TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE
SCHOOL OF APPLIED TECHNOLOGY
DEPARTMENT OF APPLIED INFORMATICS AND MULTIMEDIA

SEMANTIC MODELING OF EDUCATIONAL CURRICULUM & SYLLABUS

by
EVANGELOS KATIS

A Thesis

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Approved by supervisor Professors

Vassilakis Konstantinos Papadakis Nikolaos Kalogiannakis Michail

2018

CONTENTS

TABLE OF FIGURES

List of Abbreviations

CCSO: Curriculum-Course-Syllabus-Ontology

AI: Artificial Intelligence

ICT: Information and Communication Technologies

CS: Computer Science

IoT: Internet of Things

RDF: Resource Description Framework

RDFS: RDF Schema

OWL: Web Ontology Language

IES: Intelligent Educational System

DC: Dublin Core

LMS: Learning Management System

IIS: Intelligent Instructional System

AI-ED: Artificial Intelligence and Education

ECTS: European Credit Transfer System

O4E: Ontologies for Education

AIISO: Academic Institution Internal Structure Ontology

SWRC: Semantic Web for Research Communities

HQA: Accreditation Template of Hellenic Quality Assurance & Accreditation Agency

CSC2013: Computer Science Curricula 2013

ACM: Association for Computing Machinery

IEEE: Institute of Electrical and Electronics Engineers

KU: Knowledge Unit

KA: Knowledge Area

BSc: Bachelor of Science

MSc: Master of Science

OOPS: OntOlogy Pitfalls Scanner

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my thesis supervisors, Professor Kostas Vassilakis as major supervisor, Professor Nikos Papadakis, Professor Michail Kalogiannakis, and Dr. Haridimos Kondylakis, for their valuable scientific support and assistance. My special thanks are extended to all professors and staff in MSc Informatics and Multimedia study program. Also, I wish to thank my family and my wife for their support and encouragement throughout my study.

Evangelos Katis

Postgraduate student in MSc Informatics and Multimedia of Department of Informatics Engineering of Technological Educational Institute (TEI) of Crete.

Heraklion, Crete, 2018.

This work is licensed under a Creative Commons license Attribution-NonCommercial-ShareAlike 4.0 International

ABSTRACT

Applying Semantic Web and Ontological technology in education offers powerful benefits in several areas of interest and will change the current education mode. Ontologies can be used for knowledge organization, knowledge inference and planning, information querying, visualization and navigation as well as for the semantic annotation of learning resources for knowledge reuse and interoperability. Learning platforms can be supported by ontologies for the construction, externalization, communication and assessment of knowledge and competence. Ontologies can also be used for describing the knowledge about a subject domain and modeling educational system architecture.

The purpose of this thesis is the semantic annotation of concepts and entities within a tertiary education curriculum. The developed ontology aims to conceptualize educational entities within Curriculum and Syllabus with sufficiency and richness in order to support rich services on top for improving curriculum management and development as well as enabling syllabus semantic searching, matching, interlinking and recommendation.

The thesis describes the methodology and development process of the educational ontology, the elements included, as well as its evaluation process.

The ontology is designed to be highly transferable and reusable for any type of study program, course and syllabus. The ontology is available online and a triple-store is used for the storage and retrieval of triples of the ontology through semantic queries.

KEYWORDS

Semantic Web, Ontology, Curriculum, Course, Syllabus.

ΠΕΡΙΛΗΨΗ

Η εφαρμογή τεχνολογιών Σημασιολογικού Ιστού και Οντολογιών στην εκπαίδευση προσφέρει σημαντικά οφέλη σε διάφορους τομείς ενδιαφέροντος αλλάζοντας το πεδίο της εκπαίδευσης. Οι οντολογίες μπορούν να χρησιμοποιηθούν για την οργάνωση της γνώσης, τον σχεδιασμό και συμπερασμό της γνώσης, την αναζήτηση πληροφοριών, την οπτικοποίηση και πλοήγηση καθώς και για τη σημασιολογική σημείωση των πόρων μάθησης ενισχύοντας με τον τρόπο αυτό τη διαλειτουργικότητα και επαναχρησιμοποίηση της γνώσης. Οι πλατφόρμες μάθησης μπορούν να βοηθηθούν από οντολογίες για την κατασκευή, την εξωτερίκευση, την επικοινωνία και την αξιολόγηση των γνώσεων και των ικανοτήτων. Οι οντολογίες μπορούν επίσης να χρησιμοποιηθούν για την περιγραφή της γνώσης σχετικά με έναν θεματικό τομέα και για τη μοντελοποίηση της αρχιτεκτονικής του εκπαιδευτικού συστήματος.

Σκοπός αυτής της διπλωματικής εργασίας είναι η σημασιολογική περιγραφή εννοιών και οντοτήτων σε ένα πρόγραμμα σπουδών Τριτοβάθμιας εκπαίδευσης. Η αναπτυχθείσα οντολογία στοχεύει στην μοντελοποίηση των εκπαιδευτικών οντοτήτων που συναντιόνται εντός του Προγράμματος Σπουδών (Curriculum) και του Σχεδίου Μαθήματος (Syllabus) με ικανοποιητική επάρκεια και λεπτομέρεια προκειμένου να υποστηριχθούν υπηρεσίες για τη βελτίωση της διαχείρισης και ανάπτυξης του Προγράμματος Σπουδών (Curriculum), καθώς και για τη διευκόλυνση της σημασιολογικής αναζήτησης, αντιστοίχισης, διασύνδεσης και σύστασης Σχεδίων Μαθημάτων (Syllabus).

