDomain(ont:hasPrerequisite ont:Unit);
ObjectPropertyRange(ont:hasPrerequisite ont:Unit)

***Data Properties*—**In the next step we show how the data properties of concepts are formalised by declaring them as OWL statements. For example, credit points of a *Unit* are declared as a data property of the type *xsd:integer* as in (54). Here the domain of the data property *creditPoints* is Unit and the range is *xsd:integer*.

Declaration(DataProperty(ont:creditPoints));
DataPropertyDomain(ont:creditPoints ont:Unit); (54)
DataPropertyRange(ont:creditPoints xsd:integer)

***Quantification*—**Once the object properties and data properties are declared, quantifications are added to make restrictions on them. Quantification specifies whether some or all object properties participate in the relation. For example, we can specify that a unit has some unit aims as in (55).

Declaration(ObjectProperty(ont:hasUnitAim));

ObjectPropertyDomain(ont:hasUnitAim ont:Unit);

ObjectPropertyRange(ObjectSomeValuesFrom(ont:hasUnitAim ont:UnitAim)) (55)

***Cardinality Constraints*—**Formalizing the cardinality constraints can be done in OWL 2 with the constructors: *maximumCardinality(), minimumCardinality(), exactCardinality()*. In example (56) below we specify that *AssessmentItem* can be an *AssessmentItemOf* exactly one *Unit*.

(AssessmentItem exactCardinality(isAssessmentItemOf 1)) (56)

Building the ABox

Once the ABox is constructed, the instances of the concepts and the relations between the instances are added to the ontology on the interface. Then, Protégé defines them in OWL and saves in the repository.

***Adding Class Instances*—**Instances are created by giving the instance name and the associated concept name. For example, *ISYS114* is an instance of the class *CoreUnit* as given in (57) and *ISYS114IlectureWk1* is an instance of the class *Ilecture* as specified in (58):

ClassAssertion(ont:CoreUnit ont:ISYS114) (57)

ClassAssertion(ont:Ilecture ont:ISYS114IlectureWk1) (58)

***Adding Relations*—**Relations between instances are created by using their names as the domain and range of the object property that relates the instances together. For example, the unit *ISYS114* is related to *ISYS114IlectureWk1* via the role name *hasIlecture* as in (59). In that *ISYS114* becomes the domain of the object property *hasIlecture* and *ISYS114IlectureWk1* becomes the domain of it.

ObjectPropertyAssertion(ont:hasIlecture ont:ISYS114 ont:ISYS114IlectureWk1) (59)

Formalising the User Profiles and Metadata Items

The user preference attributes and learning object metadata items are organised separately and then linked with the learning ontology. In the subsections below we elaborate on how they are formally represented in OWL 2 by Protégé when they are saved in the repository.

Formalising the User Profiles

We include the user profiles as data properties in the learning ontology. In Protégé when the data property *preferredDifficultyLevel* is added as a user preference, a formal statement is inserted to the TBox of the learning ontology as in (60). This property is a subdata property of the data property *preferredAttribute*. The data property *preferredDifficultyLevel* includes domain and range restrictions. In the example in (60), all the instances of the domain of this property belong to the class *Person* and all the values of the range belong to the data type *xsd:string*:

Declaration(DataProperty(ont:preferredDifficultyLevel));
SubDataPropertyOf(ont:preferredDifficultyLevel ont:preferredAttribute);
DataPropertyDomain(ont:preferredDifficultyLevel ont:Person);
DataPropertyRange(ont:difficultyLevel xsd:string) (60)

The relations between the students and the preference attributes are described in the ABox. For example, the relation between the student *S42010013* and the preference attribute *preferredDifficultyLevel*, is declared in (61):

DataPropertyAssertion(ont:preferredDifficultyLevel ont:S42010013 "easy"^^xsd:string) (61)

Formalising Metadata Items

The metadata items of the learning objects are also specified as data properties of the learning ontology for the MQ University. The following OWL 2 Learn statement in (62) is used in the TBox to define the *difficultyLevel* as a LOM attribute of a *LearningResource*:

SubDataPropertyOf(ont:difficultyLevel ont:lomAttribute);
DataPropertyDomain(ont:difficultyLevel ont:LearningResource); (62)
DataPropertyRange(ont:difficultyLevel xsd:string)

The data property *difficultyLevel* allows for the instances that belong to the concept *LearningResource* in the domain, and for data values that belong to the data type *xsd:string* in the range. The data property *difficultyLevel* is then used in the ABox in the following way (63):

DataPropertyAssertion(ont:difficultyLevel ont: ISYS114LectureSlidesWk1 "easy"^^xsd:string) (63)

This relation connects the instance *ISYS114LectureSlidesWk1* with the value *easy* that belongs to the data type xsd:string.

However, building an ABox with all the instances of an institution is a tedious task. It demands lot of time and can lead to errors. Current e-learning systems utilize relational databases that store data related to entities, relationships and the attributes of the learning domain. If we can use these data in existing learning databases we can circumvent the burden of populating an ABox of a learning ontology with instances. Hence, in this architecture, we propose to automate the population of the ABox with instances by mapping a legacy learning database of an institution to a learning ontology. This is discussed in Chapter 6.

5.3.5 Evaluating the Learning Ontology

Even though evaluation of an ontology is listed as the last step, an ontology can be evaluated at different stages of developing an ontology. After determining the scope for the learning ontology, we used a list of sample questions that the e-learning system should be able to answer to evaluate the initial domain ontology model. These questions help to check whether the ontology includes the relevant domain knowledge, the basic concepts, the concept hierarchy and the roles among them that are required to answer those questions. Table 5.5 shows those questions and the ontology elements that are required to answer those questions.

The ontology that was developed in the previous step is evaluated as the last step. The evaluation of an ontology could be done directly to check whether it has a complete and correct TBox and an ABox. When an ontology is evaluated directly, the classes, relations and constraints on them need to be checked. Also, the consistent use of the instances needs to be checked. An ontology can also be evaluated indirectly using some queries. Queries can be used to check the capabilities of the ontology to answer the queries. In evaluating ontologies using queries, the two criteria: recall and precision have been used in (Li et al., 2008).

Table 5.5: Questions to evaluate the initial learning ontology

#	Query	An example of axioms required to answer the query
1	Show me the learning resources of ISYS114 that are on 'understanding of the requirements gathering process'.	learningResourceOf (ISYS114ILectureWk1 ISYS114) hasLearningOutcome (ISYS114ILectureWk1 understanding of the requirements gathering process)
2	What prerequisites does COMP249 have?	prerequisiteOf (COMP125 COMP249)
3	Is COMP249 a core unit of the Software Technology major?	hasCoreUnit (SW COMP249) CoreUnit(COMP249)
4	Is COMP115 a prerequisite of COMP125?	prerequisiteOf(COMP115 COMP125)
5	Show me the different types of learning resources that COMP249 has?	learningResourceOf(COMP249ILectureWk1 COMP249) ILecture(COMP249ILectureWk1) SubClassOf(ILecture LearningResource)
6	What types of assessment tasks are there in ISYS114?	hasAssignment (ISYS114 Assignment) SubClassOf(Assignment Assessment)
7	What types of teaching staff are involved in ISYS114?	hasTeachingStaff (ISYS114 DebbieRichards) TeachingStaff (DebbieRichards)

Precision = (number of retrieved relevant documents / number of retrieved documents)

Recall = (number of retrieved relevant documents / number of relevant documents)

In our work, we evaluate the learning ontologies using a set of queries on them. We elaborate on the evaluation of learning ontologies in Chapter 7 and Chapter 8.

5.4 Discussion and Conclusion

In this chapter, firstly we explored the different approaches and tools used for ontology development with the intention of identifying an approach to develop a learning ontology. Current approaches to ontology development includes benefits and problems specific to each approach. Current approaches include different processes followed, different methods and tools used. Some early approaches such as (Grüninger & Fox, 1995) did not consider evaluation of the ontology and some approaches like those explained in (Rector et al., 2004) have focused only on the development of an ontology. They did not discuss what is done before or after development of an ontology. However, some approaches such as (Uschold, 1996) and (Fernández-López et al., 1997) have included ontology evaluation as an important step. As pointed out in (Simperl & Luczak-Rösch, 2014) we noticed that the current trend in ontology development is towards collaborative ontology development. Also, different tools

have been proposed over time for ontology development and Protégé has gained popularity in the ontology community due to its extended features with plugins and user friendliness (Khondoker & Mueller, 2010).

Based on our analysis of the current approaches we follow a blended approach with five steps that include ontology evaluation and develop learning ontologies in a similar way as explained in (Rector et al., 2004). We propose to use a set of queries to evaluate an ontology from an early stage. In our work, we use the ontology development tool Protégé in different steps. We demonstrated that the approach we followed helped us to develop learning ontologies that can be plugged to an adaptive e-learning system that we suggest. In the next chapter we elaborate on population of a learning ontology with the instances from the data in a legacy database.

Chapter 6: Mapping and Populating a Learning Ontology from a Legacy Database

As we mentioned in the previous chapters manual population of an ontology is a tedious task. Besides that, the databases used in an institution include data that provides facts about the domain. Hence, in this chapter, we elaborate on our study on legacy database to learning ontology mapping with the intention of finding an appropriate approach to learning database(s) to a learning ontology. To do that, firstly we analyse the current approaches to database to ontology mapping. Secondly, we elaborate the approach we followed to map and populate a legacy database to a learning ontology.

The interest in database to ontology mapping and transformation has also grown in the last decade. Research studies such as Protégé Ontop (Calvanese et al., 2015) and KARMA (Knoblock et al., 2012) have focussed on database to ontology mapping whereas others, such as DB2OWL (Ghawi & Cullot, 2007) and RDOTE (Vavliakis, Grollios, & Mitkas, 2013) have focussed on database to ontology transformation. There is a subtle difference between the database to ontology mapping and database to ontology transformation. Database to ontology (or ontology to database) mapping assumes that both a relational database and an ontology exist and defines links between them, whereas the database to ontology transformation assumes that only a relational database exists and database to ontology transformation rules are applied to generate an ontology (Astrova & Kalja, 2008). Numerous approaches to database to ontology mapping approaches have been proposed over time and they have been implemented using different tools. We study such different approaches with the intention of adapting one approach in our work. Then, we elaborate on mapping a legacy database(s) of an institution to a learning ontology.

The learning ontology that resulted in the mapping process is to be plugged in to an adaptive e-learning system developed according to the *plug and play architecture* that is discussed in Chapter 3.

The rest of this chapter is organised as follows. In Section 6.1, we analyse the current approaches to database to ontology mapping including few database to ontology transformation methods. Section 6.2 introduces different tools used for database to ontology

mapping. Section 6.3 provides an overview of the mapping rules followed in most of the mapping tools. Then, Section 6.4 introduces the approach we followed to populate a learning ontology from legacy databases and the conclusion is given in Section 6.5.

6.1 Approaches to Mapping a Database to an Ontology

The process of making the data in a RDB available in an e-learning system for querying goes through two main stages. They are: firstly, mapping the database schema of the learning database to the learning ontology; and secondly, populating the learning ontology with the instances from the data in the learning databases. These two stages are supported by different technologies and tools. In the next section we provide an overview of the recent approaches to database to ontology mapping and accessing data in the relational databases.

6.1.1 Approaches to Database to Ontology Mapping

There are different manual and semi-automatic approaches (Handschuh & Staab, 2002; Golbeck, Grove, Parsia, Kalyanpur, & Hendler, 2002) to generate metadata from existing information such as digital libraries. This metadata is used to specify the domain that makes a simple ontology. This early approach is on database to ontology transformation rather than mapping. These approaches consider web pages and information sources as static. However, some web pages are dynamic as they are automatically updated from the associated databases. Hence, databases can be annotated rather than annotating the individual web pages (Handschuh, Staab, & Volz, 2003). Most of the early approaches have focused on annotation of the web pages or the databases. That has led to specify the mapping between the databases and the web pages.

The work done in (Handschuh, Staab, & Volz, 2003) attempts to define mapping rules between database and ontology for semantic querying. That work is called as ‘deep annotation’ as annotation is not done at the presentation layer (in HTML), but metadata is created in the data description layer. They also propose a framework for creating metadata that is created from a database with the participation of the database owner. This approach assumes that many web sites will participate in the Semantic Web and will be involved in sharing their information by providing the structure of the information in the web pages. This leads for users to use 1. information proper, 2. information structure, and 3. information context. Then the user may use these details to map the information structure of that site to an information structure of his or her own (an ontology) (Handschuh, Staab, & Volz, 2003).

The Deep Annotation Process given in (Handschuh, Staab, & Volz, 2003) consists of four main steps. Firstly, the database owner produces markups for the web pages on the server side based on the information structure of the database. Secondly, an annotator produces client-side annotations based on client ontology and server-side markups. Thirdly, the annotator publishes the client ontology and the mapping rules and finally the user queries the database using the second party ontology and mapping rules.

R_2O has been proposed as an extensible and declarative mapping language to describe mappings between relational database schemas and ontologies that are implemented in RDF(S) or OWL (Barrasa, Corcho, Shen, & Gomez-Perez, 2004). This approach maps the components in the SQL schema of the database with the ontology components in the OWL or RDFS format. The mapping definitions generated by R_2O are verified automatically by itself. R_2O also allows using the mapping definitions even to verify the components of the database according to the components of the ontology.

The approach proposed in (Xu, Zhang, & Dong, 2006) automatically constructs the mappings between a relational database and an ontology. This process is based on a set of predefined heuristic rules that considers the conceptual correspondences between the database schema and the ontology. This automatic mapping process has been implemented in a prototype tool called D2OMapper. In addition to the database to ontology mapping, D2OMapper has some secondary functions that help the user to create and maintain the mappings manually (Xu et al., 2006). In the Semantic Web, even though the web pages are annotated with ontologies, current technologies annotate only the static web pages. Hence, their approach suggests annotating the corresponding relational database of the dynamic web pages. They have proposed extending a conventional dynamic Web-based system to automatically generate semantic annotations for the dynamic contents, by integrating three main components:

1. A Web ontology to capture the knowledge of the relational database underlying the Web site;
2. A set of generic mappings between the database schema and the ontology. These mappings map the implicit semantics of the database schema to the explicit and formal ontology; and,
3. A universal algorithmic procedure—in responding to page requests the procedure is invoked by the dynamic page scripts that are executed by the Web server. This procedure uses schema-to-ontology mappings to produce ontological instances that semantically describe the database query results contained in the pages.

The tool DB2OWL proposed in Ghawi and Cullot (2007) automatically generates mappings between the ontologies and the information sources in addition to generating ontologies from database schemas. So, first it does database to ontology transformation. In this approach, conceptual elements in the database are detected before converting the database components to the matching ontology components. The mapping process followed in this approach involves 6 steps. In those steps the database tables are converted to classes and subclasses in the ontology, one to many or many to many relationships between tables are converted to object properties and columns are converted to datatype properties. This approach is implemented as a prototype named DB2OWL that creates OWL ontology from a relational database (Ghawi & Cullot, 2007; Cullot, Ghawi, & Yétongnon, 2007).

With the recent growth of medical knowledge discovery systems and decision support systems, medical researchers have to write complex queries on databases (Munir, Odeh, & McClatchey, 2009). Hence, as a supporting tool for query formulation, ontology is proposed. Still, a query generator has to map schema of the relational database to an ontology. The approach proposed in Munir et al. (2009), includes a method to map a relational database schema to an ontology that can be used in the process of formulating relational database queries. This aims at assisting end users who have no prior knowledge of complex data structures in the query formulation process. Besides that, this approach allows users to change terms, and provide access to the database with no transitional data. Generation of database queries is supported by exploiting the semantic relations and assertion capability of OWL-DL ontology.

An ontology-assisted query formulation framework has already been proposed in Munir, Odeh, and McClatchey (2008). Furthermore, several heuristic-based algorithms are proposed in the above work to translate relational queries based on ontology. They have focused on domain metadata representation for ontology query formulation, and storing and retrieving mapping between relational schema and ontology during query formulation.

A simple SQL-based RDB to RDF/OWL mapping approach is demonstrated in Bumans (2010). This approach defines the relationship between the database tables and ontology classes, again between the fields of database tables and object/data type properties of OWL ontologies. RDF triples are generated as class instances from the relational data in a database by using the automatically generated SQLs statements. This approach in Bumans (2010) allows someone to use the database engine to process the mapping information and generate

SQL statements. When the SQL statements are executed RDF triples that are instances of OWL classes and OWL object/ data type properties are created.

Several difficulties in the mapping process that need to be overcome are discussed in (Bumans, 2010).

1. In a database schema, a table can have a field that refers to another table (a foreign key), but that field may not refer to an exact object property.
2. A data type property may correspond to a combination of fields that refer to multiple tables.
3. The subclass relations in ontologies can be many-to-many relations. However, in databases many-to-many relations are removed during the database design by the normalisation process. This removes the actual semantic relations between concepts and this hinders the direct mapping.

The approach described in (Hazber et al., 2016) includes two phases: developing an ontology from a RDB schema and automatically populating the ontology with instances from the data in a relational database. As it does not use an ontology at the beginning, it becomes a database to ontology transformation approach rather than a mapping approach. The resulting ontology is in RDF(S) or OWL and the ontology instances are RDF triples. This is a fully automated approach and it has many advanced features to the current similar approach to data transformation. It includes many advanced features and covers many mapping rules in relational database to ontology transformation (Hazber et al., 2016).

6.2 Database to Ontology Mapping Tools

The approaches to database to ontology mapping have led to the development of different database to ontology mapping tools. They include: R2O, D2RQ, Virtuso RDF views and DartGrid. In R2O, mapping information is provided in an XML file whereas D2RQ does that using SQL. In both D2RQ and Virtuoso RDF view retrieves data from a RDB using SPARQL queries. A number of database to ontology mapping tools have been compared in (Sicilia, Nemirovski, & Nolle, 2017). Mappings generated by most of the recent mapping tools is based on the mapping rules of RDB to RDF Mapping Language (R2RML). R2RML is the W3C recommendation for RDB to ontology mapping and an introduction to it has been provided in (Das, Sundara, & Cyganiak, 2012).

A tool named RBA (**R**2RML **B**y **A**ssertion) that can automatically generate R2RML mappings has been demonstrated in (Neto, Vidal, Casanova, & Monteiro, 2013). RBA tool is

based on a set of semantic mappings for modelling the relationship between the database schema and a target ontology in RDF. In that work, a set of assertions that are simple to understand are used to specify the semantic mappings.

A database to ontology mapping tool named BootOx as presented in Jiménez-Ruiz et al. (2015) has been implemented in a system named OPTIQUE (Kharlamov et al., 2016). This tool also uses the R2RML as the mapping language and generates OWL 2 and OWL 2 profile ontologies. In addition to that, it also considers mapping many xsd data types and handles the xsd data types that are not in OWL 2.

Ontology-Based Data Access (OBDA) proposed in (Calvanese et al., 2015) allows creating database to ontology mappings and allows direct access to the relational data. An OBDA system consists of an ontology, a relational database and mappings between them. However, when a set of data sources is used, then it is called Ontology-Based Data Integration (OBDI) (Calvanese et al., 2017). OBDA systems use relational database schema as the source and a RDF(S) or OWL file as a destination. The database to ontology mappings are done according to the rules in R2RML (Das et al., 2012). The conceptual layer, the ontology, is connected to the database through the underlying R2RML mappings. The ontology is also able to provide an integrated view of the data and grant access to data sources. It creates declarative mappings between the data source and the ontology to connect the ontology to the data sources (Lembo, Mora, Rosati, Savo, & Thorstensen, 2015). In query-answering, instead of retrieving the triples from the ontology, the SPARQL queries are translated into SQL queries to retrieve the data from the database. Ontop has been introduced as an open source OBDA framework (Calvanese et al., 2015). Ontop includes four layers in its architecture: inputs, Ontop core, high level APIs and the applications (Figure 6.1).

Taking a step forward, a formal analysis of mapping in OBDA has been done especially paying attention to the problems of identifying the mapping inconsistency and redundancy in (Lembo et al., 2015). In their work, different ontology languages including OWL 2, OWL 2 profiles and mapping languages that have different expressiveness over relational databases, have been examined. They also have provided algorithms and established complexity bounds to solve the above problems (Lembo et al., 2015).

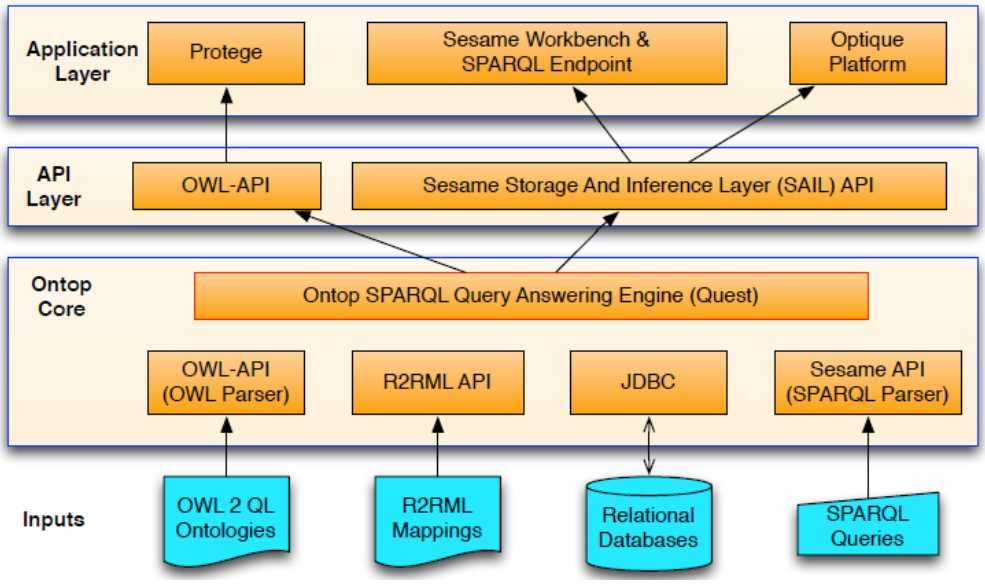


Figure 6.1: Architecture of the Ontop system (Calvanese et al., 2015)

A drawback of adopting OBDA is the greater amount of effort that is required to create manual mappings. A system named AutoMap4OBDA, that automatically generates R2RML mappings from the intensive use of relational source contents and an ontology, has been proposed in (Sicilia & Nemirovski, 2016). In that work, ontology-learning techniques are utilised to generate the class hierarchies. The mappings are generated based on the string similarity metrics, the target ontology labels and graph structures. AutoMap4OBDA system has a comparatively higher performance than most of the advanced state-of-the-art mapping generators (Sicilia & Nemirovski, 2016). Map-On is a tool proposed in Sicilia, Nemirovski, and Nolle (2017) provide a web-based editor that allows visual ontology mapping. The Map-On editor provides an interactive graph layout with a point-and-click interface that simplifies the mapping creation process. The editor is capable of generating a R2RML, IRI patterns and SQL queries document based on user inputs.

A benchmark suite named RODI for automatic mapping of relational databases to ontology has been proposed in (Pinkel et al., 2015). They have discussed the different challenges in the mapping process, the data sets for both database and ontology, the queries used in the benchmark and the results they have obtained. They have used a conference database for the evaluation, several criteria for the evaluation. The four mapping tools: BootOx, IncMap, MIRROR and Ontop have been evaluated using the RODI benchmark. The evaluation results have shown differences between their capabilities and they could handle only simpler mapping challenges. They have failed to handle more advanced challenges and need further improvements to deal with real-world problems.

OBDA systems discussed above allow access to the data in a database through an ontology without populating the ABox of an ontology. The semi-automatic approach proposed in (Knoblock et al., 2012) allows mapping the structured data into an ontology and allows populating an ontology with RDF triples and saving them. The proposed approach has been implemented in a system named Karma that provides an interface to avoid ambiguities in mapping. The resulting ontology instances in Karma can be queried using an ontology query language like SPARQL or nRQL. The modelling process followed in Karma consists of four main steps (Figure 6.2): 1. Assign Semantic Types—in this step each column of a source is mapped to a node in the ontology; 2. Construct Graph—this step involves building a graph that clearly specifies the space of mappings between the source and the ontology; 3. Refine Source Model—in this step the graph is updated and refines the model based on user input; and, 4. Generate Formal Specification—in the final step a formal specification of the source model is generated from the Steiner tree that is computed in the previous step.

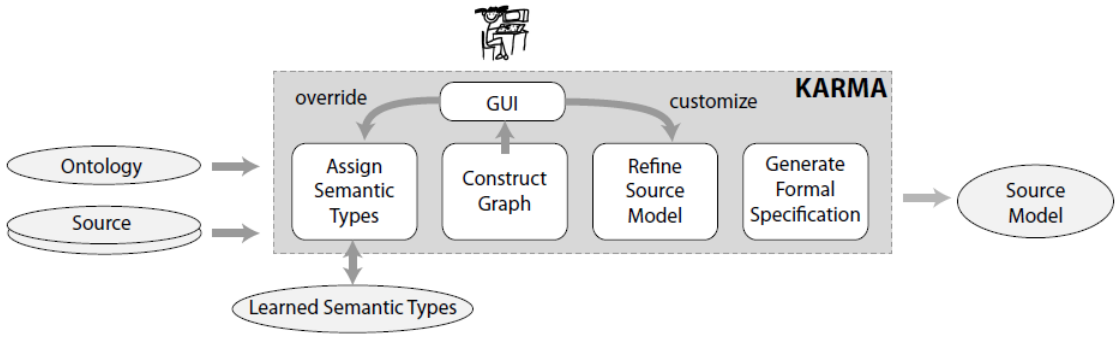


Figure 6.2: Modelling a structured source in KARMA (Knoblock et al., 2012)

6.3 Database to Ontology Mapping Rules

Database to ontology mapping is based on specific mapping rules. A number of attempts have been made to identify different mapping rules. A classification of the database to ontology mapping approaches has been provided in (Spanos, Stavrou, & Mitrou, 2012). Several rules for mapping databases to ontology have been illustrated in (Astrova, 2009). The mapping rules that are illustrated in their work are related to mapping the database tables, columns, data types, constraints and rows. Mapping rules followed in the above work can be summarised as follows.

1. *A database table is mapped* to an ontology concept if it does not have a composite primary key.

2. However, *if a database table has a composite primary key* and each part of the key is a foreign key, it indicates that the table represents a many-to-many relationship between two other tables. In such a case, that table is converted into an object property with an inverse property (Astrova, 2009).
3. When *a column of a database* table is mapped, each column is mapped to a data type property. Also, it is assigned with the maximum cardinality of one.
4. Each *data type in an SQL Schema* is mapped to an associated XML xsd: data type.
5. The *column constraint, UNIQUE* is mapped in an inverse functional property.
6. The *column constraint, NOT NULL* is mapped to a minimum cardinality of one.
7. The *PRIMAY KEY constraint* that is given as a column constraint or a table constraint is mapped to an inverse functioned property. The minimum cardinality of that property is set to one.
8. If a foreign key is not a part of the primary key, then it makes a one-to-many relationship and it is mapped to an object property.
9. When a foreign key is a part of the primary key it is mapped to an object property with the cardinality one.
10. When a foreign key is a primary key, it is mapped as an inheritance relation.
11. When the column constraint REFERENCES specifies a column in the same database table it shows a unary relation. Then, it is mapped to a symmetric property with the same class as the domain and the range of the property.
12. If the column constraint REFERENCES is accompanied by the trigger ON DELETE CASCADE then the foreign key is mapped to a transitive property.
13. The column constraint CHECK is mapped to a value restriction.
14. However, a column constraint CHECK with enumeration is mapped to an enumerated data type.
15. Each raw in a database table is mapped to an instance of a concept in an ontology.

Nine different database to ontology mapping rules, that are a subset of the above rules, have been introduced in (Mogotlane & Fonou-Dombeu, 2016). These mapping rules also include rules on mapping ontology concepts, object properties, data properties, inverse properties,

and cardinality. The mapping rules have been standardised by the W3C and they are elaborated in (Arenas, Bertails, Prud, & Sequeda, 2012). These mapping rules have been applied in different applications and in the automatic mapping tools.

When we consider the application of these mapping rules to our work, some of the rules are not applicable, as OWL 2 Learn profile excludes some of the OWL 2 constructors. For example, in our work, we do not apply the afore-mentioned rule 13 and rule 14 that refer to value restriction and enumerated data type, but we applied the rest of the rules. However, in our work, these mappings are handled by the mapping tool that we utilise and we do not have to apply them manually.

6.4 Protégé Ontop Vs Karma

As there are several mature tools for database to ontology mapping, in our work, we adopt an existing tool. Based on the literature and our experience we find that Protégé plugin Ontop and Karma as two options for our work. However, we see that each of them has their own pros and cons and the applicability of each of them becomes situation specific.

The tool Ontop provides an ontology engineer to write the mappings between the elements of the ontology and the database and save them as an obda file. It also provides the opportunity to update the mapping file. As the reasoner should be started before any OBDA happens, the reasoner is able to check the consistency of the mappings to the ontology and raise any issues for the ontology engineer to fix. An obda file created stores only the mappings and it is used to access a database to satisfy the SPARQL queries posed on the associated ontology. Ontop does not populate the associated ontology nor store the associated instances in the ontology. Hence, it avoids the need of maintaining an ontology ABox and updating it.

On the other hand, Karma provides a GUI for mapping a database schema to an ontology and updating the mappings. Hence, it avoids the need of writing the mappings manually and creates a mapping file in the background. Once, mappings are completed, it transforms the data in the legacy database into the associated ontology instances to populate the ontology. However, if the legacy database experiences regular updates they are not synchronised with the ontology in Karma. This would create the problem of the learning ontology not providing the up-to-date results.

Due to the ability of saving the populated ontology in Karma for the purpose of demonstration, in our work, we use Karma to map and populate the learning ontologies.

According to the *plug and play architecture* that we proposed in Chapter 3, the populated learning ontology is plugged into the e-learning system that is used later in query-answering.

6.5 Mapping a Legacy Learning Database to a Learning Ontology

Similar to the systems used in the other domains most of the data in the e-learning systems are available in legacy databases. These data include the details of the students, courses, units, learning resources, and assessments. In addition to that, e-learning systems store the learning resources in a repository. The *plug and play architecture* for an adaptive e-learning system's framework that we discussed in Chapter 3 includes three layers: the user interface layer, the components layer and the repository layer. Data and the ontology of the e-learning system belong to the repository layer of the architecture. Before we use the data in a legacy database in an ontology-based e-learning system, we need to map the learning database and its contents to a learning ontology. A simplified view of what is required to be done in database to ontology mapping is depicted in Figure 6.3.

Before any mappings are done we should have developed the TBox of an ontology and selected a learning database(s). We also have to deeply analyse the elements in both the database schema and the ontology TBox to get a crisp understanding of the elements in both the database schema and the learning ontology. Once we have a clear idea of the elements of both the database and the ontology we start mapping each element in the ontology with the corresponding elements in the schema of the legacy database. Once the mapping process is completed, the mapping definitions are stored in the system and used to populate the learning ontology with the data from the legacy database.

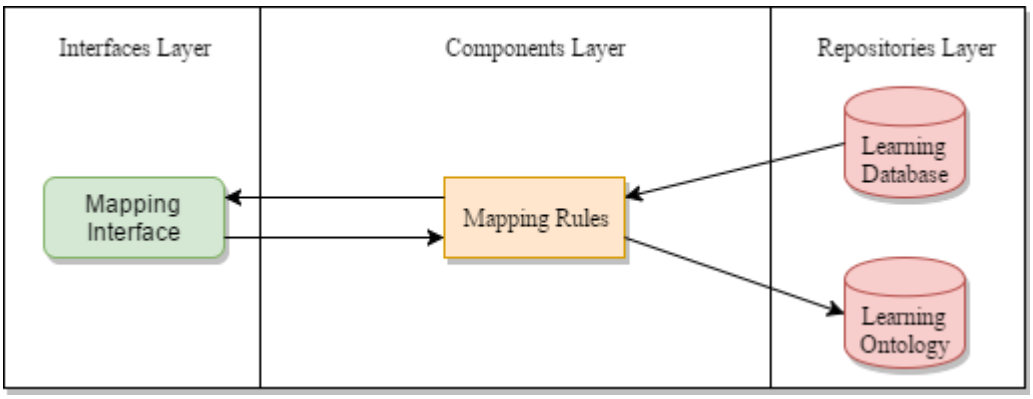
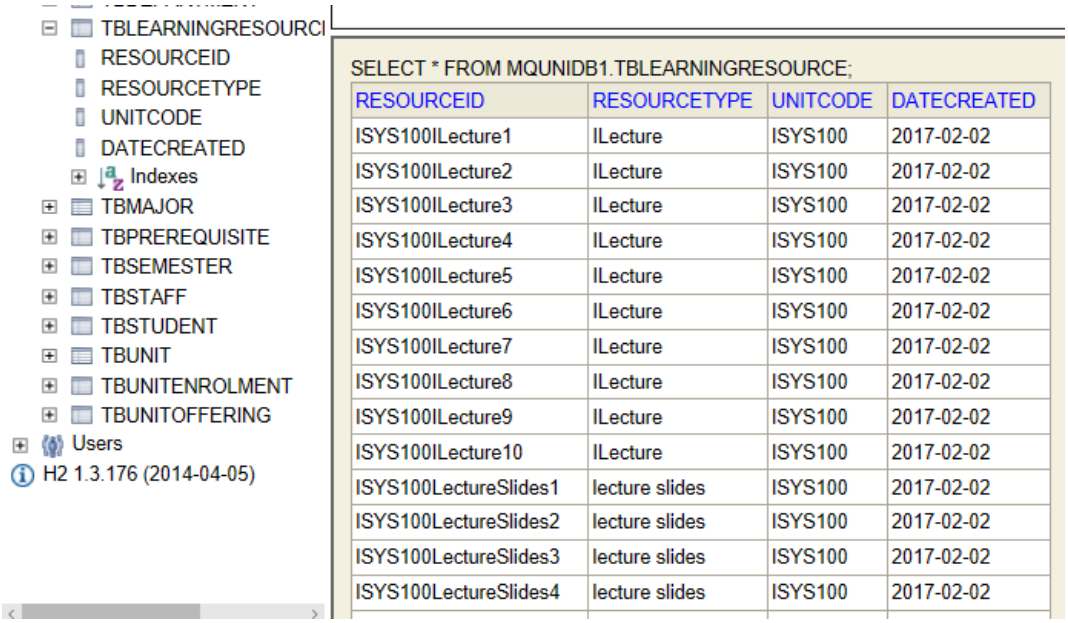


Figure 6.3: A simple representation of mapping a database to a learning ontology

6.5.1 A Sample Relational Database Schema for MQ

We start mapping a sample legacy database into a learning ontology with the database schema and ontology. In this section, we introduce a sample database schema for MQ University (Figure 6.4).

This figure of the sample database schema includes the relational tables, the relationships between them, attributes of the tables and the data. For examples, there exists a relationship between the table *tbStudent* and the table *tbCourse*. This relationship has been created by introducing the key attribute, *courseCode*, the primary key of the *tbCourse* into the table *tbStudent* as a foreign key. Each table includes several key and non-key attributes. For example, the table *tbStudent* includes studentid as the key attribute and student type, *firstName, lastName, adddress*, etc. as the non-key attributes. Each table in this database includes a collection of data and the screen shot below shows a sample set of data in the *tbLearningResource* (Figure 6.5).



RESOURCEID	RESOURCETYPE	UNITCODE	DATECREATED
ISYS100ILecture1	ILecture	ISYS100	2017-02-02
ISYS100ILecture2	ILecture	ISYS100	2017-02-02
ISYS100ILecture3	ILecture	ISYS100	2017-02-02
ISYS100ILecture4	ILecture	ISYS100	2017-02-02
ISYS100ILecture5	ILecture	ISYS100	2017-02-02
ISYS100ILecture6	ILecture	ISYS100	2017-02-02
ISYS100ILecture7	ILecture	ISYS100	2017-02-02
ISYS100ILecture8	ILecture	ISYS100	2017-02-02
ISYS100ILecture9	ILecture	ISYS100	2017-02-02
ISYS100ILecture10	ILecture	ISYS100	2017-02-02
ISYS100LectureSlides1	lecture slides	ISYS100	2017-02-02
ISYS100LectureSlides2	lecture slides	ISYS100	2017-02-02
ISYS100LectureSlides3	lecture slides	ISYS100	2017-02-02
ISYS100LectureSlides4	lecture slides	ISYS100	2017-02-02

Figure 6.4: A sample database table with data

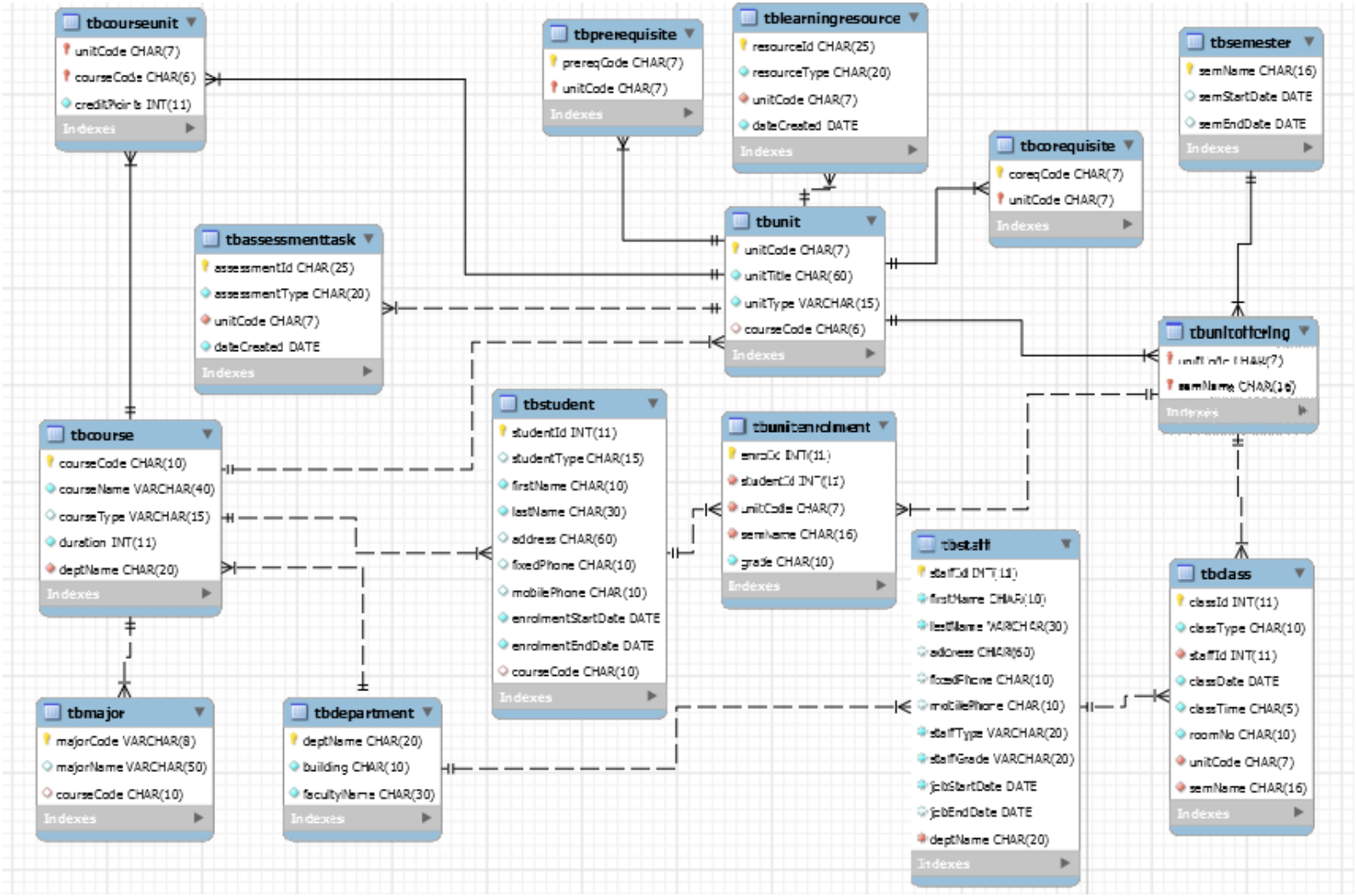


Figure 6.5: A sample MQ database schema

The tables in a database schema are related to each other by a foreign key. For example, in the sample MQ database schema, a relationship between the table *tbStudent*, has been created with the table *tbCourse*, by introducing the primary key of the *tbCourse*, *courseCode* into the table *tbStudent* as a foreign key as in Figure 6.6.

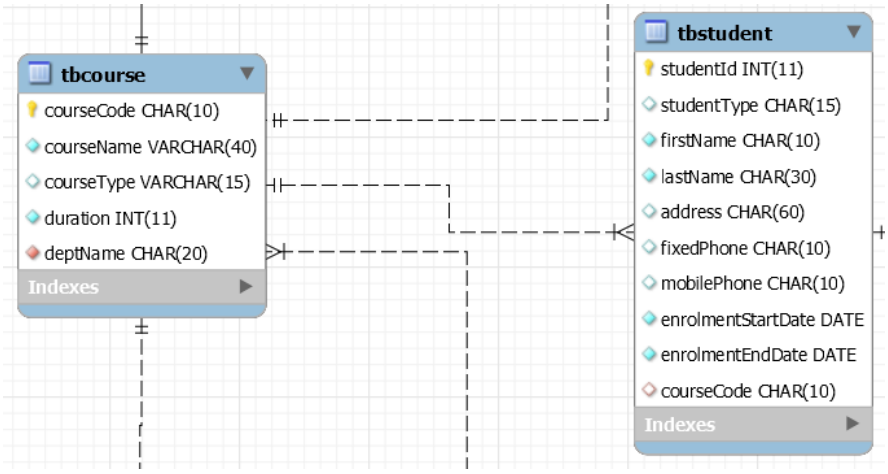


Figure 6.6: A relationship in the MQ database schema

6.5.2 MQ Ontology TBox

The second thing that we need for the database to ontology mapping is a learning ontology. Figure 6.7 below provides an overview of the MQ ontology created in Protégé. This includes sections of concepts/classes, object properties and the datatype properties in the MQ Ontology.

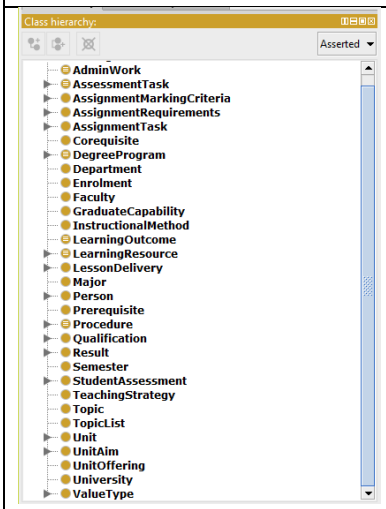
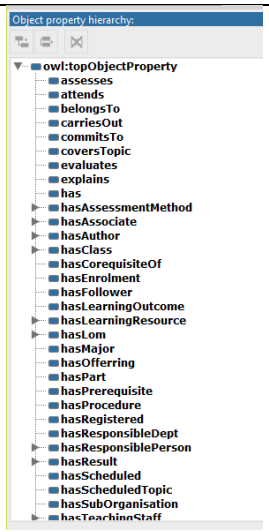
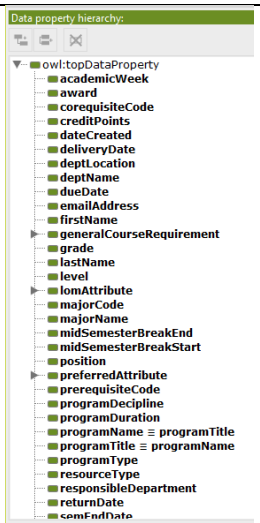
Concepts/Classes	Object Properties	Datatype Properties
		

Figure 6.7: MQ Ontology and its elements

Some classes include subclasses that make class hierarchies as well. As discussed in Chapter 4, object properties show the relationships between the classes and the data properties show the characteristics of the domain classes. Some of the object/data properties also include subproperties.

6.5.3 Mapping the MQ Database to MQ Ontology

We create the database to ontology mapping using the mapping tool, KARMA (Gupta et al., 2012). In KARMA first we open the database file and the ontology file as shown in Figure 6.8. Then, we have to select a table in the database that needs to be mapped. That makes the columns in that table visible on the interface. Then, each column of the selected database table can be mapped to a relevant data property of the matching concept in the ontology. Also, the foreign keys that represent the relationships between tables are mapped to the object properties of a class. A mapping done in KARMA includes a source or a table from the relational database and a target or a class in the ontology. For example, the Figure 6.9 shows a mapping done between the ontology concept *Learning Resource* and the database table *tbLearningResource*.

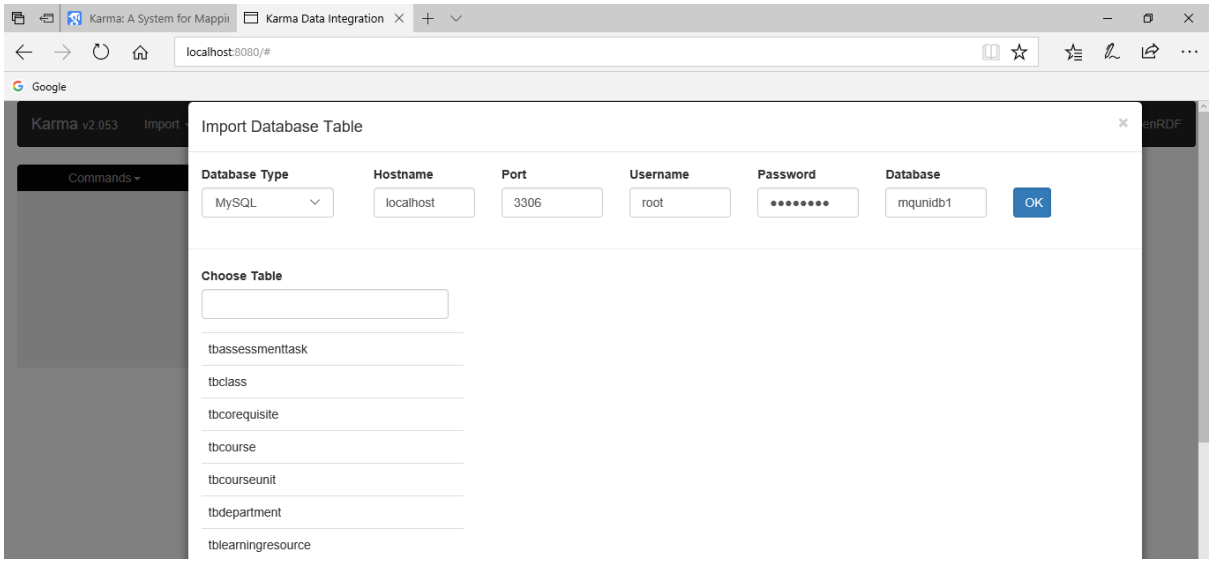


Figure 6.8: Opening the database file and ontology in KARMA

Once mapping each table in the database schema into the concepts in the ontology is completed, the ontology can be populated with the data in the database. This is done using the export option in KARMA. When the results are exported, the data in the database are saved to an RDF file with the ontology instances in the RDF triple format. The resulting learning ontology is plugged in to the ontology-based e-learning system and used for query-answering.

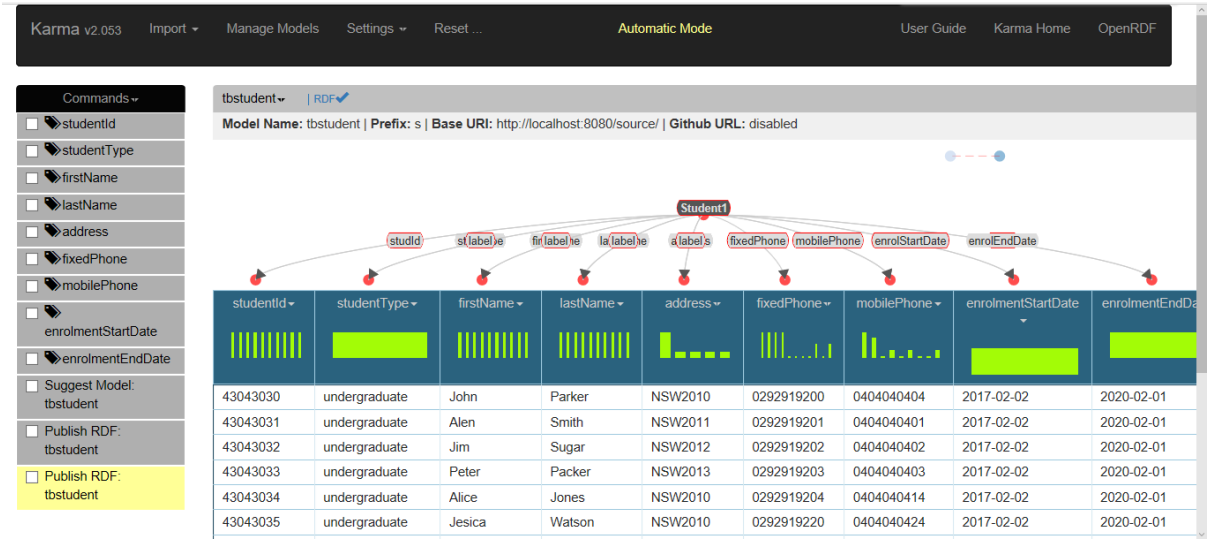


Figure 6.9: The KARMA mapping interface

6.6 Discussion and Conclusion

In this chapter, we aimed at analyzing the current approaches to database to ontology mapping with the intention of identifying an approach to map legacy databases to learning ontologies. Our analysis helped us to identify different mapping approaches ranging from manual approaches to semi-automated to the fully-automated approaches. All these mapping approaches untilise mapping rules to specify which component of a database should be mapped to which component of an ontology. We found that several tools that are based on semi-automated approaches have gained popularity. Due to the fact that there are matured current tools for mapping, we decided to use one of the existing tools for our purpose of mapping a legacy database to a learning ontology. We short-listed the current mapping tools that could be usable for our purpose to Ontop and KARMA. Ontop is one such tool that is used as a Protégé plugin and it is based on OBDA. This tool allows access to relational data through an ontology. As we intended, eventually we could get the populated learning ontology using KARMA. KARMA is a semi-automated tool that leads to obtain an ontology populated with the instances from the data in a legacy database. Hence, to serve our purpose of obtaining a learning ontology with the instances, we selected KARMA. KARMA provides a graphical user interface for mapping which makes the mappings clear to the users, yet it generates the formal mapping definitions in the background. These mapping definitions are utilized in the ontology population process.

In the next step, we search for a way to evaluate a learning ontology and check its capabilities for answering the queries in different situations of e-learning.

Chapter 7: Evaluating OWL 2 Learn Ontologies and the Framework for Ontology-based E-Learning Systems

In Chapter 4, we have proposed the OWL 2 Learn profile, an OWL 2 sublanguage for the specification of learning ontologies. We have discussed the development and population of learning ontologies in Chapters 5 and 6. In this chapter, we identify a query suite to be used in evaluation of the query-answering and inference capabilities of a learning ontology. We expect that this query suite would help us to verify the behavior of a learning ontology within the OWL 2 learn profile.

We also elaborate on the evaluation of query-answering and inference capabilities of learning ontologies; a main component of the framework. It is also assessed whether a learning ontology includes all or some of the constructors given in the OWL 2 Learn profile introduced in Chapter 4. Therefore, given a learning ontology, we need to see whether all, or at least some, of the benchmark queries of the proposed query suite can be answered.