Η εργασία περιγράφει τη μεθοδολογία και τη διαδικασία ανάπτυξης της εκπαιδευτικής οντολογίας, τα στοιχεία που περιλαμβάνονται σε αυτήν, καθώς και τη διαδικασία αξιολόγησης της.

Η οντολογία έχει σχεδιαστεί ώστε να είναι μεταφέρσιμη και επαναχρησιμοποιήσιμη για κάθε τύπο Προγράμματος Σπουδών, Μαθήματος και Σχεδίου Μαθήματος. Η οντολογία είναι διαθέσιμη στο διαδίκτυο ενώ έγινε χρήση online Βάσης Δεδομένων αποθήκευσης Οντολογιών για την αποθήκευση και την ανάκτηση των τριπλέτων της οντολογίας μέσω σημασιολογικών ερωτημάτων.

GENERAL PART
CHAPTER 1. INTRODUCTION

During the last decades, the rapid development of *Information and Communication Technologies* (*ICT*) has significant impact in many domains of society providing a wide range of opportunities and applications. In the area of Education, the application of information technologies can improve the efficiency of teaching and learning process and enrich the learning experience while it can support the management and reduce the cost.

Effective knowledge and information sharing between systems and applications is one of the key objectives of Knowledge Management Systems in order to reduce the cost in money and time. In World Wide Web and Databases there is a huge amount of information and resources that can shared and reuse. However, the lack of standardization combined with the diversity of applications and the incompatibility of languages and vocabularies are major issues that need to be addressed in order that information and knowledge can be easily discovered and reused.

Semantic Web technologies and Ontologies offer great potential for improving the discoverability and reusability of information and resources. Ontology conceptualizes a domain of knowledge using formal language understood not only by humans but also by computers. Ontologies describe the concepts in a domain with the relationships between them, and also provide vocabulary for the used terms. The semantic annotation of concepts as well as the used terminology can be processed and comprehended by humans and machines. Ontologies facilitate the semantic interoperability and automatic inference of information and improve knowledge sharing and reuse among heterogeneous systems and applications.

The purpose of this thesis is the semantic annotation of concepts and entities within an academic institution aiming to model the core concepts of a tertiary education curriculum. The developed ontology conceptualizes educational entities within curriculum and syllabus. Curriculum, Course and Syllabus are considered as top-level entities that provide important information and play significant role in many educational processes and learning activities in an educational organization. Our ontology comprises of 41 concepts in a taxonomy, 9 of

which considered as top-level concepts of the ontology. It also includes 54 objects properties for establishing relations between concepts and 76 data properties for describing concepts characteristics in detail. The ontology aims to support rich services on top for improving curriculum management and development as well as enabling syllabus semantic searching, matching, interlinking and recommendation. It designed to be suitable for all types of study programs in tertiary education and can easily reused and configured from any educational institution. We also provide rich internal and external documentation with various pieces of information for the developed ontology. Internal documentation includes information annotated in ontology elements as metadata. All classes and properties have been written in two languages, English and Greek. External documentation comprised mainly by this thesis document, where the whole ontology development process is described in detail.

The thesis document is structured as follows. Chapter 1 gives an initial introduction in Semantic Web and Ontologies, their purpose, usage and applications. Chapter 2 reviews the literature of ontologies in Education, classified in five main categories. Moreover, ontologies related to the context of Curriculum and Course, are further analyzed.

Chapter 3 describes the methodology and development process of the educational ontology in detail, including purpose, class definitions, description of class properties, summary of definitions, and descriptive presentation of each class. Chapter 4 analyzes the results of the evaluation of our ontology and the documentation provided. Finally, Chapter 5 concludes with the synopsis of our work and discusses about some difficulties encountered and possible scenarios for using our ontology.

1.1. SEMANTIC WEB

It is beyond doubt that World Wide Web has changed the way people communicate and work. Its structure and content is mainly consumed by humans who can browse and retrieve information from Web documents. However, as new revolutionary technologies emerge, such as Internet of Things (IoT) and Artificial Intelligence, new challenges have arisen for a new model of Web that can be used also by machines and applications.

Web 3.0 has emerged as an evolution for a more connected, open and intelligent Web, where humans and machines browse, communicate and share content. Semantic Web plays key role in that.

Tim Berners-Lee and his fellows in the 2001 Scientific American article [1], presented an evolutionary transformation of the existing Web to the, so called, *Semantic Web*.

> *"The Semantic Web is an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation."*

According to Tim Berners-Lee, Semantic Web will be an extension of today's Syntactic Web that will become reality by attaching metadata to meaningful information included in web pages and also linking databases to the web. Information served in Semantic Web could be automatically processed and understood by machines as well as humans. Semantic Web will facilitate significant improvement in the search, exchange, merge, manage and usage of information worldwide.

Sometimes, the term Semantic Web is used as a synonym for Web 3.0, however, those two terms are significantly different. The term Web 3.0 suggested in 2006 by *John Mark off* of New York Times as third generation of the Web [2]. Actually, Semantic Web is the key component for the development of Web 3.0 as the meaningful connected and shareable content, offered by Semantic Web, will empower other features of Web 3.0. *Connectivity* of everyone and everything means that every device is connected and the information shared is more connected thanks to semantic metadata. In addition, the *Ubiquity* of content offers information accessible by multiple applications and services in every device. All these features, in combination with natural language processing, will lead *Artificial Intelligence* on

a new era where computers and machines will be able to comprehend information like humans.