In this chapter, in addition to evaluating learning ontologies, we also evaluate the e-learning system that is an implementation of the *plug and play architecture* (see Chapter 3). Here, we see whether it is possible for the instances of the adaptive system's prototype to be used for query-answering on multiple institution-specific ontologies. For this evaluation, we use the two sample ontologies, MQ and CSU ontologies that we used for evaluating learning ontology.

The importance of the evaluation of learning ontologies is that we can assess not only their capabilities but also their weaknesses and their problems that could be later alleviated (Brank, Grobelnik, & Mladenić, 2005). To achieve this goal, different methods have been proposed to evaluate ontologies over time. These ontology evaluation methods have been categorized in a number of ways by authors such as Brank et al. (2005) and Duque-Ramos et al. (2013). The method proposed in Duque-Ramos et al. (2013) is based on the evaluations applied in software engineering. Their approach evaluates an ontology using qualitative attributes (Duque-Ramos et al., 2013). Most of these evaluations are performed by directly inspecting the elements in the ontologies or their syntax of definitions or their qualitative attributes.

Query suites have been used to benchmark the response time and scalability of knowledge base systems (KBS). In spite of that, to the best of our knowledge such query suites have not been used to specifically evaluate the query-answering and inference capabilities of learning ontologies. There exist a number of ontology benchmarks that have been proposed to evaluate KBSs or ontology repositories. For example, Lehigh University Benchmark (LUBM) (Guo, Pan, & Heflin, 2005) and University Ontology Benchmark (UOBM) (Ma et al., 2006) are two of the popular benchmarks. These ontology benchmarks use query suites to evaluate the completeness of ontologies and response time of KBSs in query-answering using different data sets. The query suites of the current ontology benchmarks would possibly be used to evaluate the capabilities of OWL 2 Learn ontologies. Hence, in our work, we propose a query suite for evaluating query-answering and inference capabilities of learning ontologies. Here we analyse a few relevant query suites of popular ontology benchmarks. By using a query suite we expect to understand: what types of queries can be answered by an OWL 2 Learn ontology, and what specific inferences are possible on an ontology. Specific inferences depend on the ontology language constructors used in the ontology and the reasoning capabilities of the DL reasoner used for evaluation. However, here we consider the ontology language constructors of the OWL 2 Learn profile and a reasoner with the DL expressivity of OWL 2 Learn, that is *SHIQ(D)*. The ability of an ontology to generate the expected query results in an evaluation is an indication of its query-answering and inference capabilities.

We have also found that there are different ontology evaluation frameworks and methodologies that integrate different ontology evaluation metrics and methods. We have studied the current ontology evaluation frameworks and methodologies to see where our evaluation method fits. We have borrowed some of the features from those methodologies and frameworks.

In general, ontology evaluation is related to two main concepts: 1) Verification – checking the structure of an ontology; and, 2) Validation – checking qualitative and quantitative characteristics of an ontology (Bilgin, Dikmen, & Birgonul, 2014). Most of the current ontology evaluation techniques use verification which is based on directly inspecting the contents of an ontology.

The ontologies follow the OWA; whereas the databases follow the CWA. When instances are not found in an ontology for a specific query, the answer ‘unknown’ is provided, whereas a database provides the answer ‘no’. However, in both cases, no instances would be provided

as the answer to a query. In our evaluation, we do not investigate how OWA and CWA affect the query results.

The approach we follow is an indirect way of inspecting an ontology. Firstly, we check the query-answering capabilities of an ontology. Secondly, we check the inference capabilities of that ontology based on the results obtained by executing the queries. Thirdly, we analyse the structural aspects of learning ontologies that are involved in query-answering and inferences. Hence, our approach could be considered as an ontology verification method.

If a learning ontology is able to answer a specific query, this means that the given ontology has the query-answering and inference capabilities demanded by that query. It also should include the structural elements, OWL 2 constructors, instances and the facts required to answer that query. Learning ontologies are mainly developed by considering the requirements of each institution and therefore they are institution specific. Hence, different learning ontologies may have different query-answering and inference capabilities. If an ontology is not able to answer a particular benchmark query, there are two main implications. One implication is that the given ontology does not include the required instances of the concepts that are being queried. The other implication is that the ontology does not include the OWL 2 Learn constructors that are required to perform the relevant inferences. However, the first problem would not occur if an ontology was fully populated.

The benefits of this evaluation approach include identifying the problems, weaknesses and incomplete aspects of an ontology. If required, this approach provides us the choice of improving the ontology to make it capable of answering more queries. However, satisfying additional queries would not be a requirement for a specific institution unless new requirements are identified.

In this chapter, we demonstrate the evaluation of two sample ontologies we have developed for the Macquarie University and Charles Sturt University. For each learning ontology we formulated queries that are analogous to the new query suite. We have populated the learning ontologies with comparable sample test data sets that would be sufficient to test the execution of the query suite. In addition to that, in this evaluation, we use instances of a proof of concept prototype that we developed according to the adaptive e-learning system's framework.

The rest of this chapter is organised as follows. Section 7.1 provides an analysis of the approaches, frameworks, metrics used in evaluating an ontology. Section 7.2 provides an

overview of using the ontology benchmarks for ontology evaluation. An overview of the query suites used on LUBM and UOBM benchmarks and their inference capabilities are provided in Section 7.3. Section 7.4 introduces the queries of the UOBM query suite and their inference capabilities. A discussion on the possibility of using the UOBM queries to evaluate OWL 2 Learn ontologies is provided in Section 7.5. Section 7.6 introduces an enhanced query suite for evaluating the capabilities of OWL 2 learn ontologies and the methodology we follow to evaluate learning ontologies. Section 7.7 provides an overview of the evaluation process we followed and the query results are analysed in Section 7.8. Section 7.9 elaborates on evaluation of the adaptive e-learning system's framework. A discussion on the evaluation of the system's framework and learning ontologies using benchmark queries is provided in Section 7.10.

7.1 Approaches, Frameworks and Metrics for Ontology Evaluation

Ontology evaluation has already been discussed by many scholars. Here, we provide an overview on the approaches, framework and metrics found in the scholarly work.

7.1.1 Approaches to Ontology Evaluation

Ontology evaluation has been discussed in the literature by different scholars. A comprehensive survey on ontology evaluation has been carried out in (Brank et al., 2005). They have pointed out the importance of evaluating the ontologies to select the best ontology for the application requirements. These ontologies could have been developed by ontology engineers; or automatically or semi-automatically generated by ontology learning processes.

Four broad categories of ontology evaluation have been identified in (Brank et al., 2005).

1. Comparing the syntax definitions of an ontology with a "Golden Standard" - the syntax specification of a formal language or another ontology,
2. Evaluating the results of an ontology in an application,
3. Comparison of a source of data (about a domain) and an ontology, and,
4. Human evaluation—whether the ontology satisfies a set of pre-defined criteria, standards and/or requirements.

In addition to that, Brank et al. (2005) have proposed a categorization of ontology evaluation approaches based on the level of ontology evaluation. An ontology could be evaluated at different levels or as a whole. Below levels have been identified based on the level of ontology evaluation:

1. Lexical, vocabulary, or data layer—checking what concepts, instances and facts are included in an ontology;
2. Hierarchy or taxonomy—checking inclusion or *is-a* hierarchy;
3. Other semantic relations—what other relations are included except is-a;
4. Context or application level—checking whether an ontology is a part of another ontology and in what application it is used;
5. Syntactic level—checking whether the ontology satisfies the syntactic requirements of a particular formal language (this is applicable when an ontology is written in an ontology language such as OWL 2), and,
6. Structure, architecture, design—checking whether an ontology has been built according to some pre-defined ontology design principles or criteria. Whether an ontology has been built fully manually by the experts or whether it is generated automatically as an aspect of the level of evaluation.

Three categories of approaches to ontology evaluation have been identified in (Duque-Ramos et al., 2013). Considering the aim of evaluation:

1. Ranking—this is used to select the best ontology for a specific task;
2. Correctness—that is the formal correctness of the contents in an ontology; and,
3. Quality evaluation—there are different ways to evaluate the quality of an ontology using quality attributes.

The OQuaRE framework for evaluating ontologies proposed in (Duque-Ramos et al., 2013) is based on the SQuaRE standard that has been proposed for software product quality. In their work, the quality of an ontology is measured based on how far an ontology confirms to functional and non-functional requirements. The framework includes three clusters of quality characteristics. A survey has been conducted by using a questionnaire to identify the usefulness and relevance of the quality metrics. A set of experts have completed the survey in two rounds: one before reading the results of their tool and one after reading the results. However, this framework focuses only on the quality criteria that is only one category of the four categories discussed in (Brank et al., 2005). Also, it is arguable how far the software engineering-based evaluation criteria are applicable to ontology evaluation.

A survey on ontology evaluation has been conducted in (Hlomani & Stacey, 2014) with the intention of finding a better way for ontology evaluation. By ontology evaluation they have considered verification – developing the ontology right; and validation – developing the right

ontology or checking the quality and the correctness of an ontology. The ontology categorization in that work is based on the findings in (Brank et al., 2005). Furthermore, they have attempted to identify some limitations in the current methods. They have identified subjectivity as a common and major limitation in ontology evaluation research. They have discussed subjectivity in three main aspects: 1. in selection of evaluation criteria; 2. thresholds for the criteria; and, 3. on the results of ontology evaluation (Hlomani & Stacey, 2014).

7.1.2 Frameworks for Ontology Evaluation

An early attempt has been made to integrate different ontology evaluation methods into a single framework in (Gangemi, Catenacci, Ciaramita, & Lehmann, 2005). They have identified three main types of measures: structural measures, functional measures and usability profiling measures. Structural measures check the graph structure of an ontology. Ex: breadth, depth and consistency of nodes and edges. Functional measures are related to how the ontology and its components would be used. Ex: agreement of experts, user satisfaction, task, topic and modularity. Usability-profiling measures are related to the level of annotation of an ontology. Ex: recognition level, efficiency level and interfacing level. Later, the ontological errors and design anomalies of ontologies have been integrated into a single framework in (Fahad & Qadir, 2008). They have aimed; 1. to help building semantically correct and error free ontologies, and, 2. to enable mapping and merging of ontologies automatically and effectively. However, this framework also limits their work to the errors and anomalies. They are more related to the structural measures that are mentioned in (Gangemi et al., 2005). Again, an ontology evaluation framework has been proposed in (Pak & Zhou, 2009). It has proposed guidelines for choosing an evaluation method considering the objectives of the ontology. It also has proposed a set of dimensions for classifying ontology evaluation methods and measurement criteria for ontology evaluation for each dimension. The dimensions that have been mentioned in (Pak & Zhou, 2009) are: scope, layer, lifecycle, quality principles and methods for ontology evaluation.

The framework that has been proposed in (Knoell, Atzmueller, Rieder, & Scherer, 2017) is scalable especially for data-driven ontology evaluation. The authors of that paper also have applied the framework in the context of ontologies with large datasets. The framework that has been proposed in their work includes: a corpus as an input, a big data framework that is used in processing results in an attributed graph. The framework includes valuation methods that gets ontology and the attributed graphs as inputs and uses a big data framework and

metrics to get the evaluation results. This framework is also more related to measuring structural aspects given in (Gangemi et al., 2005).

Even though ontology evaluation is an integral part of ontology development, it is often done separately as quality assurance to identify errors and inconsistencies of the ontologies (Amith, He, Lossio-Ventura, & Tao, 2018). They have suggested to show that ontology evaluation and quality assurance need to be integrated to the ontology development life cycle.

7.1.3 Metrics used in Ontology Evaluation

A new methodology to evaluate ontologies has been proposed in (Bandeira, Bittencourt, Espinheira, & Isotani, 2016) is based on three main principles: 1. empirical evaluation that depends on Goal, Question Metrics (GQM) approach; 2. the goals are based on five roles of KR; and, 3. the evaluation of an ontology is based on the type of the ontology (top level, domain or application ontology). Three steps are followed in their methodology: 1. the type of the ontology is defined by the evaluator; 2. the evaluator repeats the GQM approach; and, 3. the quality of the ontology is calculated. Their methodology also includes a statistical model to automatically assess the quality of an ontology.

An ontology pitfall scanner named OOPS has been introduced in (Poveda-Villalón, Gómez-Pérez, & Suárez-Figueroa, 2014). They have used different types of pitfalls in ontology development to evaluate a corpus of ontologies. OOPS is an online tool that is independent from the ontology development tools. The authors of OOPS have used a catalog of 40 pitfalls and OOPS uses a pitfall catalog creation and maintenance process in the ontology evaluation process. However, they do not focus on evaluating the query-answering and inference capabilities of the ontologies.

A set of ontology evaluation metrics that is based on the semiotic theory has been identified in (Leukel & Sugumaran, 2009). Their metrics includes five categories of semiotic metrics: syntactic quality, semantic quality, pragmatic quality and social quality. In their approach, firstly they pre-check whether the metrics can be applied on a particular ontology, secondly, the metrics is applied on a selected ontology and it is evaluated, and, finally they analyse the evaluation results.

The frameworks that we discussed above evaluate an ontology mostly based on the structural aspects. The approach we follow for evaluating the query-answering and inference capabilities of learning ontologies is more related to the functional metrics (task) that are mentioned in (Gangemi et al., 2005). Again, checking the inference capabilities, and the

OWL 2 Learn constructors involved in them, is related to structural metrics of the above framework. However, we do not evaluate the usability aspects of the learning ontologies.

In our work, we think of an evaluation method that is related to the first two categories proposed in (Brank et al., 2005). That is, a method to check the ontology language constructors and to check the inference and query results of a learning ontology. In the literature, we find that a number of query suites have been proposed for benchmarking the KBSs. These benchmarks queries could possibly be used to evaluate learning ontologies in two main ways. Firstly, to check what language constructors are used in an ontology and secondly, to check what query results could be generated in relation to answering the benchmark queries. In the sections below, the evaluation methodology we follow is introduced.

7.1.4 A Methodology for Evaluating Learning Ontologies

In this section, we describe the methodology that we followed to conduct the ontology evaluation. In our approach, we use an evaluation methodology that includes clear evaluation goals, evaluation criteria and an evaluation process. An ontology evaluation methodology involves several essential elements. Here, we consider various aspects of evaluating learning ontologies. They include these elements:

1. Evaluation goals
2. Learning ontologies to be evaluated
3. Evaluation methods
4. Evaluation criteria
5. Conducting evaluation of learning ontologies
6. Analysing results of ontology evaluation.

Each element of this methodology is briefly introduced in the following subsections.

Evaluation Goals

In order to conduct an ontology evaluation, we need to have a set of clear goals; or we need to clearly know ‘why do we conduct this evaluation?’ In this evaluation, we have these two main goals:

1. Interrogate what benchmark queries can be answered by a given ontology.
2. Identify what inferences are possible on a learning ontology in answering the benchmark queries.

Select Evaluation Methods

In this work, we have identified the OWL 2 Learn benchmark queries as the evaluation method and it is elaborated in Section 7.6. We expect to use it to evaluate the structural and functional aspects of an ontology.

In evaluating a specific learning ontology, we need to write benchmark queries that are specific to that ontology yet analogous to OWL 2 Learn queries. First we write benchmark queries to evaluate the MQ ontology. When we want to evaluate another ontology, we have to write analogous queries for that particular learning ontology. Hence, here we also rewrite the analogous queries for the CSU ontology.

Sample Ontologies for Evaluation

In this chapter, we evaluate two sample learning ontologies that we have developed for the computing departments of two different institutions: Macquarie and Charles Sturt Universities. These two ontologies have already been analysed and their details have already been provided in Chapter 4. They include a number of concepts, object properties and data properties. We have populated the sample ontologies with some sample data/instances for the evaluation.

Evaluation Criteria

We need evaluation criteria to check whether we achieved the evaluation goals or 'how did we achieve the goals?' The evaluation criteria needs to be defined before we start the evaluation. We use these criteria to evaluate the query-answering and inference capabilities of a given learning ontology:

- Which benchmark queries can be answered by a specific ontology?
- What inferences of the OWL 2 Learn profile are possible and what inferences are not possible on a given learning ontology and by the proposed query suite?

Conducting the Evaluation of Learning Ontologies

The evaluation environment that we use for this ontology evaluation consists of several tools. They are predominantly the ontology editing tool, Protégé; the DL reasoner HermiT; and, the SPARQL query language. First, we load the populated ontology in Protégé and then we also start the Hermit reasoner that is required for inferencing as a part of query-answering. Then we execute each query in the query suite one by one (or as a batch). During the execution of a

query, if any errors are raised by the SPARQL query parser we have to correct our SPARQL query. If no errors are raised, Protégé will display the query results.

The steps that we follow to complete the evaluation of the learning ontologies make up the evaluation process. They are elaborated in Section 7.7.

Analysing the results of ontology evaluation

Here we elaborate on how we execute the benchmark queries on a sample ontology according to the evaluation process. The evaluation process generates a set of query results. We analyse the query results to understand the inferences involved in them and the capabilities of the ontologies.

Based on this analysis, we are able to determine the query-answering and inference capabilities of a given learning ontology. If a particular learning ontology is capable of answering all the benchmark queries, we can conclude that it has the full expressivity of the OWL 2 Learn profile. If some queries cannot be answered on a specific learning ontology then we can conclude that it does not have the full expressivity of the OWL 2 Learn profile.

In the next section we elaborate on current ontology benchmarks.

7.2 Current Ontology Benchmarks and Their Features

Here, we provide an overview of ontology benchmarks that we found in the literature. Among them, some of the benchmarks—LUBM, UOBM, Berlin SPARQL Benchmark (BSBM) and DBPedia SPARQL Benchmark (DBPSB)—have been discussed broadly in the literature (Morsey, Lehmann, Auer, & Ngomo, 2011). These benchmarks consist of numerous elements – query suites, KBS, test data/instances, etc.

The LUBM benchmark has been proposed as a method for benchmarking KBS that support inference on RDF/RDFS or OWL files (Guo et al., 2005). LUBM has been used to evaluate KBS with different reasoning and storage capabilities and to choose a suitable KBS for a large OWL based system. LUBM is based on three main goals: 1. supporting extensional queries; 2. arbitrary scaling of instances/data; and, 3. benchmarking an ontology of moderate size and complexity. LUBM includes 14 queries and they include different query constructs of SPARQL query language. Answering the LUBM queries involve different number of classes and properties of the learning ontologies. However, answering the queries of LUBM query suite involves only a few OWL constructors.

Subsequently, UOBM has been proposed as an extension to LUBM to evaluate three ontology repositories: DLDB-OWL, OWLIM and Minerva (Ma et al., 2006). In addition to the improvement of instance generation, the UOBM provides a complete coverage of OWL Lite and OWL DL constructors. Compared to LUBM, the UOBM benchmark covers all the OWL Lite constructors and OWL DL constructors (Ma et al., 2006). However, UOBM is based on OWL and not OWL 2. For this reason, UOBM does not cover the new features of OWL 2 (Golbreich et al., 2009) and the OWL 2 Learn profile that we introduced.

In addition to the two benchmarks, LUBM and UOBM, other ontology benchmarks have been proposed over time. However, they focus more on using different constructs of the SPARQL language to evaluate different RDF storages with different scalabilities. They also have focused on different domains other than on the learning domain and paid less attention to reasoning on ontologies with different inference capabilities. A summary of the prominent features of the current SPARQL benchmarks is given in Table 7.1. In that #Qs is the number of queries.

The OWL 2 constructors and inferences that are involved in answering benchmark queries are the important features of a query suite that is required for our purpose. We could not find these details and details of the queries that were used in some of the benchmarks. The queries in all these ontology benchmarks have been written in the SPARQL query language. In the next section we provide an overview of the SPARQL query language.

7.2.1 SPARQL Query Language

Two categories of query languages (QLs) for the Semantic Web have been identified in (Sirin & Parsia, 2007):

1. RDF-based QLs - RDQL3, SeRQL4 and W3C recommendation SPARQL.
2. Description Logic (DL)-based QLs – DIG ask queries and nRQL.

As SPARQL continued to gain popularity as a QL for the Semantic Web, the QLs for the Semantic Web have evolved in two directions. Firstly, making improvements or extensions to SPARQL to make it suitable for querying OWL 2 ontologies, and, secondly, developing totally new QLs which would be capable of querying OWL 2 ontologies.

SPARQL 1.1 has been introduced with some new features to overcome the weaknesses in SPARQL 1.0 (Arenas, Gottlob, & Pieris, 2014). SPARQL-DL is a powerful and expressive query language for OWL-DL that can combine TBox, RBox and ABox queries (Sirin &

Parsia, 2007). SPARQL-DL is a subset of SPARQL and the OWL-DL reasoners can be used to reason on it. SPARQL-DL has been aligned with SPARQL to improve the interoperability of applications on the Semantic Web.

Table 7.1: A comparison of the ontology benchmarks

#	BM	Aim/Purpose	Application Domain	ABox	#Qs	What is evaluated
1	LUBM (Guo et al., 2005)	Supporting extensional queries, arbitrary scaling, of data and ontology of moderate size & complexity.	Learning	RDF and OWL	14	KBS using a University ontology
2	UOBM (Ma et al., 2006)	Provide a benchmark for an improved evaluation of existing ontology systems	Learning	RDF and OWL	15	KBS using two University ontologies
3	BSBM (Bizer & Schultz, 2009)	Evaluate the performance of native RDF stores	e-commerce	Native RDF	25	Data from multiple data sources
4	SP^2Bench (Schmidt, Hornung, Lausen, & Pinkel, 2009)	testing the performance of SPARQL engines	Digital Bibliography	RDF	17	Large amount of data in DBPL data set
5	DBPSB (Morsey et al., 2011)	Propose a SPARQL benchmark procedure for benchmark creation.	DBpedia	RDF	25	Triple stores
6	SRBench (Zhang, Duc, Corcho, & Calbimonte, 2012)	Propose a benchmark for comparing streaming RDF/SPARQL engines	streaming RDF/SPARQL engines	RDF	17	streaming RDF/SPARQL engines
7	ParlBench (Tarasova & Marx, 2013)	Propose a benchmark for evaluating parliamentary proceedings with different scaling	Parliamentary proceedings	RDF	19	Parliamentary proceedings
8	Object-UOBM (Ledvinka & Křemen, 2015)	Propose an ontological benchmark for domain specific object-oriented access	Learning/ education	OWL	8	Ontological storages
9	DBOD (Wang, Staab, & Tiropanis, 2016)	Map OLAP operations to SPARQL and construct analytic benchmarks	DBPedia	RDF	12	SPARQL engines

Sematic entailment relations are considered as another way to use SPARQL queries on OWL ontologies. To query the OWL 2 ontologies using SPARQL, basic graph pattern matching has to be defined using semantic entailment relations instead of explicitly given graph structures (Glimm et al., 2013). A way for a range of standard Semantic Web entailment relations has been proposed in (Glimm et al., 2013). Such extensions of the SPARQL semantics are called SPARQL entailment regimes. An entailment regime defines: which entailment relation is used, which queries and graphs are well-formed for the regime, how the entailment is used, and what kinds of errors can arise (Glimm et al., 2013). An entailment regime specifies two main things: 1. a subset of RDF graphs that are well-formed for the

regime, and 2. an entailment relation between subsets of well-formed graphs and well-formed graphs (Glimm et al., 2013).

A SPARQL query includes variables for instances to be retrieved and a set of triples on the type concepts, instances constraints on them that are being queried. In addition to them, SPARQL uses the keyword FILTER to provide additional constraints. SPARQL also includes built in functions to perform general functions such as finding the sum, average, and maximum.

In this chapter, we analyse the query suites of the two benchmarks: LUBM and UOBM that have been written in SPARQL. We write the benchmark queries in SPARQL that are analogous to the UOBM query suite. These queries are written according to the SPARQL 1.1 Query Specification that is elaborated in (Harris, Seaborne, & Prud'hommeaux, 2013).

7.3 Query Suites and Inference Capabilities of LUBM and UOBM

We studied the query suites of the two ontology benchmarks LUBM and UOBM to see whether we could adopt them to evaluate the capabilities of the learning ontologies. Query-answering in LUBM and UOBM involve some inferences in the ontologies being evaluated. The inferences involved in answering the benchmark queries are associated to specific OWL 2 Learn constructors used in the learning ontologies. In our work, we see what OWL 2 Learn constructors are related to the LUBM and UOBM benchmark queries, in order to select a benchmark query suite to evaluate the OWL 2 Learn ontologies.

7.3.1 The LUBM Query Suite and its Capabilities

The 14 queries that are used in the popular ontology benchmark LUBM (Guo et al., 2005) could be possible candidates for evaluating the OWL 2 Learn ontologies. The benchmark queries in LUBM have been proposed considering five main factors: input size, selectivity, complexity, assumed hierarchy information and assumed logical inference (Guo et al., 2005). The main goals of this benchmark are to support queries on instances of classes, arbitrary scaling of data and ontology of moderate size and complexity. However, the LUBM benchmark queries do not involve, or are not based on, complex DL reasoning. LUMB does not rely on reasoning with disjunction or cardinality restrictions.

Answering the LUBM queries involve only a limited number of OWL Lite constructors to support inference on the ontologies. Those constructors are shown in Table 7.2 in bold. Even though the inference factors of LUBM are useful, they are not sufficient to evaluate the OWL

2 Learn ontologies. Hence, the LUBM does not become the right candidate to evaluate the OWL 2 Learn ontologies. In the next section, we examine the possibility of using the UOBM query suite that is an extension to LUBM.

Table 7.2: OWL Lite constructors supported by LUBM (source: Ma et al., 2006)

OWL Lite Constructors		
Property Characteristics:	*RDF Schema Features:*	*Property Restrictions:*
_ **ObjectProperty**	_ **rdfs:subClassOf**	_ allValuesFrom
_ **DatatypeProperty**	_ **rdfs:subPropertyOf**	_ **someValuesFrom**
_ **inverseOf**	_ **rdfs:domain**	
_ **TransitiveProperty**	_ **rdfs:range**	*(In)Equality:*
_ SymmetricProperty		_ equivalentClass
_ FunctionalProperty	*Restricted Cardinality:*	_ equivalentProperty
_ InverseFunctionalProperty	_ minCardinality (only 0 or 1)	_ sameAs
	_ maxCardinality (only 0 or 1)	_ differentFrom
Class Intersection:	_ cardinality (only 0 or 1)	_ AllDifferent
_ **IntersectionOf**		_ distinctMembers

Note: OWL Lite constructors used only in LUBM are given in **bold characters**.

7.3.1 The UOBM Query Suite and its Capabilities

Answering the 15 queries used in the UOBM benchmark (Appendix: Table A.3) involve different inferences on a learning ontology. The UOBM queries cover all the OWL constructors involved in the LUBM queries. In addition to that, the UOBM queries cover the OWL constructors of the OWL Lite and OWL DL (Ma et al., 2006) that are shown in Table 7.3.

OWL Lite has the DL expressivity of *SHIF(D)* and OWL DL has the DL expressivity of *SHOIN(D)* (Wang, Parsia, & Hendler, 2006). OWL 2 Learn profile has the DL expressivity of *SHIQ(D).* Answering UOBM queries involve OWL DL that has a higher expressivity than OWL 2 Learn and the OWL DL has constructors that are counterparts of the OWL 2 Learn constructors. Hence, UOBM becomes a potential candidate for evaluating the OWL 2 Learn ontologies. A detailed analysis of the OWL constructors and inferences involved in each UOBM query is given in the next section.

7.4 The UOBM Queries, Inferences and the OWL 2 Constructors

Answering each UOBM query suite involves several inferences on a learning ontology. We analysed each UOBM query to identify what OWL 2 constructors are involved in those inferences and required to answer each of them.

7.4.1 Inferences in UOBM Queries

The inferences associated with each UOBM query are listed in the Appendix of (Ma et al., 2006). When a SPARQL query is executed on an ontology, each query atom is evaluated by a query parser and the inferences are done by a DL reasoner to generate the query results. These inferences are associated to specific OWL 2 constructors. Hence, the analysis of the inferences associated with each query helps us to identify the OWL 2 constructors required in answering each query. Specific inferences and the OWL 2 constructors involved in each UOBM query is given in the Appendix: Table A.4.

Table 7.3: OWL constructs supported by UOBM (source: Ma et al., 2006)

OWL Lite		**OWL DL**
RDF Schema Features:	*Property Restrictions:*	*Class Axioms:*
_ rdfs:subClassOf	_ allValuesFrom	_ oneOf, dataRange
_ rdfs:subPropertyOf	_ someValuesFrom	_ disjointWith
_ rdfs:domain		_ equivalentClass (applied to class
_ rdfs:range	*Restricted Cardinality:*	expressions)
	_ minCardinality (only 0 or 1)	_ rdfs:subClassOf (applied to class
Property Characteristics:	_ maxCardinality (only 0 or 1)	expressions)
_ ObjectProperty	_ cardinality (only 0 or 1)	*Boolean Combinations of Class*
_ DatatypeProperty		*Expressions:*
_ inverseOf	*(In)Equality:*	_ unionOf
_ TransitiveProperty	_ equivalentClass	_ complementOf
_ SymmetricProperty	_ equivalentProperty	_ intersectionOf
_ FunctionalProperty	_ sameAs	*Arbitrary Cardinality:*
_ InverseFunctionalProperty	_ differentFrom	_ minCardinality
	_ AllDifferent	_ maxCardinality
Class Intersection:	_ distinctMembers	_ cardinality
_ IntersectionOf		*Filler Information:*
		_ hasValue

7.4.2 Inferences and the OWL 2 Constructors Associated to UOBM Query Suite

We tabularise the OWL 2 constructors involved in answering each UOBM query in Table 7.4. One UOBM benchmark query has several associated OWL 2 constructors that are shown

by a tick. The 15 queries of the UOBM benchmark exploits various constructors of the OWL 2 constructors. Several OWL 2 constructors: *Class(), ClassAssertion()* and *ObjectPropertyAssertion()* are required in answering all the UOBM queries. Also, answering most of the queries involve the OWL 2 constructor *ObjectProperty()*. In addition to them, each query includes several other OWL 2 constructors as listed in Table 7.4.

The OWL 2 constructor *SubPropertyOf()* is used in queries 2, 3, 4, 8, 9, 12 and 13. *ObjectIntersectionOf()* is involved only in the query 12. The constructor *UnionOf()* is required to answer queries 4 and 14. The universal quantification, *ObjectAllValuesFrom()* is involved in answering queries 9, 12 and 14. The existential quantification, *ObjectSomeValuesOf()* is required to answer queries 3, 4 and 8. *Domain()* and *Range()* of a property are required to answer queries 3, 4, 10 and 12. The *FunctionalProperty()* is involved in answering only the query 11. Answering queries 5 and 7 involves the *TransitiveObjectProperty(). SubObjectPropertyOf()* is involved in answering the query the query 8. Answering the query 6 requires inference on the OWL 2 constructor *InverseObjectProperties()*. Cardinality restrictions, *Object{Max/Min/Exact}Cardinality()* are involved in answering queries 3, 12, 13 and 15. The constructor *DisjointClasses()* is involved only in answering queries 4 and 14.

Table 7.4: An analysis of the OWL 2 constructors involved in the UOBM queries

OWL 2 Constructor	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8a**	**Q8b**	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15**
Class()	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
SubClassOf()		√	√	√				√	√	√			√	√		
ObjectIntersectionOf()													√			
UnionOf()				√											√	
ObjectAllValuesFrom()										√			√		√	
ObjectSomeValues From()		√	√			√		√	√							
ObjectProperty	√	√	√	√	√	√		√	√	√		√	√	√	√	√
Domain(), Range()		√	√								√		√			
ClassAssertion(), Object PropertyAssertion()	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
FunctionalProperty()												√				
TransitiveObject Property()					√		√									
SubObjectPropertyOf() SubDataPropertyOf()								√	√							
InverseObject Properties()						√										
Object{Max/Min/Exact} Cardinality()			√										√	√		√
DisjointClasses()				√											√	
ObjectComplementOf()															√	
SameAs(), AllDifferent()							√					√				√
OneOf(), HasValue()									√							
SymmetricObject Property(),							√				√					

In addition to this, answering only query 14 involves *ObjectComplementOf()* and answering queries 7 and 11 involves equality or inequality, *SameAs()* and *DifferentFrom()*. Answering query 8 on OWL DL ontology used in the test involves nominal, *OneOf()* (Ma et al., 2006). Inference on *SymmetricObjectProperty()* is required in answering only query 10. Here, query 8 has been run on two different ontologies that have different DL expressivities. The first instance of query 8 (Table 7.4: 8a) has been run on an OWL Lite ontology and the second instance of query 8 (Table 7.4: 8b) has been run on an OWL DL ontology (Ma et al., 2006). Hence, the second instance of query 8 (Table 7.4: 8b) involves more inferences than the first inference of query 8 (Table 7.4: 8a). This indicates that the type of inferences involved in answering a query depends on the ontology being queried.

7.5 UOBM Benchmark Queries and Evaluating the OWL 2 Learn Ontologies

The two ontology benchmarks: the LUBM and the UOBM have focused on achieving scalability of query-answering on KBS. It has been pointed out that, due to the use of incomplete OWL 2 reasoners, at least one query in such a query suite cannot be executed (Grau, Motik, Stoilos, & Horrocks, 2012). This has adverse effects on critical applications such as health care. Several limitations of the empirical completeness testing have been highlighted in (Grau et al., 2012). They include:

1. Use of tests that are not generic – LUBM and UOBM use instance generators to generate the test data. The test data used in the tests have repetitive and fixed structures.
2. Use of non-exhaustive test data – test data is used to test the KBSs w.r.t. only a limited number of data sets.
3. Non-verifiability of the query answers – as the complete reasoners fail to work on large data sets, the results generated by them it may not be possible to use them to verify the results generated by the incomplete reasoners.

These factors could affect the evaluation of OWL 2 learn ontologies differently.

1. Instead of using an instance generator to generate the test data we use the instances of concepts and roles that have been mapped from a relational database. In Chapter 6 we already discussed database to ontology mapping to get the instances from data in the legacy databases. The relational database has been populated with those data over time.

2. The ontologies that we evaluate have been built independent of the benchmark queries. Database to ontology mapping to populate the ontology with instances is expected to be completed. To the best of our knowledge the data sets are sufficient to evaluate the capabilities of the learning ontologies.
3. Learning domain includes a comparatively small data set compared to larger domains such as the medical domain with over 1,000,000 instances. However, we don't take scalability as a criterion in this evaluation.
4. Instead of using the same data set for evaluating the capabilities of the ontologies we use different data sets that are specific to each institution. In our work, we evaluate different institutional ontologies with different instances.

7.5.1 The UOBM Queries and the OWL 2 Learn Profile

Here, we compare the OWL constructors that are involved in the inferences of the UOBM queries with the OWL 2 constructors that are used in the OWL 2 Learn profile. Most of the OWL constructors involved in answering the UOBM queries are part of the OWL 2 Learn profile. However, a few queries of the UOBM uses OWL constructors have been excluded from the OWL 2 Learn profile. They are equality, inequality, symmetric object property and nominal and are given in the last three lines of the Table 7.4. We have listed those OWL 2 constructors in Table 7.5.

Table 7.5: OWL DL constructors that are not in OWL 2 Learn profile

OWL DL Constructors required to answer UOBM and excluded from OWL 2 Learn
(In)Equality: equivalentProperty(), sameAs(), differentFrom(), AllDifferent(), distinctMembers()
Property Characteristics: SymmetricProperty()
Class Axioms: oneOf, dataRange()
Filler Information: hasValue()

The six queries of the UOBM query suite (Appendix: Table A.3): queries 7, 8, 10, 11, 14 and 15 involve these OWL 2 constructors that are excluded from the OWL 2 Learn profile. In the analysis we conducted in Section 7.4, we see that only ten queries (including query-8) of the UOBM are included in the OWL 2 Learn profile. Five queries, 7, 10, 11, 14 and 15 that demand the OWL 2 constructors that are excluded from the OWL 2 Learn profile are excluded from the query suite that we derive from the UOBM.

We also see that some of the OWL 2 Learn constructors are not utilised in inferences in answering the queries in the UOBM query suite. They are qualified cardinality restrictions,

disjoint properties and cardinality restrictions on data properties (Table 7.6). This shows that UOBM query suite involves a majority of the inference features of the OWL 2 Learn profile. However, it does not cover all the OWL 2 constructors of the OWL 2 Learn profile. Hence the UOBM query suite is not directly applicable to evaluate the OWL 2 Learn ontologies. This indicates that we need a customised or an enhanced version of the UOBM to evaluate the OWL 2 Learn ontologies.

7.6 A Query Suite for Evaluating the Capabilities of OWL 2 Learn Ontologies

Based on the analyses that we have completed in the above sections, we attempt to devise a query suite that would help us to evaluate the capabilities of the query-answering and inference learning ontologies. As we cannot apply all the 15 queries of the UOBM to the OWL 2 Learn profile we adopt only ten queries from the UOBM and exclude five queries of the UOBM (queries 7, 10, 11, 14 and 15).

Table 7.6: OWL 2 Learn Constructors that are not used in the UOBM query suite

OWL 2 Learn Constructors not used in OWL DL		
Qualified Cardinality on Object Properties:	*Qualified Cardinality on Data Properties:*	*Disjoint Properties:*
_ ObjectMaxCardinality(n R D)	_ DataMaxCardinality(n R D)	_ DisjointObjectProperties(p1, p2)
_ ObjectMinCardinality(n R D)	_ DataMinCardinality(n R D)	_ DisjointDataProperties(p1, p2)
_ ObjectExactCardinality(n R D)	_ DataExactCardinality(n R D)	

Even though answering query 8 on an OWL DL ontology involves Nominal (O) answering query 8 on an OWL Lite ontology does not involve Nominal. Hence, the UOBM query 8 is not excluded from the derived query suite. In addition to that, we introduce three new queries which use the three constructors of the OWL 2 Learn profile which are not used in the UOBM.

7.6.1 Writing Queries Analogous to UOBM Queries

The queries in the UOBM benchmark have been run on two University ontologies with data for 1, 5 and 10 universities. The learning ontologies we need to evaluate are different to the ontologies used in (Ma et al., 2006). That makes it difficult for us to use the UOBM benchmark queries as it is on another learning ontology. We need to write queries analogous to the queries in the UOBM query suite each time we evaluate a new learning ontology. Here we discuss writing the analogous queries to the MQ ontology, one of the sample ontologies to be evaluated. The query suite and the ontology used in the UOBM are comparable to the MQ ontology and other OWL 2 Learn ontologies in the following ways.

Use of Synonyms

The two ontologies used in the UOBM benchmark and the OWL 2 Learn ontologies that we need to evaluate include some common concepts and some common object/data properties. For example, both of them include the terms such as *student*, *lecturer*, and *course*. Still, we notice that some synonymous terms also have been used in the UOBM ontologies and OWL 2 Learn ontologies for the concept and object properties. For example, the university ontology used in UOBM uses the terms, *employee*, and *alumni* whereas in the MQ ontology we use the terms *staff* and *graduate*.

Hence, we have to rewrite the selected UOBM queries to make them executable on the learning ontologies that we evaluate. In writing the queries analogous to the UOBM we kept the original format of the UOBM queries. We also attempt to keep the concept and property names the same. If this is not possible we use synonyms. For example, the UOBM query 1 uses the concept *UndergraduateStudent* and the object property *takesCourse*. In the analogous query for the MQ ontology we use the same concept *UndergraduateStudent* and the synonym of the *takesCourse*, studies. In case we don't find matching concepts or properties to the UOBM ontology in a learning ontology, we select concepts and properties with a similar ontology structure for querying. For example, query 5 involves the transitive property, *subOrganisationOf* on classes: University, Faculty and Research Group. However, in our work, we consider only a single university and cannot use the concept, University. Hence, we use the transitive property, *prerequisiteOf* on the classes Unit and Prerequisite (Appendix: Table A.3).

Use of Domain Specific Classes and Properties

The UOBM benchmark queries include concepts and properties that are specific to the ontologies that are used in the UOBM benchmark. For example, the UOBM queries 8 and 13 include the concepts, *SportsLover* and *PeopleWithHobby*. It is less likely that a real-world university would need such information in a learning system. Hence, we replace those concepts used in queries 8 and 13 by realistic and relevant concepts, *Lecturer* and *TeachingStaff* in the MQ ontology to get the analogous queries 8 and 13 that are given in the Appendix: Table A.3.

7.6.2 New Benchmark Queries for Evaluating OWL 2 Learn Ontologies

Here we introduce three new benchmark queries that we propose to evaluate the OWL 2 Learn ontologies. These new queries include the OWL 2 constructors given in Table 7.6 that are not covered by the UOBM queries.

New Query 1 with Data Properties and Learning Object Metadata

The first query we introduce involves metadata on learning resources and user profile attributes that are defined as data properties in the MQ ontology. For example, the new query 1 in (Table 7.7) retrieves the learning resources that are of the type *presentation* and in the format *pdf*. This query involves a defined class *LearningResource* and two subdata properties, *contentType* and *format* of the data property *lomAttribute*. These two data properties have *xsd:string* values as the range of each (Table 7.7).

New Query 2 with Cardinality Restrictions on Data Property

The second query we introduce involves cardinality restriction on the data property. This query in (Table 7.7) is intended to retrieve the units with less than three assessment items. This query is satisfied by the MQ ontology by using the class *Unit* that includes a cardinality restriction on the data type property *noOfAssignments*. That is: *noOfAssignments min 3 xsd:integer*.

Table 7.7: New benchmark queries for OWL 2 Learn ontologies

#	Query in Natural Language	SPARQL Query	Related constraints
1	Find learning resources that are only presentations and in the pdf format.	SELECT DISTINCT ?x WHERE {?x rdf:type :LearningResource. ?x :contentType ?y. ?x :format ?z FILTER ((?y='presentation'^^xsd:string) && (?z='pdf'^^xsd:string))}	Range(contentType, xsd:string) Range(format, xsd:string)
2	Find units with less than three assignments.	SELECT ?x COUNT(?y) WHERE {?x rdf:type :Unit. ?x :hasAssignment ?y.} GROUP BY ?x HAVING (COUNT(?y) <3)	noOfAssignments min 3 xsd:integer
3	Find all assessment tasks with their due date and the return date.	SELECT DISTINCT ?x ?y ?z WHERE {?x rdf:type :AssessmentTask. ?x :dueDate ?y. ?x :returnDate ?z.}	DisjointDataProperties (:dueDate :returnDate)

OWL 2 Learn profile includes the OWL 2 constructors disjoint object and data properties and they have been used in the sample OWL 2 Learn ontologies. The new query 3 is written to retrieve all the assessment tasks with their *dueDate* and the *returnDate* from the MQ ontology. Answering the new query 3 involves *dueDate* and *returnDate* that have been

defined as disjoint data properties in the ontology as in (64). So, the *dueDate* of an assignment, must be different from its *returnDate*.

$$DisjointDataProperties(:dueDate :returnDate) \quad (64)$$

New Query 3 with Disjoint Data Properties

With these three new queries we form a new query suite of 13 benchmark queries to evaluate the OWL 2 Learn ontologies. In the next section we provide a discussion on the inferences involved in the queries of the new query suite that we have derived.

7.6.3 The Inferences Involved in the Three New Queries

The query-answering process involves several inferences performed by the DL reasoner on the MQ ontology. The ten OWL 2 Learn queries that we have derived from, and analogous to, the UOBM queries involve analogous inferences to the UOBM queries as well. Those analogous inferences depend on the query, the learning ontology we are able to logically derive from them based on the query results. The analogous inferences for the ten analogous queries are given in Table A.5.

When we consider the first new query and the query results, it involves the inferences below with conjunction and data type properties as in (65):

$$\begin{gathered} <x\ rdf{:}type\ :LearningResource>, <x\ contentType\ y>, <x\ format\ z> \\ <y = 'presentation'\text{^^}xsd{:}string>, <z= 'pdf'\text{^^}xsd{:}string> \end{gathered} \quad (65)$$

Here the reasoner checks whether the instance 'x' is of the type LearningResource has a *xsd:string* value as the *datatype* and as the *format* of it.

The second new query involves the inferences that are listed in (66):

$$<x\ rdf{:}type\ Unit>, <x\ noOfAssessments\ y> <y <3\text{^^}\ xsd{:}integer > \quad (66)$$

Here the reasoner checks whether there at most three assessment items for a given unit. Here the *noOfAssessmentItems* has been defined as a data type property.

The new query 3 involves inferences on disjoint data properties. Therefore, it checks whether the *dueDate* is different from the *returnDate*. They are given in (67):

$$\begin{gathered} <x\ rdf{:}type\ AssessmentTask>, <x\ dueDate\ y>, <x\ returnDate\ z>, \\ DisjointDataProperties(dueDate\ returnDate) \end{gathered} \quad (67)$$

The complete query suite we derive includes 13 queries. In the next subsections we provide a discussion on writing the queries that are analogous to the UOBM queries and the three new queries we introduce.

7.6.4 The Inferences Involved in the Queries with Cardinality Restrictions

Answering the UOBM queries 3, 12, 13 and 15 involve unqualified cardinality restrictions: *ObjectMinCardinality(n R), ObjectMaxCardinality(n R), ObjectExactCardinality(n R).* However, OWL 2 Learn profile includes qualified cardinality restriction. So, when we write queries that are analogous to queries 3, 12, 13 and 15 of UOBM we use the qualified object cardinality restrictions: *ObjectMinCardinality(n R D), ObjectMaxCardinality(n R D), ObjectExactCardinality(n R D)*. For example, in answering the new query 2, qualified minimum cardinality restriction is used on the MQ ontology inference. The defined class *Unit* includes the qualified cardinality restriction: *ObjectMinCardinality(noOfAssignments 3 Unit)*.

7.6.5 OWL 2 Learn Benchmark Query Suite and the OWL 2 Constructors

We name the new query suite that we derive to evaluate the learning ontologies as the OWL 2 Learn benchmark query suite. The complete query suite includes the ten analogous queries and these three new queries that equals 13 queries. The complete set of queries included in the OWL 2 Learn benchmark are listed in Table 7.8.

Answering the OWL 2 Learn query suite involves inference in association to different OWL 2 Learn constructors in the sample ontologies. We list the OWL 2 Learn constructors that are required in answering all the new queries in Table 7.9. The first ten queries in Table 7.9 are analogous to the UOBM queries. Hence, the OWL 2 constructors involved in the analogous queries are same as the constructors involved in the counterpart queries in the UOBM query suite. The three new queries are listed in the last three columns of Table 7.9. The constructors that are required to satisfy queries 11 to 13 are shown in the last three rows of Table 7.9.

7.7 Conducting an Ontology Evaluation

In these subsections we briefly introduce the elements of the evaluation environment we use.

Protégé - Here we use Protégé as the ontology evaluation environment. Protégé is used to load each of the learning ontology that we have already developed and populated. The HermiT reasoner and SPARQL query interface are available as plugins of Protégé. The SPARQL query interface is capable of editing, executing and displaying the query results.

Table 7.8: OWL 2 Learn benchmark query suite

Q#	Query in natural language	Query in SPARQL
1	Find all the undergraduate students who study ISYS114.	SELECT DISTINCT ?x WHERE {?x rdf:type :UndergraduateStudent. ?x :studies :ISYS114 }
2	Find all the staff members.	SELECT DISTINCT ?x WHERE {?x rdf:type :Staff.}
3	Find all the students of the Computing department.	SELECT DISTINCT ?student WHERE {?x rdf:type :Student. ?x :isStudentOf :Computing }
4	Find all the learning resources authored by teaching staff of the Computing department.	SELECT DISTINCT ?x WHERE {?x rdf:type :LearningResource. ?x :hasAuthor ?y. ?y rdf:type :TeachingStaff. ?y :isStaffOf :Computing}
5	Find all the prerequisites of COMP365.	SELECT ?x WHERE { ?x rdf:type :Prerequisite. ?x :isPrerequisiteOf+ ?y. FILTER (?y=:COMP365)}
6	Find all the graduated students of the Computing department.	SELECT DISTINCT ?x WHERE { ?x rdf:type:Person. :Computing :hasGraduate ?x.}
7	Find all the lecturers of the Computing department.	SELECT DISTINCT ?x WHERE {?x rdf:type :Lecturer. :Computing :hasStaff* ?x}
8	Find all the undergraduate units of the Faculty of Science and Engineering.	SELECT ?x WHERE { ?x rdf:type :UnitUG. ?x :isTaughtBy ?y. ?y :isStaffOf ?z. ?z subOrganisationOf :FSE}
9	Find all the students who study the units taught by Debbie Richards.	SELECT DISTINCT ?x WHERE{?x rdf:type :Student. ?x :studies ?y. ?y :hasLecturer :DebbieRichards}
10	Find all teaching staff who teach some units in the Computing department.	SELECT DISTINCT ?x WHERE {?x rdf:type :TeachingStaff. ?x :isStaffOf :Computing}
11	Find the learning resources that are only presentations and in the pdf format.	SELECT DISTINCT ?x WHERE {?x rdf:type :LearningResource. ?x :contentType ?y. ?x :format ?z FILTER ((?y='presentation'^^xsd:string) && (?z='pdf'^^xsd:string))}
12	Find the units with less than three assignments.	SELECT ?x COUNT(?y) WHERE {?x rdf:type :Unit. ?x :hasAssignment ?y.} GROUP BY ?x HAVING (COUNT(?y) <3)
13	Find all the assessment tasks with their due date and the return date.	SELECT DISTINCT ?x WHERE {?x rdf:type :AssessmentTask. ?x :dueDate ?y. ?x :returnDate ?z.}

Notes: + - SPARQL syntax for transitive properties, * - SPARQL syntax for subproperties

Table 7.9: The OWL 2 Learn Constructors Involved in the OWL 2 Learn BM Queries

OWL 2 Constructor	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**
Class()	√	√	√	√	√	√	√	√	√	√	√	√	√
SubClassOf()		√	√				√	√	√				
ObjectIntersectionOf()									√				
UnionOf()				√									
ObjectAllValuesFrom								√	√				
ObjectSomeValuesFrom		√	√				√						
ObjectProperty	√	√	√	√	√	√	√	√	√	√			
Domain(), Range()		√	√						√				
ClassAssertion, ObjectPropertyAssertion	√	√	√	√	√	√	√	√	√	√	√	√	√
TransitiveObjectProperty					√								
SubObjectPropertyOf SubDataPropertyOf							√			√			
InverseObjectProperties						√							
Object{Max/Min/Exact}Cardinality			√						√				
DisjointClasses				√									
DataProperty, DataPropertyAssertion, xsd:{integer, string, …}											√	√	√
Data{Max/Min/Exact}Cardinality												√	
DisjointDataProperties													√

DL Reasoner – In evaluating a learning ontology, we recommend the use of any DL reasoner that has a DL expressivity equal to, or higher than, an OWL 2 Learn profile. In this work, we used the DL reasoner HermiT that has been introduced as an OWL 2 reasoner. OWL 2 has the DL expressivity of SROIQ(D) that is a higher DL expressivity that the OWL 2 Learn profile.

SPARQL Query Language – SPARQL is the W3C recommended query language for the Semantic Web. The query suites of all the ontology benchmarks we analysed in this chapter have been written in SPARQL.

Another option for an evaluation environment in which to conduct this evaluation is to use the e-learning system prototype that we have developed. The current version of our prototype uses a GUI for composing the queries. It uses the reasoner RacerPro and the query language nRQL. They are core components of the RacerPro systems architecture (Haarslev et al., 2012). It is possible to translate SPARQL queries to nRQL queries and vice versa. SPARQL has been proposed as the standard query language for RacerPro (Haarslev et al., 2012). In this section, we evaluate the learning ontologies using Protégé with SPARQL plugin to execute the queries on learning ontologies.