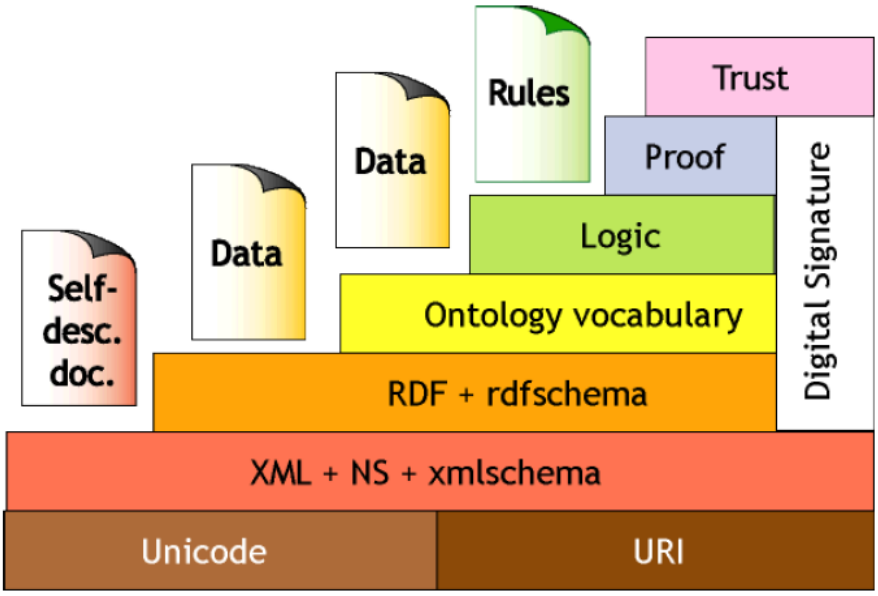


Figure 1: Semantic Web Layer Cake (source: W3C[1])

Figure 1 presents the layer structure of Semantic Web, as envisioned by Tim Berners-Lee [1]. *XML* is a markup language for creating structured Web documents using a user-defined vocabulary. *RDF* is a data model for expressing simple statements about Web objects. *Logic* layer is used to enhance ontology language further and to allow the writing of application-specific knowledge declarations. *Proof* layer enables the deductive process as well as the representation of proofs and proof validation. *Trust* layer will emerge through the use of digital security mechanisms when users have trust in its operations and quality of information [3].

Humans understand the top three layers, *Trust*, *Proof*, *Logic* but machines do not understand them because computers cannot decipher meaning and relations from these layers. On the other hand, machines comprehend meaning from lower layers, *URI*, *XML* and *RDF*. Building the top three layers upon an *Ontology vocabulary* layer which is built upon

[1] Semantic Web Layer Cake (W3C): www.w3.org/2001/09/06-ecdl/slide17-0.html

RDF layer and linking them to each other, makes it possible for machines, by using the lower layers, to read and comprehend the higher layers.

1.2. ONTOLOGY

1.2.1. WHAT IS AN ONTOLOGY

Many definitions of ontologies have been given but one of the most known definitions is the one given by *Gruber* in 1993, who defines ontology as "*an explicit specification of a conceptualization*" [4]. Gruber's definition emphasizes in the *specification* of concepts and relationships which will facilitate knowledge sharing and reuse.

In 1997, *Borst* defined ontology as "*a formal specification of a shared conceptualization*" [5]. This definition underlines that a conceptualization should express *shared* representation of knowledge and should be expressed in a machine readable format.

In 1998, *Studer et al*. merged these two definitions [6]:

"*An ontology is a formal, explicit specification of a shared conceptualization*".

They also explained the keywords of their definition. '*Formal'* means that the expressions used should be machine readable. '*Explicit*' means that the concepts and their relations should be explicitly defined. '*Shared*' expresses the fact that the knowledge captured by ontology should not be private but shared between several parties. As far as the term '*conceptualization*' is concerned, a comprehensible definition given by *Genesereth & Nilsson*: "*A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose*" [7].

According to a recent definition by *Gruber* again, in the area of computers and information science, "*an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse*" [8]. Ontology models a concrete domain of knowledge, interest or functions.

1.2.2. WHAT IS IN AN ONTOLOGY

In the last definition by Gruber [8], the '*representational primitives*' are typically *classes* (or sets), *properties* (or attributes) of classes and *relationships* (or relations) among classes. Additional information about their usage and meaning is also included (e.g. axioms, rules, restrictions).

Typically, any ontology contains:

- **Classes** that represent all types of concepts (either physical/specific or abstract/conceptual). They can be organized in taxonomy using superclass-sub-class relations.
- **Relationships** (or relations) that establish connection between concepts.
- **Properties** (also called attributes or slots) used to describe details of the concepts.
- **Axioms** state expressions that are always true.
- **Rules** express cases of assumptions (*if*) and logical conclusions (*then*).
- **Restrictions** set the requirements for an individual to be member of a class.
- **Functions** represent special cases of relation.
- **Instances** represent specific individuals or elements.

1.2.3. USAGE AND APPLICATION

Ontologies is becoming of great importance due to the necessity of communication and information exchange among humans, organizations and software systems that aims to enhance knowledge sharing and reuse.

Ontologies have a wide range of use in various subjects. Literature is rich in descriptions of ontologies and their intended purposes.

Uschold & Gruninger [9] classified the uses of ontologies in three main categories: *Communication*, *Interoperability* and *Systems Engineering*.

a. Communication

Ontologies reduce conceptual and terminological confusion by providing a unified framework for describing the concepts of the world and their relations. Thus, they improve

communication and cooperation between people and organizations with different needs and viewpoints.

b. Interoperability

With the increasing popularity of computer networks, where different users are exchanging information or using different software, the issue of interoperability has become major. Ontologies can be used as inter-lingua, accomplishing the translation on terms with the same meaning between the parties of a communication system. In the area of databases, ontologies will deliver important services such as database interoperability, cross database search and integration with web services.

c. Systems Engineering

In the area of systems engineering, ontology brings benefits in the following subjects:

- **Reusability**, by providing a shared vocabulary of terms and library of concepts within a domain.
- **Reliability**, since a formal conceptualization helps in consistency checking of software systems with respect to the specifications.
- **Specification**, by assisting software engineers with a shared universal understanding of the problem.

According to *Sure & Studer* [10], successful applications of ontologies in numerous fields are:

- Natural Language Processing and Machine Translation
- Knowledge Engineering
- Knowledge Management
- Engineering Disciplines
- Electronic Commerce
- Information Retrieval and Information Integration
- Web Catalogs
- Intelligent Search Engines
- Digital Libraries

- Enhanced User Interfaces
- Software Agents
- Business Process Modeling

Furthermore, ontologies play significant role in the context of *Semantic Web* as well as in *Education*, two aspects that will be discussed below.

1.2.4. Types of Ontologies

In the literature can be found several classification approaches of ontologies according to their features and functionality.

Lassila & McGuinness categorized ontologies in a linear spectrum (*Figure 2*) based on the richness of their internal structure [11]:

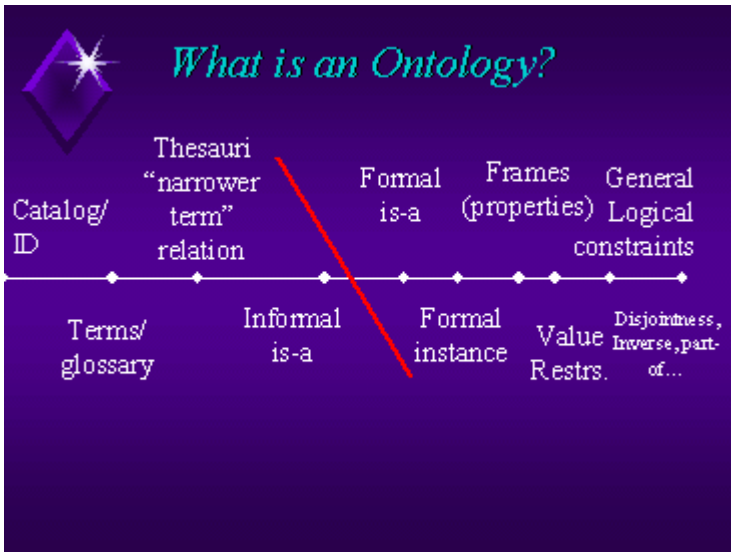


Figure 2: Ontologies categorization by *Lassila & McGuinness* [11]

- **Controlled vocabularies** are one of the simplest notions of ontology, like a catalog for example.
- **Glossaries** are another simple notion of ontology for creating a list of terms and meanings where meanings are expressed as natural language statements.
- **Thesauri** provide some additional semantics for the relations between terms such as synonyms.
- **Informal is-a hierarchies** refer to hierarchies where a more specific class is not always and in any case also an instance of a more general class.

- **Formal is-a hierarchies** include strict sub-class relationships where if A is a superclass of B, then an instance of B is certainly always instance of A as well.
- **Formal is-a hierarchies that include instances of the domain** refer to classification schemes that include class names and ground individual content.
- **Frames** (**properties**) involve class property information with strict class hierarchy and instance relationships.
- **Ontologies that express value restriction** in filling class properties – i.e. the value of property 'age' can be a number between 0 and 130.
- **Ontologies that express general logical constraints** such as disjointness of classes or inverse relationships – i.e. an instance of class A cannot simultaneously be instance of B if A and B are disjoint classes.

Guarino distinguished four different kinds of ontologies according to their level of generality, as shown in *Figure 3* [12]:

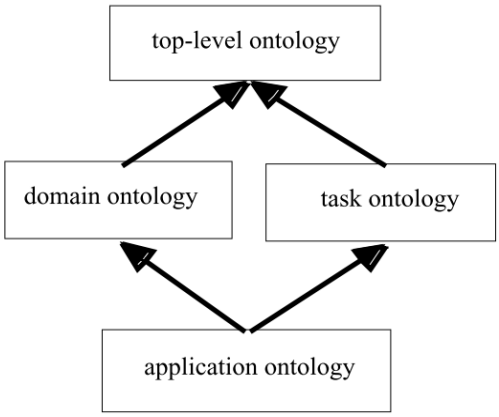


Figure 3: The four kinds of ontologies by *Guarino* [12]

- **Top-level ontologies** describe very general concepts like space, time, events, *etc*.
- **Domain ontologies** describe the vocabulary related to a specific domain (like medicine, cars or wine) by specializing the terms introduced in a top-level ontology.
- **Task ontologies** describe the vocabulary related to a generic task (like selling or eating) by specializing the terms introduced in a top-level ontology.
- **Application ontologies** are a specialization of both domain and task ontologies describing the roles played by domain entities while performing a task.