The evaluation of each learning ontology is conducted by execution of the 13 benchmark queries (in Table 7.8), firstly on the MQ ontology. Secondly, the 13 analogous queries that

are given in Table 7.10 have been formulated for, and executed on, the CSU ontology. We observe each query's results.

Table 7.10: Analogous queries for the CSU ontology

#	Query in natural language	Query in SPARQL
1	Find all the undergraduate students who study ITC106.	SELECT DISTINCT ?x WHERE {?x rdf:type :UndergraduateStudent. ?x :studies :ITC106 }
2	Find all the staff members.	SELECT DISTINCT ?x WHERE {?x rdf:type :Staff.}
3	Find all the students of the School of Computing and Mathematics.	SELECT DISTINCT ?student WHERE {?x rdf:type :Student. ?x :isStudentOf :ComputingAndMaths }
4	Find all the learning materials authored by academic staff of the School of Computing and Mathematics.	SELECT DISTINCT ?x WHERE {?x rdf:type :LearningMaterial. ?x :hasAuthor ?y. ?y rdf:type :AcademicStaff. ?y :isStaffOf : ComputingAndMaths}
5	Find all the prerequisites of ITC309.	SELECT ?x WHERE { ?x rdf:type :Prerequisite. ?x :isPrerequisiteOf+ ?y. FILTER (?y=:ITC309)}
6	Find all the graduated students of the School of Computing and Mathematics.	SELECT DISTINCT ?x WHERE { ?x rdf:type:Person. :ComputingAndMaths :hasGraduate ?x.}
7	Find all the lecturers of the School of Computing and Mathematics.	SELECT DISTINCT ?x WHERE {?x rdf:type :Lecturer. :ComputingAndMaths :hasStaff* ?x}
8	Find all the undergraduate subjects of the Faculty of Business, Justice and Behavioral Studies (BJBS).	SELECT ?x WHERE { ?x rdf:type :SubjectUG. ?x :isTaughtBy ?y. ?y :isStaffOf ?z. ?z subOrganisationOf :BJBS}
9	Find all the students who study the subjects taught by Sudath Heiyanthuduwage.	SELECT DISTINCT ?x WHERE{?x rdf:type :Student. ?x :studies ?y. ?y :hasLecturer : SudathHeiyanthuduwage}
10	Find all the academic staff who teach some subjects in School of Computing and Mathematics.	SELECT DISTINCT ?x WHERE {?x rdf:type :AcademicStaff. ?x :isStaffOf :ComputingAndMaths}
11	Find the learning materials that are only presentations and in the *pdf* format.	SELECT DISTINCT ?x WHERE {?x rdf:type :LearningMaterial. ?x :contentType ?y. ?x :format ?z FILTER ((?y='presentation'^^xsd:string) && (?z='pdf'^^xsd:string))}
12	Find the subjects with less than three assignments.	SELECT ?x COUNT(?y) WHERE {?x rdf:type :Subject. ?x :hasAssignment ?y.} GROUP BY ?x HAVING (COUNT(?y) <3)
13	Find all the assessment items with their due date and the return date.	SELECT DISTINCT ?x WHERE {?x rdf:type :AssessmentItem. ?x :dueDate ?y. ?x :returnDate ?z.}

When a query is executed its results are displayed on the query interface in Protégé. Figure 7.1 shows the execution of the benchmark query 4 on MQ and CSU ontologies. We could execute all 13 queries on each sample ontology. Even though they generate similar results, they could vary slightly due to the differences in terminology and the structure of each ontology. An analysis of them is provided in the next section.

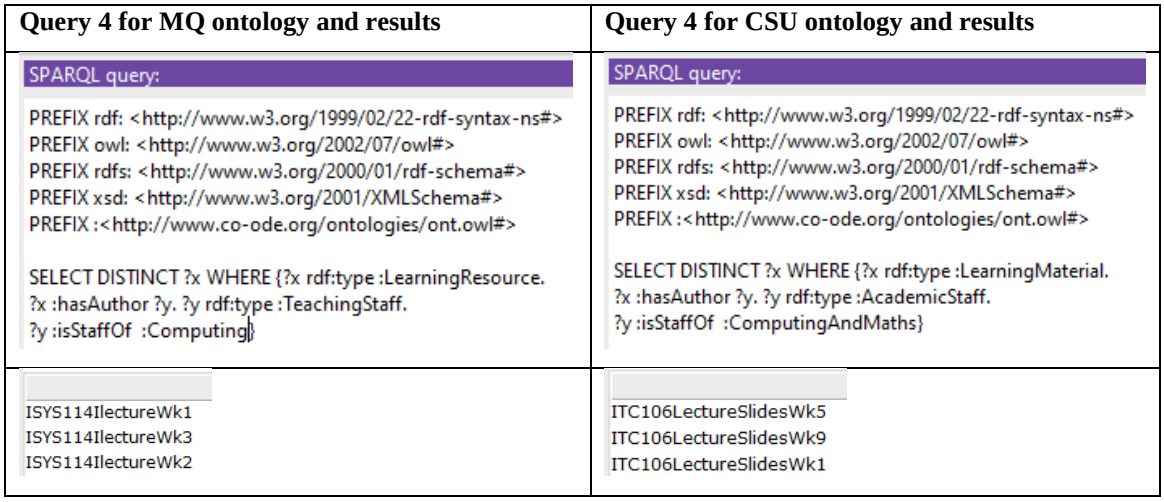


Figure 7.1: Executing benchmark queries in Protégé

7.8 Analysing the Results of the Ontology Evaluation

Even though both ontologies are capable of generating the query's results, 'how they generate the query results?' or 'what inferences are involved in satisfying them?' could be different. Hence, we analysed the inferences involved in satisfying the benchmark queries on each ontology and made a comparison.

7.8.1 Differences in the Learning Ontologies

We observe the differences between the two learning ontologies we analysed:

Size of the ontologies

These sample ontologies vary by the size of their TBox; the number of concepts, object properties, data properties and the other TBox constraints. The size of an ontology specifically affects the scalability and efficiency of query-answering (Ma et al., 2006). However, as we do not evaluate the learning ontologies by their scalability and efficiency we do not analyse their size.

Terminological differences in Concept Names

Another aspect that makes the learning ontologies different from each other is the terminology used in their TBox. Different names are given to their classes, object properties,

data properties and constraints. For example, when we consider the terms used for a concept, MQ ontology uses the terms *program*, *major* and *unit* whereas CSU ontology uses the terms *course*, *specialisation* and *subject*. Also, MQ ontology uses the terms *recourses* and *assessment tasks* that are synonymous with the learning materials and assessment items in the CSU ontology.

We also notice that synonymous object and data type properties are used in the two sample ontologies. For example, MQ ontology uses the object property *hasUnit* whereas CSU ontology uses the object property *hasSubject*. Again, the MQ ontology uses the data property *unitTitle* whereas the CSU ontology uses the data property *subjectName*. Of course, the names given to ABox instances are also different to each learning ontology that depends on the data.

Terminological difference needs to be considered especially in ontology merging and alignment. However, in our work, we do not attempt to merge or align the ontologies. We respect the fact that each institution would like to keep the terminology and other features unique. Hence, to facilitate this requirement we propose to use the adaptive e-learning system's framework with institution-specific learning ontologies.

7.8.2 Similarities in the Learning Ontologies

Even though the learning ontologies differ from each other in some aspects like their size and terminology, they are similar in some other aspects. We analyse the two sample ontologies to see whether they are similar in their ontological structures and inference capabilities.

Similarities in Ontological Structures

Each learning ontology is built considering the requirements of each institution. Still, most of the institutions have common requirements that make their ontological structures similar. However, it is possible for each institution to have minor differences due to their business rules. Business rules of a domain are specified as constraints in an ontology.

When we consider the MQ and the CSU ontology we identify many similarities in their ontological structures. The similarities between these learning ontologies exist between the structural elements: classes, class hierarchies, object properties, object property hierarchies, data properties, data property hierarchies and property characteristics. Also, the constraints: quantification; universal and existential; and cardinality restrictions used in the learning ontologies, could be similar.

The TBoxes of the sample ontologies have similar class hierarchies and property hierarchies. For example, MQ's ontology includes a hierarchy of learning resources that is similar to CSU's ontology hierarchy of learning materials. They also are similar by the type of the relations: inverse properties and transitive properties that are used to define the ontology. We have given an example for each structural element of each ontology in Table 7.11.

Table 7.11: Structural similarities in MQ and CSU ontologies

OWL 2 Learn Constructor	**An example from MQ Ontology**	**An example from CSU Ontology**
Classes	Class(UndergraduateStudent)	Class(UndergraduateStudent)
class inclusion	SubClassOf(UndergraduateStudent, Student)	SubClassOf(UndergraduateStudent, Student)
object properties	ObjectProperty(unitOf())	ObjectProperty(subjectOf())
object property hierarchies	SubObjectPropertyOf(AssignmentOf(), AssessmentTaskOf())	SubObjectPropertyOf(AssignmentOf(), AssessmentItemOf())
data properties	DataProperty(format)	DataProperty(format)
data property hierarchies	SubDataProperty(format, LOMAttribute)	SubDataProperty(format, LOMAttribute)
inverse properties	InverseProperties(hasGraduate(), isGraduateOf())	InverseProperties(hasGraduate(), isGraduateOf())
transitive properties	TransitiveProperties(isPrerequisiteOf())	TransitiveProperties(isPrerequisiteOf())
Universal constraints	ObjectAllValuesFrom(studies() UndergraduateUnit)	ObjectAllValuesFrom(studies() UndergraduateUnit)
existential constraints	ObjectSomeValuesFrom(isTeachingStaffOf(), Department)	ObjectSomeValuesFrom(isAcademicOf(), School)
cardinality constraints	studies(Student, Unit (ObjectMinCardinality(1)))	studies(Student, Subject (ObjectMinCardinality(1)))

Similarities in Inferences Involved in Query-Answering

The similar structures of the ontologies make the inferences performed on them by a reasoner also similar. They eventually lead the queries to generate similar answers as well. Most of the analogous queries on MQ ontology involve generating answers that include resources, people, units and the assessments. Similarly, the analogous queries on the CSU ontology involve generating answers that include materials, people, subjects and assessment items.

If we would consider inferences performed on the class, *Student* and its subclass *UndergraduateStudent* in MQ and CSU ontologies, they involve similar inferences. For example, if we would consider the query in (68):

Find all the students of computing department of MQ ontology. (68)

It involves the inferences in (69): *x is an undergraduate student, so, x is a student.*

UndergraduateStudent(x) → *Student(x)* → *x* (69)

We notice similarities in the inferences that involve other structural elements as well. Table 7.12 shows the similarities between the inferences that would be performed on the MQ ontology and the CSU ontology. The inferences involved answering each benchmark query on the two sample ontologies. MQ ontology and CSU ontology are given in the Appendix: Table A.5.

Based on the evaluation we conducted we obtained some insight into query-answering and inference capabilities of the two sample learning ontologies. We observed that both of them have similar structural and inference capabilities that are within the OWL 2 Learn profile. Both of the learning ontologies include the OWL 2 Learn constructors that are required to answer the 13 benchmark queries. Therefore, both of them can be considered to have complete query-answering and inference capabilities of the OWL 2 Learn profile.

7.9 Evaluation of the Adaptive E-Learning System's Framework

In Chapter 3 we suggested an *ontology-based plug and play architecture* for developing adaptive e-learning systems that would alleviate the duplicated effort in developing them. We also suggested that this framework is adaptable to each e-learning system and would allow institutions to keep their identity by using their own terminology, course and unit structures.

Here we discuss the evaluation of adaptability of the e-learning system's framework, to prove or disprove whether we have achieved the expected research goals. For this purpose, we have developed a proof of concept prototype of the adaptive e-learning system's framework. We also have developed two sample ontologies that are presented in Chapters 5 and 6. Here we evaluate two instances of the proof of concept prototype using the two sample ontologies that we developed for MQ and CSU. In this evaluation, we examine: 1) the possibility of using instances of the proof of concept prototype by just plugging institution-specific ontologies with no changes to the proof of concept prototype; and, 2) question whether this prototype allows each institution to generate answers to the queries with its own terminology.

Table 7.12: Similarities in inferences on MQ and CSU ontologies

OWL 2 Learn Constructor	Inference involved in MQ Ontology	Inference involved in CSU Ontology
Classes and instances	UndergraduateStudent(x) → x	UndergraduateStudent(x)→ x
class inclusion	UndergraduateStudent(x) → Student(x) → x	UndergraduateStudent(x) → Student(x) → x
object properties	unitOf(x, y) → x	subjectOf(x, y) → x
object property hierarchies	assignmentOf(x, y) → assessmentTaskOf(x, y) → x	assignmentOf(x, y) → assessmentItemOf(x, y) → x
data properties	format(x, 'pdf'^^xsd:String) → x	format(x, 'pdf'^^xsd:String) → x
data property hierarchies	format(x, 'pdf'^^xsd:String) → LOMAttribute(x, 'pdf'^^xsd:String)→ x	format(x, 'pdf'^^xsd:String) → LOMAttribute(x, 'pdf'^^xsd:String)→x
inverse properties	hasGraduate(x, y) → isGraduateOf(y, x) → y	hasGraduate(x, y) → isGraduateOf(y, x) → y
transitive properties	isPrerequisiteOf(x, y) → isPrerequisiteOf(y, z) → x	isPrerequisiteOf(x, y) → isPrerequisiteOf(y, z) → x
Universal constraints	UndergraduateStudent ≡∀studies.UndergraduateUnit, studies (x, y) → <x rdf:type UndergraduateStudent>	UndergraduateStudent ≡∀studies.UndergraduateSubject, studies (x, y) → <x rdf:type UndergraduateStudent>
existential constraints	Lecturer⊑∃isLecturerOf.Unit, isLecturerOf(x, y) → Lecturer(x) →x	Lecturer⊑∃isLecturerOf.Subject, isLecturerOf(x, y) → Lecturer(x) →x
cardinality constraints	Lecturer ⊒ isLecturerOf. ≥1 Unit, isLecturerOf(x, y) → Lecturer(x) →x	Lecturer ⊒ isLecturerOf. ≥1 Subject, isLecturerOf(x, y) → Lecturer(x) →x

To demonstrate the adaptability of the e-learning system's prototype we plug each ontology to each instance of the prototype. Then, we execute different types of queries to see whether we can execute all the queries on both instances of the prototype. We observe whether both instances generate similar answers with different terms, yet specific to each institution, to those queries.

7.9.1 Proof of Concept Prototype for Adaptive E-Learning System's Framework

The proof of concept prototype is used to provide an idea of the ontology-based adaptive e-learning system's framework and query-answering on an institution-specific ontology. The proof of concept prototype is also an implementation of the *ontology-based plug and play architecture* that has been given in Chapter 3, Figure 3.4. Hence, the proof of concept prototype of the adaptive e-learning system's framework (Figure 7.2) includes the three layers: an interface layer, a components layer and a repository layer.

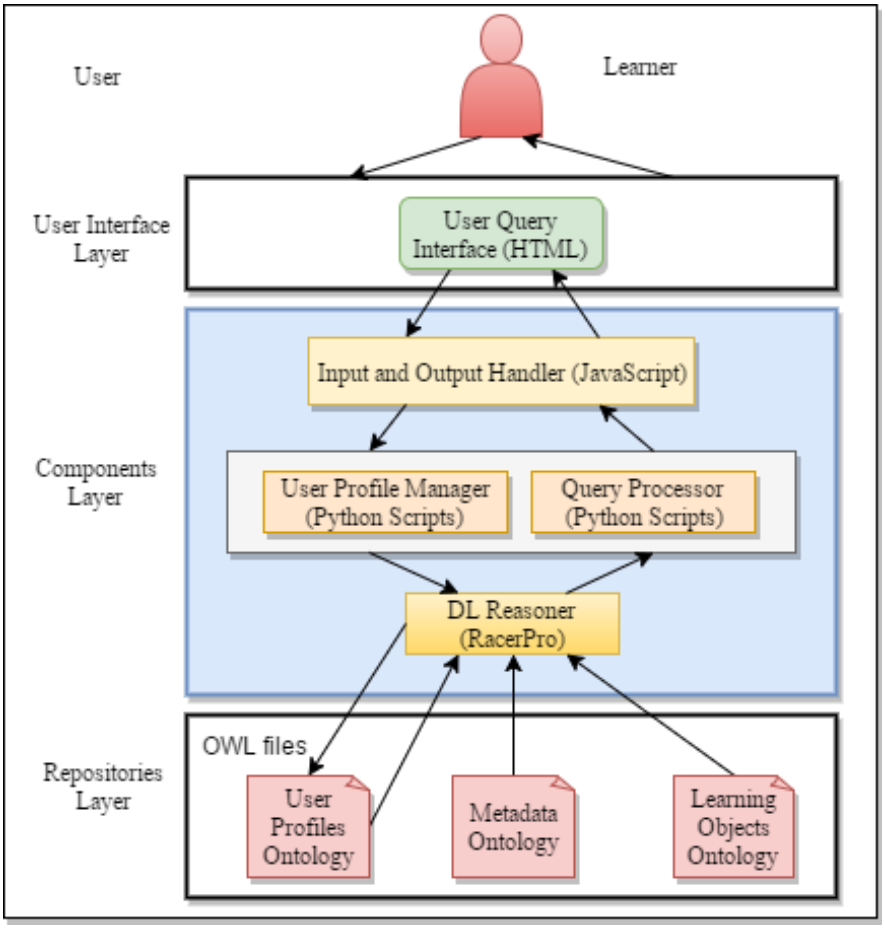


Figure 7.2: Adaptive e-learning system's framework

A set of technologies was used to build the proof of concept prototype. The interface layer is built using the server-side scripts written in Python programming language with embedded HTML. The query's inputs and outputs are handled by JavaScript. The components layer that performs main system functions is written in Python and the server-side scripts that perform these functions are executed on a Python web server. RacerPro is used as the DL reasoner. The populated OWL ontologies represent the repository layer. JASON notation is used to pass the information between the layers. The queries composed in the interface layer are converted into nRQL queries that are executed on the learning ontology by the DL reasoner.

7.9.2 Instances of Proof of Concept

As already mentioned, in this evaluation, we use two instances of the proof of concept prototype; one for MQ and the other for CSU. Each instance is plugged with an institution-specific learning ontology. When an instance of the prototype is executed, primarily its query interface becomes visible to the user. Then the user can use the query interface to formulate a query that is eventually executed on the learning ontology. In the following subsections we

elaborate on formulating and executing three types of queries on each learning ontology: 1) simple queries, 2) conjunctive queries, and, 3) negative queries.
This query interface includes several interface components that allow the learner to select the concepts, roles and constraints that are required to form a query. The query interface is populated with the initial instances of the concepts and roles at the time the system is loaded. When a learner logs into the system it gets refreshed with the instances specific to the learner.

The query's interface allows the user to enter a query as a set of query atoms that consist of a subject, role and an object. The query's interface allows a learner to navigate through the ontology and to select the relevant ontology elements to compose a new query.

In this query interface, to compose the first query atom, the learner can select a concept of what he/she wants to query as the first input. This results in displaying the roles that are associated with that concept in the drop down list next to it. When the learner selects a specific role, the associated objects are visible in the drop down list next to that role. This process is continued to add more query atoms and to complete the query.

The query interface includes the relevant interface components to deal with complex queries with conjunction, disjunction and negation as well. In the background, the queries are composed according to the selected constructors on the query interface. In the next section we introduce the use of the query interface to compose a query.

7.9.3 Query Formulation in the Query Interface

The user starts to use the interface of the proof of concept prototype to find the required learning objects that correspond to his profile. Let us assume that the user intends to pose the question in (70) to the learning system:

Which learning resources of ISYS114 on 'understanding of the requirements gathering process' are taught by the MQLecturer2? (70)

The user navigates the ontology on the interface level and the query handler generates a series of nRQL queries in the background (Haarslev et al., 2012). The DL reasoner answers these queries, and the interface is updated after each user interaction using standard AJAX technology.

In our case, the interface is populated with a refined list of domain concepts by executing the query in (71) that returns the taxonomy of the TBox (Haarslev et al., 2012). The user selects a domain concept, in our case *learning resource*, from the concept list.

(taxonomy) (71)

This selection triggers new nRQL queries and the DL reasoner returns a number of associated concept roles as the answer. In our case, the user selects the role *resource of*, that is, the inverse role of *hasResource*, and the query handler generates the following two queries (72) and (73) in a sequence (*role-range* is a role query in nRQL [Haarslev et al., 2012]):

(role−range <role name>) (72)

(retrieve (?x) (and (?x <concept name>))) (73)

Those queries are eventually passed to the DL reasoner. The DL reasoner returns the concept *Unit* as answer to the first query. The second query returns the instances of the concept *Unit* as answer. These answers are eventually displayed in a drop-down menu (Figure 7.3). From this drop-down menu the user selects the unit, *ISYS114* as the object of the role *resource of*. The nRQL query atoms generated in this case are given in (74):

(?x LearningResource) (?x ISYS114 resourceOf) (74)

The interface allows the user to extend a question by adding additional query atoms. In our case, the user selects the roles *taught by* and *expects learning outcome* that displays a list of lecturers and learning outcomes of *ISYS114* as a result. The user selects the desired lecturer *MQ Lecturer 2* and the learning outcome *understanding of the requirements gathering process*. This generates query atoms in nRQL as in (75) and (76):

(?x MQLecturer2 taughtBy) (75)

(?x understandingOfTheRequirementsGatheringProcess expectsLearningOutcome) (76)

Once the user has completed the entire query, he/she submits the query to the system. The query handler completes the query by inserting additional predicates that are related to the LOM attributes of learning resources based on user preferences. In this process, the query handler matches the LOM attributes with the user preferences. For example, let us assume that the current user prefers the learning resources with a difficulty level *easy*, content type *lecture*, format *wmv* and the content level *level100*. This generates the nRQL query as in (77):

(retrieve (?x) (and (?x LearningResource)

(?x ISYS114 resourceOf) (?x MQLecturer2 taughtBy)

(?x understandingOfTheRequirementsGatheringProcess
expectsLearningOutcome) (77)

(?x easy hasDifficulty) (?x lecture hasContentType)

(?x wmv hasFormat) (?x level100 hasLevel)))

After submission of this query, the learning resources that satisfy all the above criteria are retrieved and returned. The final state of the GUI is shown in Figure 7.3. The preferences attributes are not shown in the GUI since they are obtained from the user profile. To obtain the learning resources, the user would need to click on any item in the result set, in this case, *ISYS114 lecture slides 1*.

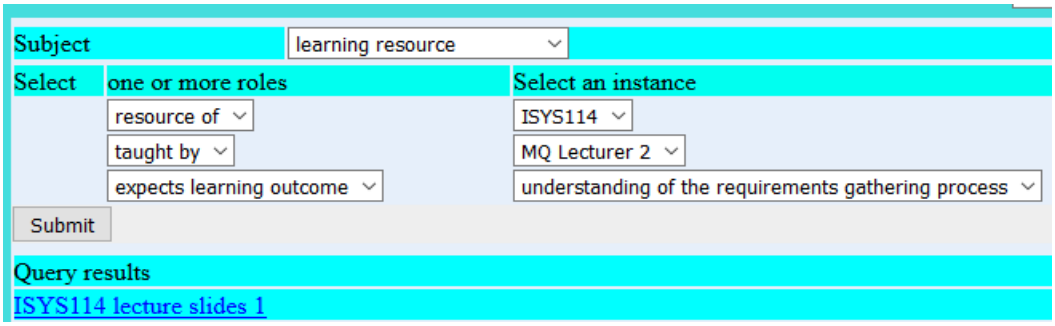


Figure 7.3: Composing query atoms in the query interface

The proposed proof of concept query interface is used to formulate different types of queries. In the following section we demonstrate how different types of queries are formulated and executed on the two sample ontologies to generate similar answers yet with different terms.

7.9.4 Simple Queries

A simple query involves a single relation between two objects or a relation between an object and its instances. The simple queries that we can build at MQ and CSU depend on the terms, *instances* and/or *relations* that are involved. For example, suppose that a student at MQ asks: 'What iLectures are available?' For that, the student selects the term *ilecture* on the interface and receives a list of available iLecture instances (Figure 7.4). Having obtained the instances, let us assume the student wants now to know 'Who teaches a particular ISYS114 instance of iLectures?' For this purpose, he/she selects the relevant instance (*ISYS114ilecture10*) and the

relation (*taughtBy*) on the interface and receives the name of the lecturer (*MQLecturer1*) as a result:

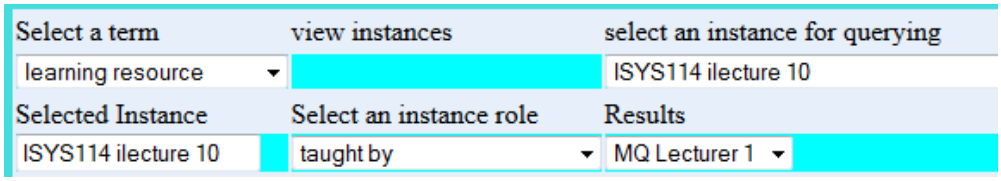


Figure 7.4: A simple DL query on the MQ ontology

In order to answer these simple queries, the e-learning system uses the support of the *DL Query Processor*. The *DL Query Processor* translates each user query into a formal query as in (78) and (79) that are passed to the DL reasoner:

(retrieve (?x) (and (?x ilecture)) (78)

(retrieve (?x) (and (ISYS114ilecture10 ?x taughtBy))) (79)

When we consider a student at CSU asking a similar query about videos (Figure 7.5), he/she would first retrieve the available instances using the term *video* on the interface, and then select an instance (*ITC114Wk10Audio2*) and a suitable role (*deliveredBy*) to find the name of the lecturer (*CSULecturer1*).