Gómez-Pérez & Benjamins classified ontologies based on the subject of conceptualization [13]:

- **Knowledge Representation (KR) Ontologies** capture the representation primitives used to formalize knowledge in a specific KR domain.
- **General** or **common Ontologies** represent common sense knowledge and include vocabulary related to very general concepts like space, time, things, events, *etc*.
- **Top-level Ontologies** provide general notions under which root terms in existing ontologies should be linked.
- **Meta-Ontologies** (also called Generic Ontologies or Core Ontologies) are reusable across domains.
- **Domain Ontologies** describe the concepts within a domain and their relationships and are reusable in that specific domain.
- **Task Ontologies** provide vocabulary of terms used in performing a generic task or solving a problem.
- **Linguistic Ontologies** describe linguistic concepts.
- **Domain-Task Ontologies** are task ontologies that cannot be reused in other domains.
- **Method Ontologies** provide vocabulary for the terms used in performing a particular task.
- **Application Ontologies** describe the available knowledge for modeling a specific application.

1.2.5. Ontology Building Languages

Due to the great potential of ontologies and Semantic Web, the development of new languages for building ontologies continues during the last years.

Ontology languages are distinguished in *traditional syntax* (or *logic-based*) *languages* and *markup syntax* (or *web-based*) *languages*. *Logic-based* or *traditional languages* appeared

chronologically first and some of the most known are: *CycL*[2] (first ontology language, used in Cyc Knowledge Base), *Loom*[3] (frame-based language), *KIF*[4] (Knowledge Interchange Format), *Ontolingua*[5] (based on KIF and one of the most expressive languages) and *F-Logic*[6] (frame and first-logic language with a variety of other languages based on it).

The emerge of XML language and its supremacy as universal language for information exchange in the web, has driven to the development of new ontology languages that use a markup scheme based on XML syntax to encode knowledge, such as *XOL*[7] (Ontology Exchange Language), *SHOE*[8] (which was previously based on HTML) and *OML*[9] (Ontology Markup Language).

The rise of *RDF* combined with *RDFS* and its great popularity that has lead to languages built on them, *OIL*, *DAML+OIL* and *OWL*, will be analyzed more in next paragraphs.

Resource Description Framework (RDF)

Noteworthy point in the development of ontology languages was the recommendation of RDF[10] (Resource Description Framework), endorsed by the W3C[11] [14]. RDF is not a language; it is a standard data model for data interchange on the web. Its basic building entity is a statement called semantic triple or just *triple*. A triplet is a set of three entities expressed in the form of *subject-predicate-object* (*Figure 4*), enabling the represented knowledge to be readable from machines.

[2] The Syntax of CycL: www.cyc.com/documentation/ontologists-handbook/cyc-basics/syntax-cycl
[3] Loom Project: www.isi.edu/isd/LOOM
[4] Knowledge Interchange Format (KIF): www.ksl.stanford.edu/knowledge-sharing/kif
[5] Ontolingua: www.ksl.stanford.edu/software/ontolingua
[6] F-Logic (Wikipedia): https://en.wikipedia.org/wiki/F-logic
[7] XOL Ontology Exchange Language: www.ai.sri.com/pkarp/xol/
[8] Simple HTML Ontology Extensions: www.cs.umd.edu/projects/plus/SHOE/
[9] Ontology Markup Language (OML): xml.coverpages.org/oml9808.html
[10] Resource Description Framework (RDF): www.w3.org/RDF
[11] World Wide Web Consortium (W3C): www.w3.org

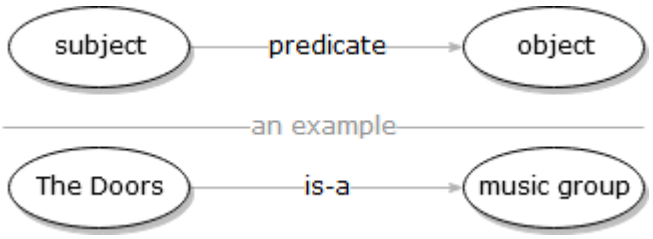


Figure 4: Semantic Triple in RDF

Subsequently, W3C presented the *RDF Schema*[12] (*RDFS*) as an extension of RDF that provides a data modeling vocabulary for RDF data [15]. RDFS provides mechanisms for defining relationships between resources and constructing knowledge models. RDFS is widely used by many tools and projects and set the basis for the Semantic Web. Three more languages have built on RDFS: *OIL*, *DAML+OIL* and *OWL*.

OIL (Ontology Interchange Language), developed in the *On-To-Knowledge* project [16], uses syntax and semantics of existing languages (XML, RDFS) and following frame-based approaches [17]. *DAML+OIL*, developed by a joint US/EU ad hoc committee, combines features of both languages, *DAML*[13] (DARPA Agent Markup Language) and *OIL* [18].

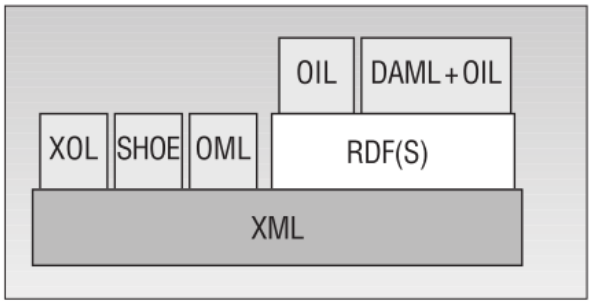


Figure 5: Languages stack in the Semantic Web (source: *Gómez-Pérez & Corcho* [19])

Web Ontology Language (OWL)

In 2001, W3C formed the *Web-Ontology Working Group* whose purpose was to develop a new markup language for Semantic Web. In 2004, *OWL* became a formal W3C recommendation. In 2007, W3C's OWL Working Group, started to extend OWL and adding new features. The new version of OWL, called *OWL 2*, announced by W3C in 2009.

[12] RDF Schema (Wikipedia): https://en.wikipedia.org/wiki/RDF_Schema
[13] DARPA Agent Markup Language (Wikipedia): https://en.wikipedia.org/wiki/DARPA_Agent_Markup_Language

OWL has built upon RDFS and is part of W3C Semantic Web language stack, as shown in *Figure 6*. OWL facilitates greater machine interpretability of Web content by providing richer vocabulary for describing classes, their relationships and their properties, along with formal semantics for knowledge modeling.

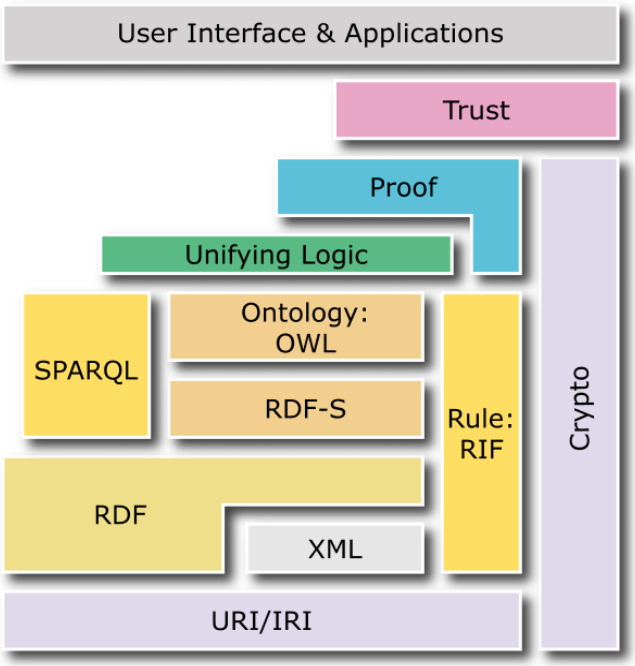


Figure 6: Semantic Web Layer Cake with languages (source: W3C[14])

OWL contains three, increasingly expressive, sublanguages: *OWL Lite*, *OWL DL* and *OWL Full*.

- **OWL Lite** is a subset of OWL-DL offering simpler and easier implementation. OWL Lite supports primarily classification hierarchy and simple constraints.
- **OWL DL** is a restricted set of OWL Full that supports maximum expressiveness while retaining computational completeness.
- **OWL Full** provides maximum expressiveness and the syntactic freedom of RDF but without computational completeness guarantee.

[14] Semantic Web, and Other Technologies to Watch: www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(24)

It must be noted that only OWL Full supposed to be an extension of RDF while the others not. OWL is now the most prominent ontology language for publishing and sharing data with ontologies on the Internet.

1.2.6. TOOLS

Alongside with the methodologies and languages, there have been developed a significant number of ontology tools suitable for knowledge domain modeling, ontology visualization, ontology merging and other modeling tasks. Many of the tools have been developed for a particular project or by a University. Some of the tools were no longer available after the project has finished while others are being improved and new tools are presented[15].

Cardoso & Escórcio in [20], made an overview of some ontology editing tools, presented their features and capabilities with comparison to each other. The tools examined and compared for their support in the most important features that a tool is considered to provide for ontology engineering:

- Robust and integrated environment that support to most of the activities involved in the ontology development process
- Free and open-source software
- Support of RDF, RDFS and OWL
- Collaborative environment
- Provide multiple ontology environment
- Offer server-based environment with support for consistency checking
- Visual creation and editing
- Query builder
- Support a methodology
- Support large scale ontologies
- Versioning
- Interoperability

[15] Ontology editors (Wikipedia): www.w3.org/wiki/Ontology_editors

- Reasoner
- Graphical view
- Plug-ins
- Tutorial and user support

Researchers presented the chosen tools and analyzed their features and capabilities based on the set of features above. The tools that have been chosen are the following:

- Protégé
- OntoEdit
- DOE
- IsaViz
- Ontolingua
- Altova SemanticWorks 2006
- OilEd
- WebODE
- pOWL
- SWOOP

Below we describe Protégé, the ontology editor that used in this thesis due to its many advantages which described in next paragraph. More information about the remainder tools can be found in [20].

Protégé

Protégé Ontology Editor is a free open-source ontology editor, developed by Stanford University [21]; Protégé Desktop is now in version 5.1.0 while there is also the *WebProtégé*[16] that offers collaborative viewing and ontology editing. Protégé provides a rich, integrated environment that gives support to a wide range of activities involved in ontology development process, such as conceptualization, reasoning, exchange, migration, *etc* [22]. It supports the most popular ontology languages of Semantic Web and follows the RDF specifications by W3C.

The user interface of Protégé is simple and intuitive while it offers advanced options and tools in building simple or complicate ontologies. The user can build a new ontology, import or merge existing ontologies and save in different ontology file formats (rdf, owl, *etc*). In

[16] WebProtégé : https://protege.stanford.edu/products.php#web-protege

addition, the support and availability of a wide range of plug-ins, gives the ability to augment its usability and expand its capabilities [17,18][23].