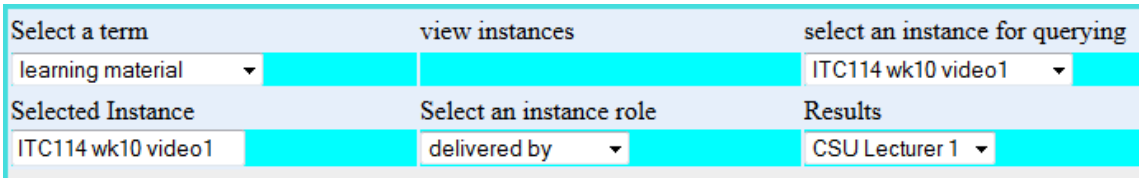


Figure 7.5: A simple DL query on the CSU ontology

The DL queries created by the *DL Query Processor* for this input are as in (80) and (81):

(retrieve (?x) (and (?x video))) (80)

(retrieve (?x) (and (ITC114Wk10Video1 ?x deliveredBy))) (81)

7.9.5 Conjunctive Queries

In our system, the user can compose conjunctive queries not only on the learning resources, but also on the LOM that are the characteristics of the learning resources and are related to user preferences. Suppose that a student at MQ wants learning resources used by a particular lecturer which are related to a particular learning outcome. Also suppose that the student wants to specify further LOM attributes that these learning resources need to satisfy. For

example: ‘Which learning resources of ISYS114 that is taught by MQ Lecturer 2 fall under a particular learning outcome and are available in pdf format, have the content type presentation and are easy to understand?’ The final state of the GUI is shown in Figure 7.6. To obtain the lecture slides, the user would need to click on any item in the result set.

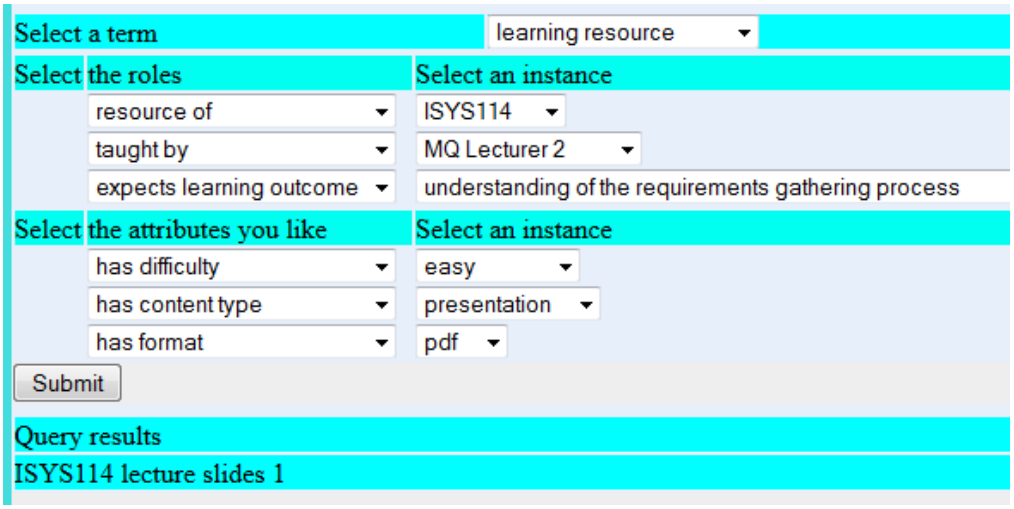


Figure 7.6: A conjunctive query with LOM attributes on the MQ ontology

The following (82) is the final query that is generated by the DL Query Processor and executed by the DL reasoner to generate the required results:

(retrieve (?x) (and (?x ISYS114 resourceOf) (?x MQLecturer2 taughtBy) (?x understandingOfTheRequirementsGatheringProcess expectsLearningOutcome) (?x easy hasDifficulty) (?x presentation hasContentType) (?x pdf hasFormat))) (82)

An analogous query can be asked by a student at CSU:

‘Which learning materials of ITC114 are delivered by CSU Lecturer 2 that cover a particular learning objective and are available in pdf format, have the content type presentation and are easy to understand?’ Again, the final state of the GUI is shown in Figure 7.7.

These two conjunctive queries use the same LOM; but they use different concepts and relations. Hence, the formal query for CSU (83) looks different to the MQ one:

(retrieve (?x) (and (?x ITC114 materialOf) (?x CSULecturer2 deliveredBy) (?x toDescribeIssuesOfDatabaseAdministration coversObjective) (?x easy hasDifficulty) (?x presentation hasContentType) (?x pdf hasFormat))) (83)

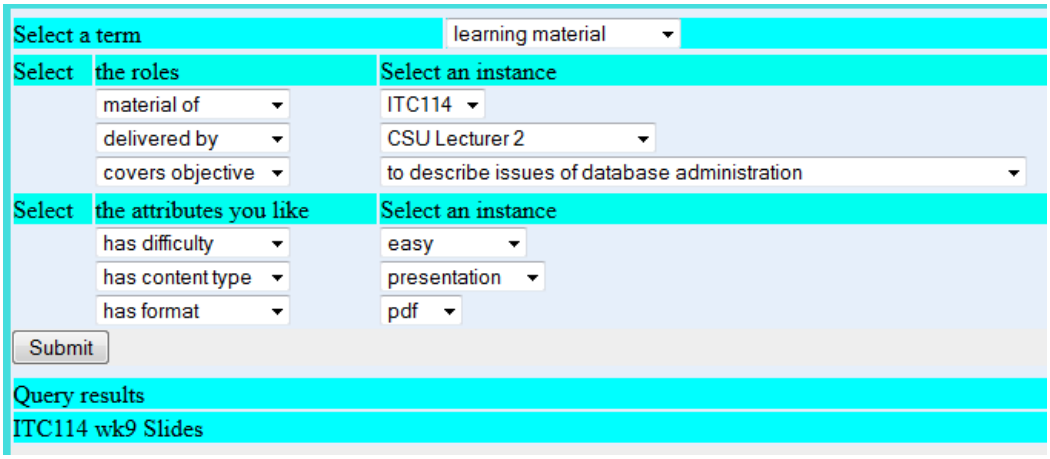


Figure 7.7: A conjunctive query with LOM attributes on the CSU ontology

7.9.6 Inclusion Queries

Recall that the term *learning resources* includes reading material which in turn includes *lecture note*, *lecture slides* and *textbook*. While forming a simple query at MQ, if a student selects the term *lecture slides*, then the system lists only lecture slides, whereas if he selects the term *learning resources*, then the system lists all types of learning resources including lecture slides by following inclusion relations. The two DL queries generated by the system are given in (84) and (85):

$$(retrieve\ (?x)\ (and\ (?x\ lectureSlides)) \tag{84}$$

$$(retrieve\ (?x)\ (and\ (?x\ learningResources)) \tag{85}$$

If a student at CSU forms a simple query by selecting the term *lecture slides*, then the system lists only lecture slides, whereas if he selects the term *learning material*, then the system lists all types of learning materials including lecture slides. Again, the two DL queries generated by the system are given in (86) and (87):

$$(retrieve\ (?x)\ (and\ (?x\ lectureSlides)) \tag{86}$$

$$(retrieve\ (?x)\ (and\ (?x\ learningMaterial)) \tag{87}$$

Besides that, we have defined meronymic relations *partOf* and *hasPart* as transitive relations in the ontology. For example, in the CSU ontology *schedule* is a *partOf* the *subjectOutline* (88), and *topic* is a *partOf* the schedule (89).

$$(schedule\ partOf\ subjectOutline\ ) \tag{88}$$

(topic partOf schedule) (89)

This makes it possible to retrieve topics listed in a subject outline by taking the transitive closure of the *partOf* relation.

7.9.7 Negative Queries

Finally, we consider queries that include negation. For example, a student at MQ wants 'all learning resources of ISYS114 that are taught by MQ Lecturer 2 and have a particular learning outcome, but do not include lecture slides'. In order to compose this query, a student has to select the term *learning resource* and select the check box for negation (not) to exclude lecture slides that are a subclass of learning resource (Figure 7.8).

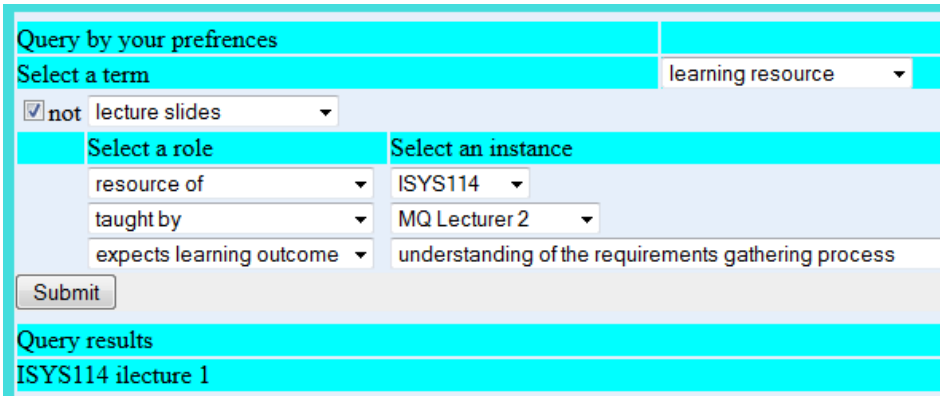


Figure 7.8: A negative query results on the MQ ontology

Following DL query in (90) is executed on the MQ ontology:

(retrieve (?x) (and (?x learningResource) (neg (?x lectureSlides))
(?x ISYS114 resourceOf) (?x MQLecturer2 taughtBy)
(?x understandingOfTheRequirementsGatheringProcess
expectsLearningOutcome))) (90)

The query's result(s), *ISYS114 ilecture1*, is displayed to the user.

Similarly, a student at CSU who is looking for: 'all the learning materials of ITC114 that are taught by CSU Lecturer 2 and covers a particular objective but that does not include lecture slides' composes and submits the following query (Figure 7.9):

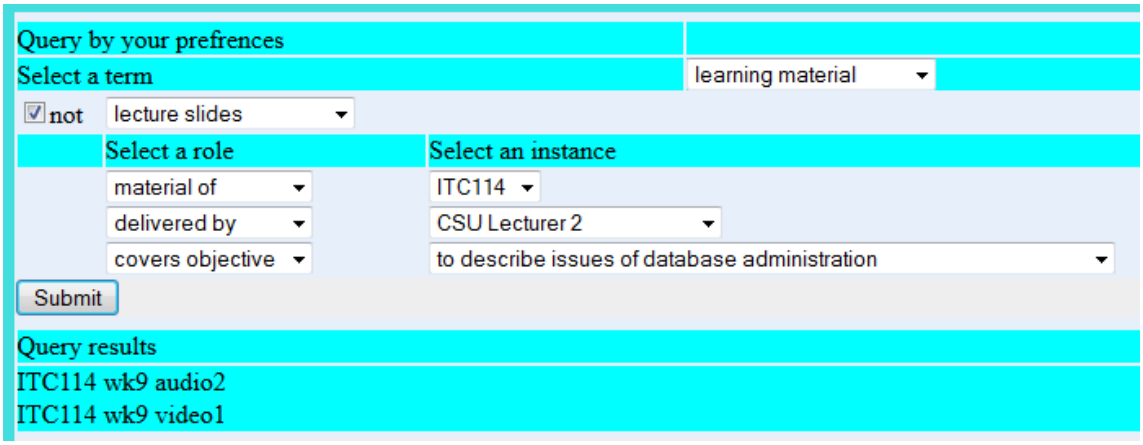


Figure 7.9: A negative query results on the CSU ontology

This generates the following DL query in (91) that is executed on the CSU ontology:

(retrieve (?x) (and (?x learningMaterial) (neg (?x lectureSlides))

(?x ITC114 materialOf) (?x CSULecturer2 deliveredBy) (91)

(?x toDescribeIssuesOfDatabaseAdministration coversObjective)))

The query result(s), *ITC114 wk9 video1* and *ITC114 wk9 audio2* are displayed to the user.

7.10 Discussion and Conclusion

The ontology evaluation approaches that have been proposed in the literature focus on directly inspecting an ontology and checking what concepts and properties have been used in an ontology and what have not been used. In some approaches, an ontology is verified against a 'Golden Standard' that is an ontology language or another ontology. In some approaches the ontologies are ranked based on the characteristics they satisfy. However, we have observed that these approaches do not focus on evaluating inference capabilities of the ontologies based on the ontology language constructors used in an ontology.

Our analysis of the current ontology benchmarks shows that the current benchmarks provide sound benefits and but also present numerous problems. That makes them not directly applicable to evaluating the query-answering capabilities of learning ontologies. That is why we need an enhanced query suite that is specific to the OWL 2 Learn profile.

In this chapter, we have discussed the possibility of using a query suite to evaluate the query-answering and inference capabilities of learning ontologies that belong to the OWL 2 Learn profile. As we evaluate query results, our approach could belong to category 1 of the categorisation of the evaluation methods given in (Brank et al., 2005). Again, as we evaluate

the inferences and constructors involved in each query, our approach could also belong to category 2 of the above categorisation as well. In our approach, we indirectly evaluate the elements of an ontology by means of a query suite that tests the query-answering and inference capabilities of an ontology.

The current query suites proposed in the popular ontology benchmarks include heterogeneous queries written in the SPARQL query language. Even though the query suite of the UOBM benchmark is a close match to the OWL 2 Learn profile, it is not directly applicable for several main reasons. Firstly, the UOBM query suite includes queries that involve OWL2 constructors that are not a part of the OWL 2 Learn profile. Secondly, some of the OWL 2 Learn constructors have not been used in the UOBM query suite. Thirdly, existing query suites including UOBM focus more on benchmarking the scalability of a KBS, but not the specific inference capabilities of learning ontologies. Finally, the UOBM benchmark queries have been written for two specific ontologies.

For these reasons we cannot directly apply the query suite of UOBM and identify a query suite to evaluate leaning ontologies by customising the UOBM query suite. To achieve this goal, we have designed ten analogous queries, excluded five UOBM benchmark queries and also introduced three new queries. The five queries we excluded contain OWL 2 constructors that are not a part of the OWL 2 Learn profile. The three new queries that are introduced evaluate the inferences associated with several OWL 2 constructors: data property assertion, quantification and disjoint data properties. Those constructors are not involved in answering the original UOBM benchmark queries. However, those constructors are a part of the OWL 2 Learn profile. The new query suite we propose is comprehensive enough to evaluate learning ontologies.

As we discussed in Section 7.1, the current ontology evaluation techniques found in the literature do not evaluate the inference capabilities of ontologies. Hence, in this thesis we have proposed a technique to evaluate inference capabilities of learning ontologies. The technique is based on the UOBM benchmark query suite that involves inference capabilities of learning ontologies. We adopt the UOBM benchmark query suite and enhance with three additional queries as a new ontology evaluation technique specially for evaluating inference capabilities of an e-learning ontology.

We see that the query suite that we derived is capable of evaluating different inference features of the sample learning ontologies and determining what different OWL 2 Learn constructors are involved in them.

This helps us to determine whether a learning ontology has full inference capabilities of OWL 2 Learn profile or not. This query suite could probably be revised if someone needs to evaluate an OWL 2 ontology with a higher DL expressivity than the OWL 2 Learn profile. In such a revision to the query suite, the new query suite will have to be extended by including the anonymous queries to the excluded UOBM queries 7, 10, 11, 14 and 15. However, it is only a minority of the learning ontologies with such a very high DL expressivity.

In our evaluation, we used two learning ontologies that we developed. However, this evaluation could be extended to other learning ontologies by writing analogous queries for them. When we analysed the OWL 2 constructors, and the associated inferences on the sample learning ontologies, we observe similar structures and inferences between them. This suggest that we could develop a standard domain ontology for the learning domain and use it as a base ontology or a meta-ontology to develop learning ontologies for other institutions. That has the advantage of replicating the same domain ontology with institution-specific terminology for each institution-specific ontology (application ontology). Each institution or application specific learning ontology is pluggable to an instance of the adaptive e-learning system that is based on the plug and play architecture. This helps us to overcome the duplicated effort in developing a learning ontology for each institution.

On the other hand, in our evaluation of learning ontologies, we used Protégé as the main tool in the evaluation environment to execute the queries. However, an ideal user interface (for the users) should allow them to enter their queries in a natural language as they would in the Google search engine. Therefore, a text-based query interface could be added to an e-learning systems prototype to allow the user to enter the queries on the learning objects. If such a query interface is used, an additional module (a *keywords extractor*) would have to be introduced to the e-learning systems architecture. That new module would be responsible for identifying the key words from the text entered by the user. Another requirement is to identify the elements of the ontology—concepts, object properties, data properties, constraints—that match the key words in the text. Then, it would compose triples from the keywords that would make sense to the ontology and pass the keywords to the *Input/Output Handler* of the *plug and play architecture*.

The evaluation that we conducted on the adaptivity of the ontology-based e-learning system's framework proves that instances of the proof of concept prototype are possible to be easily adapted at different institutions for query-answering. The evaluation also shows that this architecture allows each institution to keep its own terminology.

This evaluation proves that we have achieved the goals of our research:

1) We are able to reduce the effort in developing e-learning systems by using the *ontology-based plug and play architecture*.
2) We can alleviate the duplicated effort to identify the ontology language constructors required to develop a learning ontology using the OWL 2 Learn profile.
3) We can alleviate the duplicated effort in developing learning ontologies by using a base learning ontology.
4) We can evaluate the query-answering and inference capabilities of learning ontologies using the benchmark query suite for evaluating OWL 2 Learn ontologies.
5) The proof of concept prototype shows that the e-learning system's framework is adaptable at different learning institutions with an institution-specific ontology.

Chapter 8: Conclusions

The purpose of this thesis is to answer the research question of 'how we enhance the development of e-learning systems by using Semantic Web technologies'. In answering this research question, we formulated two subquestions. 1. How to use SW technologies to overcome duplicated effort in developing e-learning systems. 2. How to use SW technologies to segregate, represent and reuse the domain knowledge of a learning domain in e-learning systems. This thesis also answers the four research problems listed in Subsection 2.2.2 of the literature review chapter.

8.1 Main Contributions

As a solution to the research question(s), we developed an architecture named *ontology-based plug and play architecture* for an adaptive e-learning system's framework. Instances of this framework can possibly be deployed at different institutions. This architecture includes three layers of components: a user interface layer; a components layer; and, a repositories layer. The majority of the elements in all the layers of this architecture could be deployed at different institutions with minor changes. According to this architecture, institution-specific domain knowledge is captured in a learning ontology. When this institution-specific ontology is plugged into an instance of the e-learning system it allows learners to query the domain knowledge on learning resources specific to that institution.

However, deploying an instance of the adaptive e-learning system demands developing a learning ontology specific to an institution that duplicates the effort in ontology development. The OWL 2 Learn profile is a sublanguage of OWL 2, that we propose in this work assists alleviating the above problem to some extent. The OWL 2 Learn profile includes OWL 2 constructors common to the majority of the e-learning ontologies of the corpus we analysed. Hence, the OWL 2 Learn profile can be used as a guide for developing e-learning ontologies.

In addition, this thesis elaborates on how an ontology-based e-learning system works with legacy databases. After the development of a learning ontology it is populated with the data from legacy databases using a legacy database to ontology mapping tool that is currently available in the public domain. The e-learning system indirectly accesses the data in the

legacy database through the learning ontology and the learning ontology is updated by using the mapping tool.

Another contribution in this thesis is a benchmark query suite that is derived from a query suite of the UOBM benchmark; a current ontology benchmark. This query suite helps us to evaluate the query-answering and inference capabilities of the e-learning ontologies within the OWL 2 Learn profile. This query suite includes SPARQL queries that invoke inferences on a learning ontology associated with different constructors of the OWL 2 Learn profile.

8.2 Limitations and Suggestions for Improvement

We observe that the OWL 2 Learn profile has the DL expressivity of *SHIQ(D)* that is sufficient to represent the majority of the learning ontologies. OWL 2 Learn profile excludes some of the OWL 2 constructors such as nominal (O) and specific object property attributes such as equality or inequality of the OWL 2 language. These restrictions could affect the query-answering and inference capabilities of a minority of learning ontologies we have analysed that would demand a higher DL expressivity. However, as a way out of that limitation, we suggest representing those excluded OWL 2 constructors using other OWL 2 constructors as explained in Chapter 4.

The *plug and play architecture* we propose includes three layers of components to perform tasks related to retrieving domain knowledge. However, this architecture could be extended to include other tasks such as student administration, course and learning management of an institution, by adding other components. That would help to build an integrated and institution-specific comprehensive system with a comprehensive institutional ontology that represents the knowledge of an institution to support its multiple tasks.

The benchmark query suite we suggest includes only 13 queries. These queries evaluate inferences involved in all the OWL 2 Learn constructors. The inferences involved in these queries are only a subset of possible inferences on a learning ontology. However, patterns of inferences could generate the same results or different results. Even the different learning ontologies could use different combinations of OWL 2 Learn constructors in different ways to specify a domain. Therefore, it is possible to extend this query suite or study about the patterns of inferences in query-answering on learning ontologies.

In evaluating each institution-specific ontology, the query suite has to be rewritten to obtain a query suite that is analogous to the MQ query suite. That makes the query suite matching an

institution-specific ontology and executable on a specific ontology. This process of rewriting the queries could be automated to generate matching queries for each learning ontology.

In this work, benchmark query suites enable an evaluation of institutional level adaptation of the proposed architecture. This evaluation could be extended to the user level of the systems architecture by an empirical survey or an experiment involving users. That could also help to evaluate the e-learning system on learners transiting between two institutions where the proposed architecture is deployed to substantiate further outcomes.

8.3 Recommendations for Further Work

In this thesis, an attempt has been made to enhance the development of an e-learning system using SW technologies. However, we cannot say that it overcomes all the possible problems. It is possible to have more problems and solving them could further improve an e-learning system. Below, we discuss some recommendations and further work related to our work.

We suggest developing a base ontology or a meta-ontology for the learning domain based on the OWL 2 Learn ontologies. Then, the base ontology could be used as a model or an instance in developing other institution-specific ontologies. This would help the ontology engineers to minimise the effort needed to develop a learning ontology for each institution.

The proposed learning ontologies in this work are hand-coded by specialised engineers who evaluate the similarity between terms such as *Unit* and *Subject*. Text mining techniques could be used to discover such similarities by looking at contents' relationships. So, semantic analytics using text-mining techniques could be used to alleviate the manual involvement of ontology engineers and streamline the evaluation of learning ontologies.

The current version of the adaptive learning system's prototype uses a graphical user interface (GUI) as the query interface. In that query interface, users compose a query as a set of triples on the concept being queried. That requires a learner to have some understanding of using the GUI to compose a query. However, it would be easy to enter a textual query in natural language instead of composing a query on a GUI.

A text-based query interface could be developed for writing and submitting a query. This requires a natural language processor to convert a natural language query into a SPARQL query before it is executed on a specific learning ontology. Using natural language, the same idea can be expressed in multiple ways. For this reason, converting a textual query into a

query in a specific query language is a challenging task. Therefore, a subset of natural language could be used to write text-based queries.

8.4 Remarks

In this thesis, we have shown the feasibility of using instances of an e-learning system built according to the *plug and play architecture* in multiple institutions. This reuse of the e-learning system is achieved by building the domain knowledge as a learning ontology that is pluggable to an e-learning system. We also showed how we can alleviate the effort in developing a learning ontology that is based on the OWL 2 Learn profile.

We also suggest that the proposed approach to the solution of the research problem discussed in this thesis could be applied not only in the learning domain, but also in other application domains.

References

Al-Yahya, M., George, R., & Alfaries, A. (2015). Ontologies in E-learning: review of the literature. *International Journal of Software Engineering and Its Applications, 9(2)*, 67-84.

Amith, M. F., He, Z. B., Lossio-Ventura, J. A., & Tao, C. (2018). Assessing the Practice of Biomedical Ontology Evaluation: Gaps and Opportunities. *Journal of biomedical informatics*.

Amorim, R. R., Lama, M., Sánchez, E., Riera, A., & Vila, X. A. (2006). A Learning Design Ontology based on the IMS Specification. *Educational Technology & Society*, 38-57.

Arenas, M., Bertails, A., Prud, E., & Sequeda, J. (2012). *A direct mapping of relational data to RDF*. W3C.

Arenas, M., Gottlob, G., & Pieris, A. (2014). Expressive languages for querying the semantic web. *In Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 14-26). ACM.

Aroyo, L., & Dicheva, D. (2004). The New Challenges for E-learning: The Educational Semantic Web. *Journal of Educational Technology & Society*, 59-69.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., & Harris, M. A. (2000). Gene Ontology: tool for the unification of biology. *Nature genetics, 25(1)*, 25-29.

Astrova, I. (2009). Rules for mapping SQL relational databases to OWL ontologies. *Metadata and Semantics*, 415-424.

Astrova, I., & Kalja, A. (2008). Automatic Transformation of SQL Relational Databases to OWL Ontologies. *WEBIST (2)*, 131-136.

Azzedin, F., & Maheswaran, M. (2002). Evolving and managing trust in grid computing systems. *Canadian Conference on Electrical and Computer Engineering, CCECE 2002.* (pp. 1424-1429). IEEE.

Baader, F., & Lutz, C. T. (2010). Small is Again Beautiful in Description Logics. *KI – Künstliche Intelligenz, vol 24(1)*, 25–33 .

Baader, F., & Nutt, W. (2003). Basic Description Logics. In F. Baader, *The description logic handbook: theory, implementation and applications* (pp. 43-95). Cambridge: Cambridge University Press.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2003). *The description logic handbook: Theory, implementation and applications.* Cambridge: Cambridge University Press.

Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., & Xiao, G. (2014). The ontop framework for ontology based data access. *In Chinese Semantic Web and Web Science Conference* (pp. 67-77). Berlin, Heidelberg: Springer.

Bajenaru, L., & Smeureanu, I. (2015). An Ontology-based Approach for Modelling E-Learning in Healthcare Human Resources Management. *Economic Computation & Economic Cybernetics Studies & Research*.

Baker, T. (2000). A grammar of Dublin Core. *D-lib magazine, 6(10)*, 47-60.