Furthermore, Protégé 5 comes with the *HermiT* OWL reasoner pre-installed. *HermiT* is an OWL reasoner that supports all OWL 2 features and also performs object and data type classification [24]. It also claims to be the fastest and more consistent OWL reasoner available. A reasoner is an inference engine that provides classification services which include consistency tests, satisfiability tests and subsumption tests [25]. So, the user can check the ontology for consistency errors, identify subsumption relationships between classes, retrieve instances or classes of a given, anonymous class expression, or even to answer a conjunctive query.

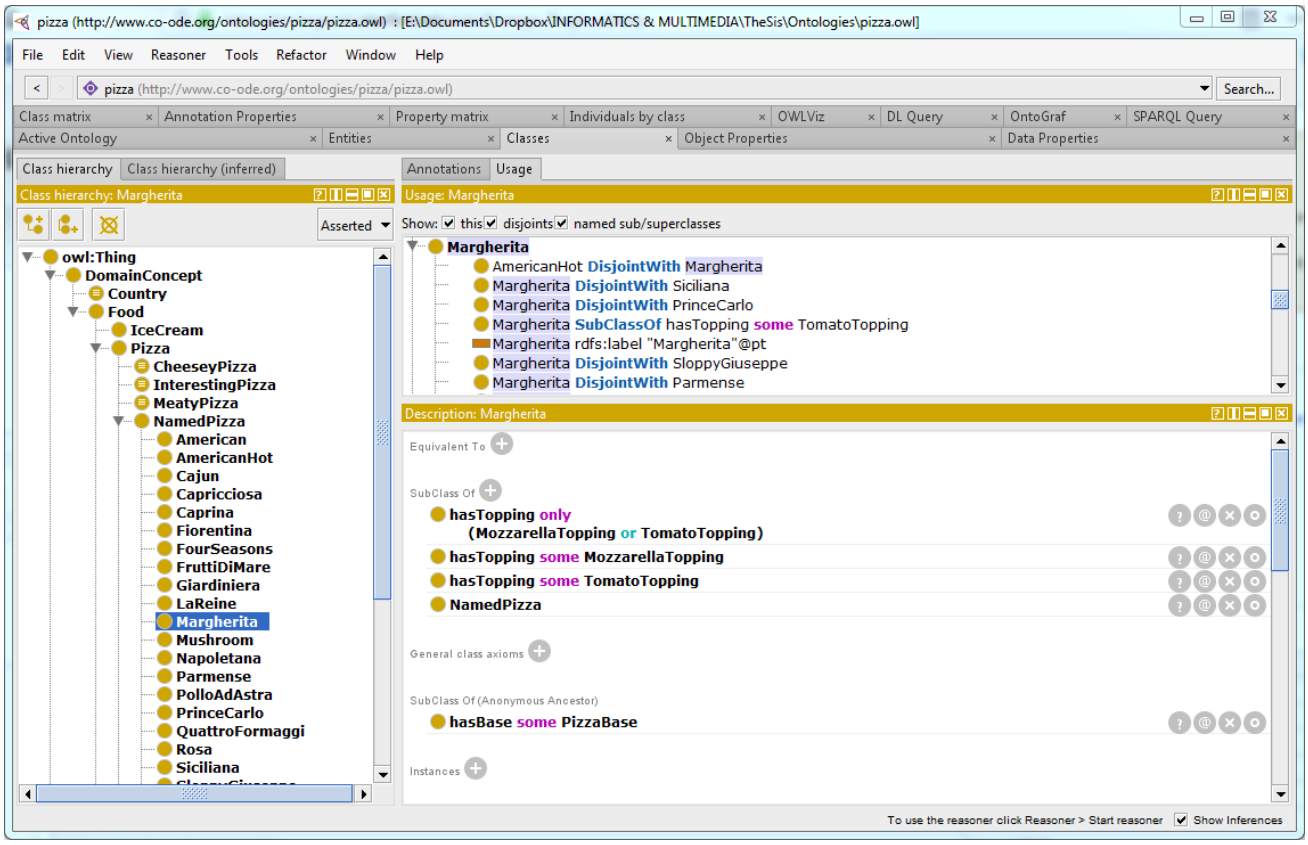


Figure 7: The environment of Protégé 5

[17] Protégé Plugin Library: https://protegewiki.stanford.edu/wiki/Protege_Plugin_Library
[18] Protégé Visualization Plugin Library: https://protegewiki.stanford.edu/wiki/Visualization

Protégé is the most widely used tool for ontology development, as shown in *Figure 8*. One of the subjects that *Cardoso* studied in his survey about Semantic Web, was the usage and popularity of ontology editors [26]. The results clearly pointed out that Protégé is by far the most popular ontology editor. In the same survey, the great impact of OWL in Semantic Web is also pointed out without doubt (*Figure 9*).