Bandeira, J., Bittencourt, I. I., Espinheira, P., & Isotani, S. (2016). FOCA: A Methodology for Ontology Evaluation. *arXiv preprint arXiv:1612.03353*.

Barrasa, J., Corcho, O., Shen, G., & Gomez-Perez, A. (2004). R2O, an Extensible and Semantically Based Database-to-ontology Mapping Language. *2nd Workshop on Semantic Web and Databases* .

Berners-Lee, T., & Fischetti, M. (2001). *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor.* DIANE Publishing Company.

Bilgin, G., Dikmen, I., & Birgonul, M. T. (2014). Ontology evaluation: An example of delay analysis. *Procedia Engineering, 85*, 61-68.

Bizer, C., & Schultz, A. (2009). The berlin sparql benchmark . *International Journal on Semantic Web and Information Systems (IJSWIS), 5(2)*, 1-24.

Bock, J., Haase, P., Ji, Q., & Volz, R. (2008). Benchmarking OWL reasoners. *In ARea2008-Workshop on Advancing Reasoning on the Web: Scalability and Commonsense.* Tenerife.

Brank, J., Grobelnik, M., & Mladenić, D. (2005). A survey of ontology evaluation techniques.

Bumans, G. (2010). Mapping between Relational Databases and OWL Ontologies: an Example. *Computer Science and Information Technologies, Vol. 756, Scientific papers, University of Latvia*, 99-117.

Calvanese, D., Cogrel, B., Kalayci, E., Komla-Ebri, S., Kontchakov, R., Lanti, D., . . . Xiao, G. (2015). OBDA with the Ontop Framework. *In SEBD*, (pp. 296-303).

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., & Xiao, G. (2017). Ontop: Answering SPARQL queries over relational databases. *Semantic Web, 8(3)*, 471-487.

Carroll, J., Herman, I., & Patel-Schneider, P. (2012). *OWL 2 Web ontology language RDF-based semantics (Second Edition).* W3C.

Chen, C. M., Lee, H. M., & Chen, Y. H. (2005). Personalized e-learning system using item response theory. *Computers & Education*, 44(3), 237-255.

Chrysafiadi, K., & Virvou, M. (2015). Advances in personalized web-based education. *Intelligent Systems Reference Library 78, Springer*, 1-114.

Chung, H., & Kim, J. (2012). Learning Ontology Design for Supporting Adaptive Learning in e-Learning Environment. *2012 International Conference on Information and Computer Networks (ICICN 2012).* Singapore: IACSIT Press, Singapore.

Cohen, E. B., & Nycz, M. (2006). Learning Objects and E-Learning an Informing Science Perspective. *Interdisciplinary Journal of Knowledge and Learning Objects*, 23-34.

Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data & knowledge engineering, 46(1)*, 41-64.

Cortés-Calabuig, A., Denecker, M., Arieli, O., Van Nuffelen, B., & Bruynooghe, M. (2005). On the local closed-world assumption of data-sources. *In International Conference on Logic Programming and Nonmonotonic Reasoning* (pp. 145-157). Springer.

Cranefield, S., Hausteiny, S., & Purvis, M. (2001). UML-Based Ontology Modelling for Software Agents.

Cullot, N., Ghawi, R., & Yétongnon, K. (2007). DB2OWL: A Tool for Automatic Database-to-Ontology Mapping . *SEBD* , (pp. 491-494).

Cyganiak, R., Wood, D., Lanthaler, M., Klyne, G., Carroll, J. J., & McBride, B. (2014). *RDF 1.1 concepts and abstract syntax.* W3C recommendation, 25(02).

Dall'Alba, G., & Barnacle, R. (2007). An ontological turn for higher education. *Studies in Higher Education*, 679-691.

Das, S., Sundara, S., & Cyganiak, R. (2012). *R2RML: RDB to RDF Mapping Language.* W3C.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. *Principles of knowledge representation, 1*, 191-236.

Duque-Ramos, A., Fernández-Breis, J., Iniesta, M., Dumontier, M., Aranguren, M., Schulz, S., . . . Stevens, R. (2013). Evaluation of the OQuaRE framework for ontology quality. *Expert Systems with Applications, vol. 40*, 2696-2703.

Dutta, B., Maddali, D. P., & Prasad, A. R. (2009). Ontology supported personalized e-learning Repositories. *2nd International Conference on the Semantic Web and Digital Libraries (ICSD).* Trento.

Fahad, M., & Qadir, M. A. (2008). A Framework for Ontology Evaluation. *ICCS Supplement, 354*, 149-158.

Felder, R. M., & Silverman, L. K. (1998). Learning and teaching styles in engineering education. *Engineering education*, 674-681.

Fensel, D. (2000). Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce.

Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE intelligent systems, 16(2)*, 38–45.

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). *Methontology: from ontological art towards ontological engineering.*

Fox, M. S., & Grüninger, M. (1997). Ontologies for enterprise modelling. *Enterprise Engineering and Integration* (pp. 190-200). Berlin, Heidelberg: Springer.

Gaeta, M., Orciuoli, F., & Ritrovato, P. (2009). Advanced ontology management system for personalised e-Learning. *Knowledge-Based Systems*, 292-301.

Gangemi, A., Catenacci, C., Ciaramita, M., & Lehmann, J. (2005). A theoretical framework for ontology evaluation and validation. *In SWAP (Vol. 166)*, 16.

Gascueña, J., Fernández-Caballero, A., & González, P. (2006). Domain Ontology for Personalized E-Learning in Educational Systems. *Sixth International Conference on Advanced Learning Technologies* (pp. 456-458). IEEE.

Genesereth, M. R., & Fikes, R. E. (1992). *Knowledge interchange format-version 3.0: reference manual.* Knowledge Systems Laboratory, Stanford University.

Ghawi, R., & Cullot, N. (2007). Database-to-Ontology Mapping Generation for Semantic Interoperability. *VLDB '07.* Vienna, Austria: ACM 978-1-59593-649-3/07/09.

Glimm, B., Ogbuji, C., Hawke, S., Herman, I., Parsia, B., Polleres, A., & Seaborne, A. (2013). *SPARQL 1.1 entailment regimes.* . W3C Recommendation 21 March 2013.

Golbeck, J., Grove, M., Parsia, B., Kalyanpur, A., & Hendler, J. (2002). New Tools for the Semantic Web. *EKAW 2002, LNCS 2473* (pp. 392–400). Springer.

Golbreich, C., Wallace, E. K., & Patel-Schneider, P. F. (2009). *OWL 2 Web Ontology Language new features and rationale. W3C working draft.* W3C (June 2009) http://www. w3. org/TR/2009/WD-owl2-new-features-20090611.

Grau, C., Motik, B., Stoilos, G., & Horrocks, I. (2012). Completeness guarantees for incomplete ontology reasoners: Theory and practice. *Journal of Artificial Intelligence Research, 43*, 419-476.

Gruber, T. R. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, 907-928.

Grüninger, M., & Fox, M. S. (1995). The role of competency questions in enterprise engineering. *In Benchmarking—Theory and practice. Springer US.*, 22-31.

Guangzuo, C., Fei, C., Hu, C., & Shufang, L. (2004). OntoEdu: a case study of ontology-based education grid system for e-learning. *In GCCCE2004 International conference*, (pp. 1-9). Hong Kong.

Guo, Y., Pan, Z., & Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web, 3(2)*, 158-182.

Gupta, S., Szekely, P., Knoblock, C. A., Goel, A., Taheriyan, M., & Muslea, M. (2012). Karma: A system for mapping structured sources into the Semantic Web. *In Extended Semantic Web Conference* (pp. 430-434). Springer, Berlin, Heidelberg.

Haarslev, V., Hidde, K., Möller, R., & Wessel, M. (2012). The RacerPro knowledge representation and reasoning system. *Semantic Web, 3(3)*, 267-277.

Handschuh, S., & Staab, S. (2002). Authoring and Annotation of Web Pages in CREAM. *The 11th International World Wide Web Conference, WWW 2002* (pp. 462–473). Honolulu, Hawaii: ACM Press.

Handschuh, S., Staab, S., & Volz, R. (2003). On Deep Annotation. *The 12th International World Wide Web Conference, WWW 2003* (pp. 431-438). ACM.

Harris, S., Seaborne, A., & Prud'hommeaux, E. (2013). *SPARQL 1.1 Query Language.* W3C recommendation, 21(10).

Hazber, M. A., Li, R., Gu, X., & Xu, G. (2016). Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology. *Appl. Math, 10(3)*, 1-21.

Henze, N., Dolog, P., & Nejdl, W. (2004). Reasoning and Ontologies for Personalized E-Learning in the Semantic Web. *Educational Technology & Society*, 82-97.

Hitzler, P., & Parsia, B. (2009). Ontologies and rules. In S. Staab, & R. Studer, *Handbook on Ontologies* (pp. 111-132). Berlin Heidelberg: Springer.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P., & Rudolph, S. (2012). *OWL 2 web ontology language primer. W3C recommendation.* W3C.

Hlomani, H., & Stacey, D. (2014). Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semantic Web Journal, 1-5*.

Holsapple, C. W., & Joshi, K. D. (2002). A collaborative approach to ontology design. *Communications of the ACM, 45(2)*, 42-47.

Horrocks, I., & Patel-Schneider, P. (2011). KR and reasoning on the semantic web: OWL. In J. Domingue, D. Fensel, & J. A. Hendler, *Handbook of Semantic Web Technologies* (pp. 365-398). Berlin Heidelberg: Springer.

Horrocks, I., & Sattler, U. (2001). Ontology reasoning in the SHOQ (D) description logic. *IJCAI Vol. 1, No. 3*, 199-204.

Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., & Motta, E. (2000). *The ontology inference layer OIL.*

Horrocks, I., Kutz, O., & Sattler, U. (2006). The even more irresistible SROIQ. *The 10th International Conference of Knowledge Representation and Reasoning*, (pp. 57-67).

Horrocks, I., Patel-Schneider, P. F., & Van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web, 1(1)*, 7-26.

Horrocks, I., Sattler, U., & Tobies, S. (1999). Practical reasoning for expressive description logics. *LPAR (Vol. 99)*, 161-180.

Hsu, I. C. (2012). Intelligent discovery for learning objects using semantic web technologies. *Journal of Educational Technology & Society, 15(1)*, 298.

Hustadt, U. M. (2005). Data complexity of reasoning in very expressive description logics. *In IJCAI (Vol. 5)*, 466-471.

IEEE. (2002). *Standard for Learning Objects Metadata. Learning Technology Standards Committee of the IEEE.* IEEE.

Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M. G., & Mora, J. (2015). BootOX: Practical mapping of RDBs to OWL 2. *In International Semantic Web Conference* (pp. 113-132). Springer, Cham.

Jones, D., Bench-Capon, T., & Visser, P. (1998). Methodologies for ontology development.

Kalogeraki, E. M., Troussas, C., Apostolou, D., Virvou, M., & Panayiotopoulos, T. (2016). Ontology-based model for Learning Object Metadata. *In Information, Intelligence, Systems & Applications (IISA).* IEEE.

Kasai, T., Yamagushi, H., Nagano, K., & Mizoguchi, R. (2005). A semantic web system for helping teachers plan lessons using ontology alignment. *12th International Conference on Artificial Intelligence in Education.* Amsterdam.

KBSI. (1994). *The IDEF5 Ontology Description Capture Method Overview.* Texas: KBSI Report .

Kepler, F., Paz-Trillo, C., Riani, J., Ribeiro, M., Delgado, K., Barros, L., & Wassermann, R. (2006). Classifying ontologies. *The 2nd Workshop on Ontologies and their Applications.* CEUR. http://ceur-ws.org/Vol-199/wonto-01.pdf.

Kharlamov, E., Brandt, S., Jimenez-Ruiz, E., Kotidis, Y., Lamparter, S., Mailis, T., & Zheleznyakov, D. (2016). Ontology-based integration of streaming and static relational data with optique. *In Proceedings of the 2016 International Conference on Management of Data* (pp. 2109-2112). AMC.

Khemaja, M., & Taamallah, A. (2016). Towards situation driven mobile tutoring system for learning languages and communication Skills: Application to Users with Specific Needs. *Educational Technology & Society*, 113–128.

Khondoker, M. R., & Mueller, P. (2010). Comparing ontology development tools based on an online survey. *World Congress on Engineering (WCE 2010)*, (pp. 188-192). London, UK.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM (JACM) , 42(4)*, 741-843.

Knoblock, C. A., Szekely, P., Ambite, J. L., Goel, A., Gupta, S., Lerman, K., & Mallick, P. (2012). Semi-automatically mapping structured sources into the semantic web. *In Extended Semantic Web Conference* (pp. 375-390). Berlin, Heidelberg: Springer.

Knoell, D., Atzmueller, M., Rieder, C., & Scherer, K. P. (2017). A Scalable Framework for Data-Driven Ontology Evaluation. *In WM*, (pp. 97-106).

Kolås, L. (2006). Topic maps in e-learning: An ontology ensuring an active student role as producer. *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, (pp. 2107-2113).

Kontopoulos, E., Vrakas, D., Kokkoras, F., Bassiliades, N., & Vlahavas, I. (2008). An ontology-based planning system for e-course generation. *Expert Systems with Applications*, 398-406.

Kougias, I., Seremeti, L., & Kalogeras, D. (2011). Ontology-Based Knowledge Management in NGEEs. *International Journal of Pure and Applied Sciences and Technology*, 54-62.

Krötzsch, M., Rudolph, S., & Hitzler, P. (2013). Complexities of Horn description logics. *ACM Transactions on Computational Logic (TOCL), 14(1)*, 2.

Lalingkar, A., Ramanathan, C., & Ramani, S. (2015). MONTO: A Machine-Readable Ontology for Teaching Word Problems in. *Educational Technology & Society*, 197–213.

Lambrix, P., Habbouche, M., & Perez, M. (2003). Evaluation of ontology development tools for bioinformatics. *Bioinformatics, 19(12)*, 1564-1571.

Ledvinka, M., & Křemen, P. (2015). Object-UOBM: an ontological benchmark for object-oriented access. *In International Conference on Knowledge Engineering and the Semantic Web* (pp. 132-146). Springer, Cham.

Lee, M., & Mizoguchi, R. (1998). Ontology Models for Supporting Exploratory Information Needs. *Advances in Knowledge Organization, 6*, 31-38.

Lembo, D., Mora, J., Rosati, R., Savo, D. F., & Thorstensen, E. (2015). Mapping analysis in ontology-based data access: Algorithms and complexity. *In International Semantic Web Conference, Springer, Cham.*, 217-234 .

Leukel, J., & Sugumaran, V. (2009). Towards a semiotic metrics suite for product ontology evaluation. *International Journal of Intelligent Information Technologies*, *5(4)*, 1-15.

Li, M., DU, Z., & Wang, S. (2005). Learning Ontology from Relational Databases. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, (pp. 3410-3415).

Li, Z., Raskin, V., & Ramani, K. (2008). Developing engineering ontology for information retrieval. *Journal of Computing and Information science in Engineering*, *8(1), 011003*.

Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., & Liu, S. (2006). Towards a complete OWL ontology benchmark . *Springer Berlin Heidelberg*, 125-139.

Magdalena, I. N. (2011). The use of distributed databases in e-learning systems. *Procedia-Social and Behavioral Sciences*, *15*, 2673-2677.

Miranda, S., Orciuoli, F., & Sampson, D. G. (2016). A SKOS-based framework for Subject Ontologies to improve learning experiences. *Computers in Human Behavior*, 609-621.

Mogotlane, K. D., & Fonou-Dombeu, J. V. (2016). Automatic Conversion of Relational Databases into Ontologies: A Comparative Analysis of Protégé Plug-ins Performances. *International Journal of Web & Semantic Technology (IJWesT) Vol.7, No.3/4, October 2016*.

Morsey, M., Lehmann, J., Auer, S., & Ngomo, A. C. (2011). DBpedia SPARQL benchmark–performance assessment with real queries on real data. *The Semantic Web–ISWC 2011* (pp. 454-469). Berlin, Heidelberg: Springer.

Motik, B., Grau, B., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (2009). *OWL 2 web ontology language: Profiles. W3C recommendation.* W3C recommendation, 27, 61.

Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., & Smith, M. (2012). *OWL 2 web ontology language: Structural specification and functional-style syntax.* W3C recommendation, 27(65), 159.

Munir, K., Odeh, M., & McClatchey, R. (2008). Ontology Assisted Query Reformulation Using the Semantic and Assertion Capabilities of OWL-DL Ontologies. *Twelfth International Database Engineering & Applications Symposium (IDEAS) 2008.*

Munir, K., Odeh, M., & McClatchey, R. (2009). Managing the Mappings between Domain Ontologies and Database Schemas when Formulating Relational Queries. *Proceedings of the 2009 International Database Engineering & Applications Symposium - IDEAS '09.*

Musa, D., Muñoz, L., & de Oliveira, J. P. (2004). Sharing Learner Profile through an Ontology and Web Services. *In Proceedings of the 15th International Workshop on Database and Expert Systems Applications* (pp. 415-419). IEEE.

Muzio, J., Heins, T., & Mundell, R. (2003). Experiences with reusable E-learning objects Experiences with reusable E-learning objects. *Internet and Higher Education*, 21-34.

Nafea, S., Maglaras, L., Siewe, F., Smith, R., & Janicke, H. (2016). Personalized Students' Profile Based on Ontology and Rule-based Reasoning. *ICST Transactions*.

Nanda, J., Simpson, T. W., Kumara, S. R., & Shooter, S. B. (2006). A methodology for product family ontology development using formal concept analysis and web ontology language. *Journal of computing and information science in engineering, 6(2)*, 103-113.

Navigli, R., & Paula, V. (2004). Learning domain ontologies from document warehouses and dedicated websites. *Computational Linguistics*, 151–179.

Neto, L. E., Vidal, V. M., Casanova, M. A., & Monteiro, J. M. (2013). R2RML by assertion: A semi-automatic tool for generating customised R2RML mappings. *In Extended Semantic Web Conference (pp. 248-252).* Berlin, Heidelberg: Springer.

Noy, N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*.

Ouf, S., Ellatif, M. A., Salama, S. E., & Helmy, Y. (2017). A proposed paradigm for smart learning environment based on semantic web. . *Computers in Human Behavior, 72*, 796-818.

Pak, J., & Zhou, L. (2009). A framework for ontology evaluation. *In Workshop on E-Business, Springer, Berlin, Heidelberg*, 10-18.

Paquette, G. (2007). An Ontology and a Software Framework for Competency Modeling and Management. *Journal of Educational Technology & Society*, 1-21.

Parsia, B., Matentzoglu, N., Gonçalves, R., Glimm, B., & Steigmiller, A. (2017). The OWL Reasoner Evaluation (ORE) 2015 Competition Report. *Journal of Automated Reasoning 59 (4)*, 455–482.

Paschke, A., & Schäfermeier, R. (2013). OntoMaven-Maven-based Ontology Development and Management of Distributed Ontology Repositories. *Synergies Between Knowledge Engineering and Software Engineering* (pp. 251-273). Springer, Cham.

Pepper, S. (2007). Expressing Dublin Core in Topic Maps. *In International Conference on Topic Map Research and Applications* (pp. 186-197). Berlin, Heidelberg: Springer.

Pinkel, C., Binnig, C., Jiménez-Ruiz, E., May, W., Ritze, D., . . . Kharlamov, E. (2015). RODI: A benchmark for automatic mapping generation in relational-to-ontology data integration. *In European Semantic Web Conference*, (pp. 21-37).

Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS), 10(2)*, 7-34.

Power, R., & Third, A. (2010). Expressing OWL axioms by English sentences: dubious in theory, feasible in practice. *The 23rd International Conference on Computational Linguistics*, (pp. 1006-1013).

Primo, T., Vicari, R., & Bernardi, K. (2012). User Profiles and Learning Objects as Ontology Individuals to Allow Reasoning and Interoperability in Recommender Systems. *In Global Engineering Education Conference (EDUCON)* (pp. 1-9). IEEE.

Pulido, J. R., Ruiz, M. A., Herrera, R., Cabello, E., Legrand, S., & Elliman, D. (2006). Ontology languages for the semantic web: A never completely updated review. *Knowledge-Based Systems, 19(7)*, 489-497.

Rani, M., Nayak, R., & Vyas, O. (2015). Anontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. *Knowledge-Based Systems*, 33-48.

Rath, A. S., Devaurs, D., & Lindstaedt, S. N. (2009). UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop. *In Proceedings of the 1st Workshop on Context, Information and Ontologies* (p. 8). ACM.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., & Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. *International Conference on Knowledge Engineering and Knowledge Management* (pp. 63-81). Berlin, Heidelberg: Springer.

Rodrıguez-Muro, M., & Calvanese, D. (2012). Quest, an OWL 2 QL reasoner for ontology-based data access. *OWLED.*

Rospocher, M., Tudorache, T., & Musen, M. A. (2014). Investigating collaboration dynamics in different ontology development environments . *International Conference on Knowledge Science, Engineering and Management* (pp. 302-313). Springer, Cham.

Roy, D., Sarkar, S., & Ghose, S. (2010). A comparative study of learning object metadata, learning material repositories, metadata annotation & an automatic metadata annotation tool. *Advances in Semantic Computing, 2(2010)*, 103-126.

Schmidt, M., Hornung, T., Lausen, G., & Pinkel, C. (2009). SP2Bench: a SPARQL performance benchmark. *In IEEE 25th International Conference on Data Engineering, 2009. ICDE'09* (pp. 222-233). IEEE.

Schoenfeld, A. H. (1985). A framework for the analysis of mathematical behavior. *Mathematical problem solving*, 11-45.

Selfa, D. M., Carrillo, M., & Boone, M. D. (2006). A database and web application based on mvc architecture. *16th IEEE International Conference on Electronics, Communications and Computers* (p. 48). IEEE Computer Society.

Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The semantic web revisited. *IEEE intelligent systems, 21(3)* , 96-101.

Sicilia, Á., & Nemirovski, G. (2016). AutoMap4OBDA: Automated generation of R2RML mappings for OBDA. *20th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2016* (pp. 577-592). Bologna, Italy: Springer International Publishing.

Sicilia, Á., Nemirovski, G., & Nolle, A. (2017). Map-on: A web-based editor for visual ontology mapping . *Semantic Web, 8(6)*, 969-980.

Sicilia, M. Á., Lytras, M. D., Sánchez-Alonso, S., García-Barriocanal, E., & Zapata-Ros, M. (2011). Modeling instructional-design theories with ontologies: Using methods to check, generate and search learning designs. *Computers in Human Behavior, 27(4), 1*, 1389-1398.

Simperl, E., & Luczak-Rösch, M. (2014). Collaborative ontology engineering: a survey. *The Knowledge Engineering Review, 29(1)*, 101-131.

Sir, M., Bradac, Z., & Fiedler, P. (2015). Ontology versus Database. *IFAC-PapersOnLine, 48(4)* , 220-225.

Sirin, E., & Parsia, B. (2007). SPARQL-DL: SPARQL Query for OWL-DL. *In OWLED (Vol. 258).*

Smith, M., McGuinness, D., Volz, R., & Welty, C. (2004). *Web ontology language (OWL) guide.* W3C.

Spanos, D. E., Stavrou, P., & Mitrou, N. (2012). Bringing relational databases into the semantic web: A survey. *Semantic Web, 3(2)*, 169-209.

Stojanovic, L., Staab, S., & Studer, R. (2001). ELearning based on the Semantic Web. *In WebNet2001-World Conference on the WWW and Internet*, (pp. 23-27).

Tarasova, T., & Marx, M. (2013). ParlBench: a SPARQL benchmark for electronic publishing applications. *In The Semantic Web: ESWC 2013 Satellite Events* (pp. 5-21). Springer Berlin Heidelberg.

Tavangarian, D., Leypold, M. E., Nölting, K., Röser, M., & Voigt, D. (2004). Is e-Learning the Solution for Individual Learning? *Electronic Journal of E-learning, 2(2)*, 273-280.

Tudorache, T., Noy, N. F., Tu, S., & Musen, M. A. (2008). Supporting collaborative ontology development in Protégé. *In International Semantic Web Conference* (pp. 17-32). Springer, Berlin, Heidelberg.

Tudorache, T., Nyulas, C., Noy, N. F., & Musen, M. A. (2013). WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web . *Semantic web, 4(1)*, 89-99.

Uschold, M. (1996). *Building Ontologies: Towards A Unified Methodology.* TECHNICAL REPORT-UNIVERSITY OF EDINBURGH ARTIFICIAL INTELLIGENCE APPLICATIONS INSTITUTE AIAI TR.

Vavliakis, K. N., Grollios, T. K., & Mitkas, P. A. (2010). Rdote-transforming relational databases into semantic web data. *International Conference on Posters & Demonstrations* (pp. 121-124). CEUR-WS. org.

Vavliakis, K. N., Grollios, T. K., & Mitkas, P. A. (2013). RDOTE–Publishing Relational Databases into the Semantic Web. *Journal of Systems and Software, 86(1)*, 89-99.

Vesin, B., Ivanović, M., Klašnja-Milićević, A., & Budimac, Z. (2013). Ontology-based architecture with recommendation strategy in java tutoring system. *Computer Science and Information Systems, vol 10(1).*, 237-261.

Villanueva-Rosales, N. (2011). *Formalizing relational databases as OWL ontologies (Doctoral dissertation).* Ottawa: Carleton University.

W3C. (2013). *W3C semantic web activity.* World Wide Web Consortium (W3C). 11th December.

Wang, T. (2014). Developing an assessment-centered e-Learning system for improving student learning effectiveness. *Computers & Education*, 189-203.

Wang, T. D., Parsia, B., & Hendler, J. (2006). A survey of the web ontology landscape. . *In International Semantic Web Conference* (pp. 682-694). Berlin, Heidelberg: Springer.

Wang, X., Staab, S., & Tiropanis, T. (2016). SPARQL benchmarking with automatically generated OLAP queries. *ISWC2016.*

Xu, Z., Zhang, S., & Dong, Y. (2006). Mapping between Relational Database Schema and OWL Ontology for Deep Annotation. *In Proceedings of the 2006 IEEE/WIC/ACM international Conference on Web intelligence* (pp. 548-552). IEEE Computer Society.

Yarandi, M., Jahankhani, H., & Tawil, H. (2013). A personalized adaptive e-learning approach based on semantic web technology. *Webology*.

Yessad, A., Faron-Zuckerc, C., Dieng-Kuntzb, R., & Laskri, M. (2011). Ontology-based semantic relatedness for detecting the relevance of learning resources. *Interactive Learning Environments. vol 19 (1) January 2011*, 63–80.

Yordanova, K. (2007). Meta-Data Application in Development, Exchange and Delivery of Digital Reusable Learning Content. *Interdisciplinary Journal of Knowledge and Learning Objects*.

Youn, S., & McLeod, D. (2006). Ontology development tools for ontology-based knowledge management. In *Encyclopedia of E-Commerce, E-Government, and Mobile Commerce* (pp. 858-864). IGI Global.

Zemmouchi-Ghomari, L., & Ghomari, A. (2013). Process of building reference ontology for higher education . *The World Congress on Engineering 2013 Vol III, WCE 2013, July 3 - 5, 2013.* London, U.K.

Zhang, D. (2003). Powering E-Learning In the New Millennium: An Overview of E-Learning and Enabling Technology. *Information Systems Frontiers*, 201-212.

Zhang, Y., Duc, P. M., Corcho, O., & Calbimonte, J. P. (2012). SRBench: a streaming RDF/SPARQL benchmark. *In The Semantic Web–ISWC 2012* (pp. 641-657). Springer Berlin Heidelberg.

Appendix

Table A.1: A summary of the key findings from the literature – the state of art

#	Paper Focus, Author(s) and year	Learning Ontology Design & Population	System Architecture	Personalisation Method	Ontology Languages, Reasoning & Querying	Technologies and Tools
1	Context, content and structure metadata to describe learning resources for e-learning (Stojanovic et al., 2001)	course ontology consists of content, context and structure ontology	Architecture of an e-learning Portal	IEE LOM to retrieve the learning materials	use the context, contents and structure details in a query interface; RDF repository, use rules	RDF, XML, ontology
2	A learning documents ontology model (Henze et al., 2004)	ontologies for three types of resources (domain, user, and observation)	a framework for adaptive/ personalised e-learning in the Semantic Web	Using LOM, IMS metadata three types of resources (domain, user, and observation)	RDF, reasoning rules for querying metadata and resources, FOL	TRIPLE for defining rules, reasoning and querying
3	A Web ontology for exchanging learner profiles (Musa et al., 2004)	Learner model ontology	Architecture for learner profile exchange using Web Services	IEEE LOM are used to describe the objects in the repositories	DAML+OIL	OilEd, RacerPro
4	Communication ontology and integration of learning ontologies (Aroyo & Dicheva, 2004)	a learner profile ontology and domain ontologies	a common reference architecture for adaptive concept-based web-based educational systems	IEEE LOM	RDF, Dialog-based language to query RDF data, get reasoning support from authoring tools.	Authoring support tools
5	Ontology-based education grid system for e-learning (Guangzuo et al., 2004)	Education ontology	Ontology-based grid computing architecture.	IMS, LOM, SCORM	OWL-S	OWL, Cocoon, Racer, WSDL, OWL-S, OWL-API
6	Ontology alignment for IT lesson planning (Kasai et al., 2005)	Ontology of the fundamental academic ability and Ontology of the goal of IT education	An architecture for ontology alignment and education resource annotation	IEEE LOM	RDF and RDF-Schema	'Hozo' editor for creating ontology
7	Generating a learning ontology from a relational database (Li, DU, & Wang, 2005)	A learning ontology	N/A	N/A	RDF, RDFS, OWL	Relational databases, database to ontology mapping rules
8	A learning design ontology based on the IMS specification (Amorim et al., 2006)	IMS LD ontology	based on intelligent agent technology, and follows a multi-layer topology	IMS LD	OWL	Protégé
9	An ontology and a	Competency	the Semantic	Metadata is used	OWL	ADISA - a

#	Paper Focus, Author(s) and year	Learning Ontology Design & Population	System Architecture	Personalisation Method	Ontology Languages, Reasoning & Querying	Technologies and Tools
	software framework for competency modelling and management (Paquette, 2007)	ontology, general skills ontology, performance indicator ontology	Web and the ontology-driven architecture of TELOS	but not specified.		web-based workbench
10	Meta-data application in development, exchange and delivery of digital reusable learning content (Yordanova, 2007)	Metadata ontology	A model for meta-data integration in RLOs sharing	IEEE LOM, Learner profiles (learner preferences, learning style of the learner, competencies, prior experience and skills)	Not clear (OWL?)	Protégé
11	An ontology-based planning system for e-course generation (Kontopoulos et al., 2008)	Competency ontology	PARSER systems architecture based on metadata and ontology	LOM repository, learner profiles, preferences, needs and abilities	RDFS	XML, RDF, LOM,
12	Ontology supported personalised e-learning repositories (Dutta et al., 2009)	Document ontology, student ontology	A conceptual learning space and call a semantic learning layer cake	IEEE LOM and Dublin Core	OWL-DL, logic rules using N3Logic	Protégé
13	Advanced ontology management system for personalised e-learning (Gaeta at al., 2009)	An integrated framework with a set of tools for representing and managing learning ontologies	Web Teacher (IWT) platform	Not clear	OWL,	Protégé, AOMS is realised using GENESIS
14	An Ontology-Based User Interaction Context Model (UICO) for automatic task detection on the computer desktop (Rath, Devaurs, & Lindstaedt, 2009)	User Interaction Context Ontology (UICO)	User's conceptual model is given as a Semantic Pyramid.	Metadata is used. Not clear which standard.	OWL-DL	protégé
15	Ontology design for creating adaptive learning path in e-pearning environment (Chung & Kim, 2012)	Curriculum ontology, Syllabus ontology, and Subject ontology	The System Architecture ontology-based learning support system	Not discussed	SPARQL or TMQL	Not discussed
16	User profiles and learning objects as ontology individuals to allow reasoning and interoperability in	Learning Objects Ontology and User profiles' Ontology	*Lassique*, an educational Recommendatio n System	Agent Based Learning Objects (OBAA) metadata standard derived from the IEEE LOM	OWL, domain rules, Reasoning techniques or reasoners used are not clear	Protégé

#	Paper Focus, Author(s) and year	Learning Ontology Design & Population	System Architecture	Personalisation Method	Ontology Languages, Reasoning & Querying	Technologies and Tools
	recommender systems (Primo et al., 2012)					
17	A personalised adaptive e-learning approach based on Semantic Web technology (Yarandi et al., 2013)	Four ontology models: domain model, user model, content model and test model	Ontology-based personalised e-learning systems architecture	account learner abilities, learning styles, preferences and levels of knowledge, FSLSM	OWL, Reasoning techniques or reasoners used are not clear	OWL
18	An ontology based approach for modeling e-learning in healthcare human resource management (Bajenaru & Smeureanu, 2015)	Student profile ontology, domain ontology, e-learning system ontology	learning systems ontology	Abstraction of knowledge over Los is represented by metadata. IMS standard. Personalisation based on ontology.	Not discussed	Protégé, WebODE and OntoEdit
19	A Machine-readable Ontology for Teaching word problems in Mathematics (MONTO) (Lalingkar et al., 2015)	Four ontologies: system/pedagogy ontology, strategy/task ontology, user model/student ontology, and domain ontology	Not discussed	Errors and hints are modelled to provide personalised training	OWL	Protégé
20	An ontology-based adaptive personalised e-learning system, assisted by software agents on Cloud storage (Rani et al., 2015)	two ontologies the course.owl and user.owl	An ontology-based adaptive personalised e-learning system architecture	Learning styles are monitored by agents and adapts the contents	OWL and OWL DL, reasoning	Protégé, Cloud-based storages (ontology and database), agents, Hermit Reasoner
21	Ontology-based model for LOM (Kalogeraki et al., 2016)	EduSor ontology	Education Assessor Model (EduSor) Architecture	LOM; adaptive learning paths	OWL, SPARQL	LOM, RDF, OWL
22	A SKOS-based framework for subject ontologies to improve learning experiences (Miranda et al. 2016)	Subject ontologies	Semantic Web-based Educational Systems (SWBESs) architecture	IEEE LOM or Dublin Core	OWL, SPARQL (four queries), reasoning is done	A visual editor tool for Semantic Web-based vocabulary.
23	Towards situation driven mobile tutoring system for learning languages and communication skills (Khemaja & Taamallah, 2016)	Context ontology, goals ontology and domain ontologies	An adaptable and re-configurable mobile Intelligent Tutoring System (ITS) architecture	Not discussed	AndroJena plug-in for reasoning among and querying ontologies	Protégé
24	Personalised students' profile	Learning domain ontology and	Student profile architecture	Analyses learning patterns	OWL, rule-based reasoning	Protégé

#	Paper Focus, Author(s) and year	Learning Ontology Design & Population	System Architecture	Personalisation Method	Ontology Languages, Reasoning & Querying	Technologies and Tools
	based on ontology and rule-based reasoning (Nafea et al., 2016)	reference ontology (user profiles)		based on Myers-Briggs Type Indicator and FSLSM		
25	A proposed paradigm for smart learning environment based on Semantic Web (Ouf et al., 2017)	Learner model ontology, learning object ontology, learning activities ontology and teaching methods ontology	An architecture with four layers: Interface layer, Semantic reasoning mechanism layer, Semantic layer and Semantic meta data layer	Using a learner model.	Semantic Web Rule Language (SWRL) for personalisation.	Protégé, SWRL

Table A.2: An Excerpt of MQ Ontology: An Example of OWL 2 Learn profile

An Excerpt of the MQ Ontology	**The OWL 2 Learn Constructor**	**Table 4.4 Ref.**
Declaration(Class(:Person))	Class	#2
EquivalentClasses(:Person ObjectSomeValuesFrom(:isAssociatedTo :Organisation))	ObjectSomeValuesFrom	#5
SubClassOf(:Person owl:Thing)	Owl:Thing	#1
	SubClassOf	
SubClassOf(:Staff :Person)		
SubClassOf(:Student :Person)		
SubClassOf(:TeachingStaff :Staff)		
Declaration(Class(:Topic))		
SubClassOf(:Topic owl:Thing)		
SubClassOf(:Topic ObjectAllValuesFrom(:hasScheduled :TopicList))	ObjectAllValuesFrom	#4
Declaration(Class:TopicList))		
SubClassOf(:TopicList owl:Thing)		
SubClassOf(:TopicList ObjectSomeValuesFrom(:hasScheduledTopic :Topic))	ObjectSomeValuesFrom	#5
SubClassOf(:Unit owl:Thing)		
Declaration(Class(:AssessmentTask))		
EquivalentClasses(:AssessmentTask	EquivalentClasses	
ObjectIntersectionOf(owl:Thing	ObjectIntersectionOf	#3
ObjectSomeValuesFrom(:isWrittenBy :TeachingStaff)	ObjectSomeValuesFrom	#5
ObjectExactCardinality(1 :isAssessmentMethodOf :Unit)))	QualifiedExactCardinality	#12
	DisjointClasses	#13
DisjointClasses(:Assignment :FinalExam)		
SubClassOf(:AssessmentTask owl:Thing)		
subClassOf(:FinalExam :AssessmentTask)		
subClassOf(:Assignment :AssessmentTask)		
Declaration(ObjectProperty(:hasPrerequisite))	ObjectProperty	#6
TransitiveObjectProperty(:hasPrerequisite)	TransitiveObjectProperty	#9
InverseObjectProperties(:isPrerequisiteOf :hasPrerequisite)	InverseObjectProperties	#11
ObjectPropertyDomain(:hasPrerequisite :Unit)	ObjectPropertyRange	
ObjectPropertyRange(:hasPrerequisite :Unit)	ObjectPropertyDomain	

An Excerpt of the MQ Ontology	**The OWL 2 Learn Constructor**	**Table 4.4 Ref.**
Declaration(ObjectProperty (:commitsTo)) InverseObjectProperties(:commitsTo :isCommitedBy) ObjectPropertyDomain(:commitsTo :Student) ObjectPropertyRange(:commitsTo :Enrolment) ObjectPropertyRange(:commitsTo ObjectMaxCardinality(5 :commitsTo :Enrolment)) Declaration(ObjectProperty:assignmentOf)) SubObjectPropertyOf(:assignmentOf :isAssessmentMethodOf) InverseObjectProperties(:has Assignment:assignmentOf)	 MaxQualifiedCardinality SubObjectPropertyOf	 #12 #10
Declaration(DataProperty(:assignmentMark)) SubDataPropertyOf(:assignmentMark :assessmentMark) DataPropertyDomain(:assignmentMark :AssignmentSubmission) DataPropertyRange(:assignmentMark^^ xsd:integer)	DataProperty SubDataPropertyOf	#14 #10
ClassAssertion(:Unit :ISYS114) ObjectPropertyAssertion(:isTutorOf :JohnParker :ISYS114)	ClassAssertion ObjectPropertyAssertion	#7 #7

Table A.3: Analog of UOBM Queries for the MQ ontology

#	**UOMB Query, source:** (Ma et al., 2006)	**Analog of UOMB Query for the MQ Ontology**
1	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:UndergraduateStudent . ?x benchmark:takesCourse http://www.Department0.University0.edu/Course0}	SELECT DISTINCT ?x WHERE {?x rdf:type :UndergraduateStudent. ?x :studies :ISYS114 }
2	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:Employee }	SELECT DISTINCT ?x WHERE {?x rdf:type :Staff.}
3	SELECT DISTINCT ?x WHERE {?x rdf:type benchmark:Student . ?x benchmark:isMemberOf http://www.Department0.University0.edu }	SELECT DISTINCT ?student WHERE {?x rdf:type :Student. ?x :isStudentOf :Computing }
4	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:Publication . ?x benchmark:publicationAuthor ?y . ?y rdf:type benchmark:Faculty . ?y benchmark:isMemberOf http://www.Department0.University0.edu }	SELECT DISTINCT ?x WHERE {?x rdf:type :LearningResource. ?x :hasAuthor ?y. ?y rdf:type :TeachingStaff. ?y :isStaffOf :Computing}
5	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:ResearchGroup . ?x benchmark:subOrganizationOf http://www.University0.edu }	SELECT ?x WHERE { ?x rdf:type :Unit. ?x :isPrerequisiteOf+ ?y. FILTER (?y=:COMP365)}
6	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:Person . http://www.University0.edu benchmark:hasAlumnus ?x }	SELECT DISTINCT ?person WHERE {?x rdf:type:Person. :Computing :hasGraduate ?x.}
7	SELECT DISTINCT ?x WHERE {?x rdf:type benchmark:Person . ?x benchmark:hasSameHomeTownWith http://www.Department0.University0.edu/FullProfessor0}	N/A
8	SELECT DISTINCT ?x WHERE {?x rdf:type benchmark:SportsLover . http://www. Department0.University0.edu benchmark: hasMember ?x}	SELECT DISTINCT ?x WHERE {?x rdf:type :Lecturer. :Computing :hasStaff ?x}
9	SELECT DISTINCT ?x WHERE {?x rdf:type benchmark:GraduateCourse . ?x benchmark:isTaughtBy ?y . ?y benchmark:isMemberOf ?z .?z benchmark:subOrganizationOf http://www.University0.edu }	SELECT ?x WHERE { ?x rdf:type :LectureSlides. ?x :isLectureSlidesOf ?y. ?y :isPrerequisiteOf+ ?z. FILTER (?z=:COMP365)}
10	SELECT DISTINCT ?x WHERE { ?x benchmark:isFriendOf http://www.Department0.University0.edu/FullProfessor0}	N/A
11	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:Person . ?x benchmark:like ?y . ?z rdf:type benchmark:Chair . ?z benchmark:isHeadOf http://www.Department0.University0.edu . ?z benchmark:like ?y}	N/A

#	**UOMB Query, source:** (Ma et al., 2006)	**Analog of UOMB Query for the MQ Ontology**
12	SELECT DISTINCT ?x WHERE {?x rdf:type benchmark:Student . ?x benchmark:takesCourse ?y .?y benchmark:isTaughtBy http://www.Department0.University0.edu/FullProfessor0 }	SELECT DISTINCT ?x WHERE {?x rdf:type :Student. ?x :studies ?y. ?y :hasLecturer :DebbieRichards}
13	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:PeopleWithHobby . ?x benchmark:isMemberOf http://www.Department0.University0.edu}	SELECT DISTINCT ?x WHERE {?x rdf:type :TeachingStaff. ?x :isStaffOf :Computing}
14	SELECT DISTINCT ?x WHERE { ?x rdf:type benchmark:Woman . ?x rdf:type benchmark:Student . ?x benchmark:isMemberOf ?y . ?y benchmark:subOrganizationOf http://www.University0.edu }	N/A
15	SELECT DISTINCT ?x WHERE {?x rdf:type benchmark:PeopleWithManyHobbies . ?x benchmark:isMemberOf http://www.Department0.University0.edu }	N/A

Table A.4: Inferences and OWL 2 constructors associated to each UOBM query

Q#	**Inferences of UOBM Queries, source:** (Ma et al., 2006)	**OWL 2 Constructors**
1	It only needs simple conjunction <a rdf:type UndergraduateStudent>, <a takesCourse http://www.Department0.University0.edu/Course0> → <a>	ObjectProperty()
2	Domain(worksFor, Employee), <a worksFor b> → <a rdf:type Employee>, Domain(worksFor,Employee), researchAssistant ⊑∃ worksFor.ResearchGroup → researchAssistant ⊑ Employee	SubClassOf(), ObjectSomeValuesFrom(), ObjectProperty(), Domain(), Range()
3	Range(takeCourse,Student) , GraduateStudent ≥1 takeCourse → GraduateStudent ⊑ Student	SubClassOf(), ObjectSomeValuesFrom(), ObjectProperty(), Domain(), Range(), ObjectMinCardinality()
4	SubClass: Faculty = FullProfessor ⊔ AssociateProfessor ⊔…⊔ ClericStaff, Publication=Article ⊔… ⊔Journal	SubClassOf(), UnionOf(), ObjectProperty(), ObjectMinCardinality(), DisjointClasses()
5	Transitive(subOrganizationOf), <a subOrganizationOf b>, <b subOrganizationOf http://www.University0.edu> → <a subOrganizationOf http://www.University0.edu>	ObjectProperty(), TransitiveObjectProperty()
6	Inverse(hasAlumni, hasDegreeFrom), <a hasDegreeFrom b> → <b hasAlumnus a>	ObjectProperty(), InverseObjectProperties()
7	Transitive(hasSameHomeTownWith), Symmetric(hasSameHomeTownWith), <a hasSameHomeTownWIth b>, <c hasSameHomeTownWIth b> → < a hasSameHomeTownWith c>	Class(), TransitiveObjectProperty(), TransitiveObjectProperty(),**SymmetricProperty(),** **SameAs()**
8	<x like y>, <y rdf:type Sports>, SportLover like.Sports → <x rdf:type SportLover> subProperty(isCrazyAbout, like), SportFan ⊑∃isCrazyAbout.Sports → SportFan ⊑ ∃portLover **On OWL DL ontology:** SwimmingLover⊑∃like.**{Swimming}**→ SwimmingLover ⊑ SportsLover …	SubClassOf(), ObjectSomeValuesFrom(), ObjectMinCardinality(), ObjectProperty(), DisjointClasses(), SubObjectPropertyOf() oneOf()
9	GraduateStudent ≡∀takesCourse.GraduateCourse, <a rdf:type GraduateStudent>, <a takesCourse b> → <b rdf:type GraduateCourse>	SubClassOf(), ObjectProperty(), ObjectAllValuesFrom()
10	Symmetric(isFriendOf), <a isFriendOf b> → <b isFriendOf a>	Domain(), Range(), **SymmetricProperty()**
11	FunctionalProperty(isHeadOf), <a isHeadof b>, <c isHeadOf b) → <a sameAs c> // there are some same individuals of chair0	Class(), ObjectProperty(), FunctionalProperty(), **SameAs()**
12	GraduateStudent ≡∀ takesCourse.GraduateCourse ⊓ ≥ 1.takesCourse,	SubClassOf(), ObjectIntersectionOf(), ObjectAllValuesFrom()

Q#	**Inferences of UOBM Queries, source:** (Ma et al., 2006)	**OWL 2 Constructors**
	Domain(takesCourse, Student) → Student ⊒ GraduateStudent	ObjectProperty(), Domain(), Range(), ObjectMinCardinality(),
13	Lite Cardinality: PeopleWithHobby(≥1 like) ⊒ SportLover, <a like b> → <a rdf:type PeopleWithHobby>	SubClassOf(), ObjectProperty(), ObjectMinCardinality()
14	<a,isStudentof b>, <b rdf:type WomanCollege>, WomanCollege ⊑ ∀ hasStudent.(¬Man), disjoint(Man, Woman), Man ⊔Woman ≡ Person → <a rdf:type Woman>	UnionOf(), ObjectAllValuesFrom(), ObjectProperty(), DisjointClasses(), **ObjectComplementOf()**
15	PeopleWithManyHobbies ⊑≥3like, <a like b1> … <a like bn>, all different(b1,b2…bn) → <a rdf:type PeopleWithManyHobbies> // n ≥3	ObjectMinCardinality(), ObjectProperty(), **AllDifferent()**

Note: Class(), ClassAssertion() and ObjectPropertyAssertion() are involved in all queries

Table A.5: Inference involved in MQ and CSU ontologies

Q#	Purpose of the Query	Inferences on MQ Ontology	Inferences on CSU Ontology
1	Find all the undergraduate students who study ISYS114. (ITC106 for CSU)	UndergraduateStudent(?x) studies(?x, ISYS114)→x	UndergraduateStudent(?x) studies(?x, ITC106)→ x
2	Find all the staff members.	Domain(isAssociatedTo, Staff), <a isAssociatedTo b> → <a rdf:type Staff > Domain(isAssociatedTo, Staff), Lecturer ⊑∃isTeachingStaffOf. Department → Lecturer⊑ Staff	Domain(isAssociatedTo, Staff), <a isAssociatedTo b> → <a rdf:type Staff> Domain(isAssociatedTo, Staff), Lecturer ⊑∃isAcademicOf. School → Lecturer⊑ Staff
3	Find out all the students of the Computing department.	Range(studies, Student) , UndergraduateStudent ≥1 studies → UndergraduateStudent ⊑ Student	Range(studies, Student) , UndergraduateStudent ≥1 studies → UndergraduateStudent ⊑ Student
4	Find all the learning resources authored by teaching staff of the Computing department.	TeachingStaff = Lecturer ⊔ Moderator ⊔…⊔ Tutor, LearningResource=LectureSlides ⊔ …⊔ ILecture	AcademicStaff = Lecturer ⊔ Convener ⊔…⊔ Tutor, LearningMaterial=LectureSlides ⊔ …⊔ ILecture
5	Find all the prerequisites of COMP365. (ITC309 for CSU)	Transitive(isPrerequisiteOf), <a isPrerequisiteOf b>, <b isPrerequisiteOf :COMP365>→ <a isPrerequisiteOf :COMP365>	Transitive(isPrerequisiteOf), <a isPrerequisiteOf b>, <b isPrerequisiteOf :ITC309>→ <a isPrerequisiteOf :ITC309>
6	Find all the graduated students of the Computing department.	Inverse(hasGraduate, isGraduateOf), <a hasGraduate b> → <b isGraduateOf a>	Inverse(hasGraduate, isGraduateOf), <a hasGraduate b> → <b isGraduateOf a>
7	Find all the lecturers of the Computing department.	<x isLecturerOf y>, <y rdf:type Unit>, Lecturer⊑∃isLecturerOf.Unit → <x rdf:type Lecturer> subProperty(isLecturerOf, isTeachingStaffOf), Lecturer⊑∃isLecturerOf.Unit→ Lecturer ⊑ TeachingStaff.	<x isLecturerOf y>, <y rdf:type Subject>, Lecturer⊑∃isLecturerOf.Subject → <x rdf:type Lecturer> subProperty(isLecturerOf, isAcademicOf), Lecturer⊑∃isLecturerOf.Subject→ Lecturer ⊑ AcademicStaff.
8	Find all the undergraduate units of the Faculty of Science and Engineering.	UndergraduateStudent ≡∀studies.UndergraduateUnit, <a rdf:type UndergraduateStudent>, <a studies b> → <b rdf:type UndergraduateUnit>	UndergraduateStudent ≡∀studies.UndergraduateUnit, <a rdf:type UndergraduateStudent>, <a studies b> → <b rdf:type UndergraduateUnit>
9	Find all the students who study the units taught by Debbie Richards.	UndergraduateStudent ≡∀studies. ≥1Unit, Domain(studies, Student) → Student ⊒ UndergraduateStudent	UndergraduateStudent≡∀studies.≥1Unit, Domain(studies, Student) → Student ⊒ UndergraduateStudent
10	Find all the teaching staff who teach some units in the	Lecturer ⊒ isLecturerOf. ≥1 Unit,	Lecturer ⊒ isLecturerOf. ≥1 Unit,

Q#	Purpose of the Query	Inferences on MQ Ontology	Inferences on CSU Ontology
	Computing department.	<a isLecturerOf b> → <a rdf:type Lecturer>	<a isLecturerOf b> → <a rdf:type Lecturer>
11	Find the learning resources that are only presentations and in the pdf format.	<a ContentType b> <a format c> → a	<a ContentType b> <a format c> → a
12	Find the subjects with less than three assignments.	<x rdf:type Unit>, <x noOfAssignments 3> COUNT(<x hasAssignment y>)<3 → x	<a rdf:type Subject> <x noOfAssignments 3> COUNT(<x hasAssignment y>)<3 → x
13	Find all the assessment tasks with their due date and the return date.	<b DisjointDataProperties c> <a dueDate b>, <a returnDate c> →b, c	<b DisjointDataProperties c> <a dueDate b>, <a returnDate c>→b, c