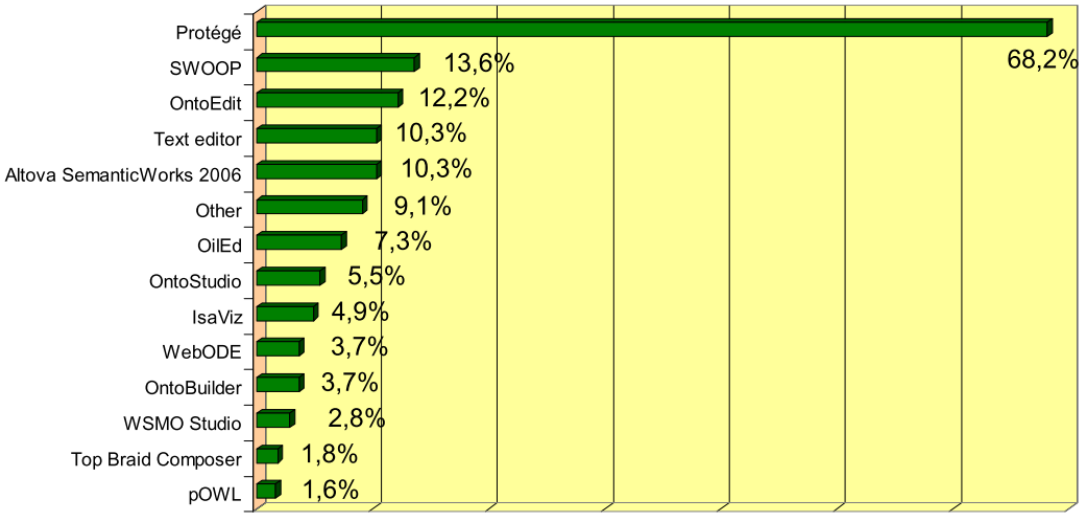


Figure 8: Usage of ontology editors (source: *Cardoso* [26])

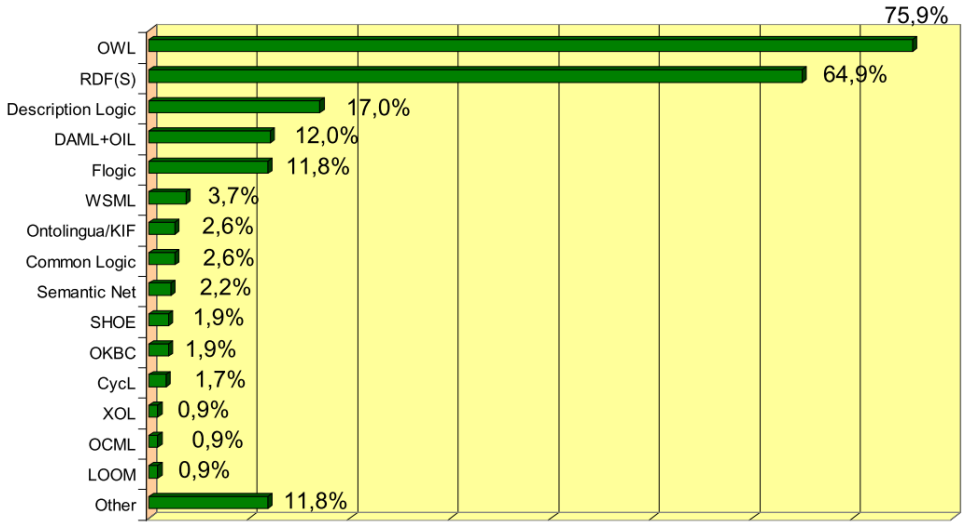


Figure 9: Ontology languages used (source: *Cardoso* [26])

CHAPTER 2. REVIEW OF LITERATURE

With the evolution of Semantic Web, researchers in the area of education envisioned a promising future for education assisted by semantic web technology. In this chapter, a literature review is performed with focus in ontologies that developed for educational purposes. Due to their large number and wide range, the related academic and research contributions are classified in categories and then presented and analyzed. Because of our special interest in the domains of *Curriculum* and *Course*, the relevant works are analyzed further in an extra, special paragraph.

2.1 ONTOLOGIES IN EDUCATION

In spite the fact that the field of applying ontological technology in education is fairly young, research has shown that ontology-based technologies and applications offer powerful benefits in education and become one of the most fashioned fields of research in educational technology.

Ontologies and Semantic Web provide a variety of applications in education domain. A growing number of papers and articles are presented in related workshops, conferences, journals, and books. As a result, this promising field tends to become quite broad and fuzzy. One of the most notable projects implemented by *Dicheva* and her colleagues in [27], who classified the available information of research in the domain. They presented an ontological overview of the *Ontologies for Education* field and a report on the development of an ontology-driven web portal for such ontologies. Their *O4E* (*Ontologies for Education*) *Web Portal* provides a single place where anyone interested (students, instructors, researchers and experts) can find available research information, projects and good practices in this field. The researchers divide the whole field according to the role ontologies play in the research, resulting in two top-level meta-concepts: *Building Ontologies for Education* and *Using Ontologies in Education*. Speaking about the *Using Ontologies in Education* field, they tried to classify it from two perspectives, *technological* and *application*, depending on the technology implemented and the role of the ontology in a project (*Figure 10*).

The classification of *Using Ontologies in Education* research topics according to *Dicheva et al.* [27]:

- **Technological perspective**
  - *Knowledge representation technology* contains the topics of knowledge organization, knowledge inference and planning
  - *Information retrieval* contains the topics of information visualization, navigation and querying on information
  - *Semantic Web technology* which contains the topics of *semantic annotation of learning resources* and *knowledge reuse and interoperability*.
- **Application perspective**
  - *Cognitive tool* for the construction, externalization, communication and assessment of knowledge.
  - *Type of knowledge* for describing knowledge about subject domain, educational system architecture or instructional knowledge.

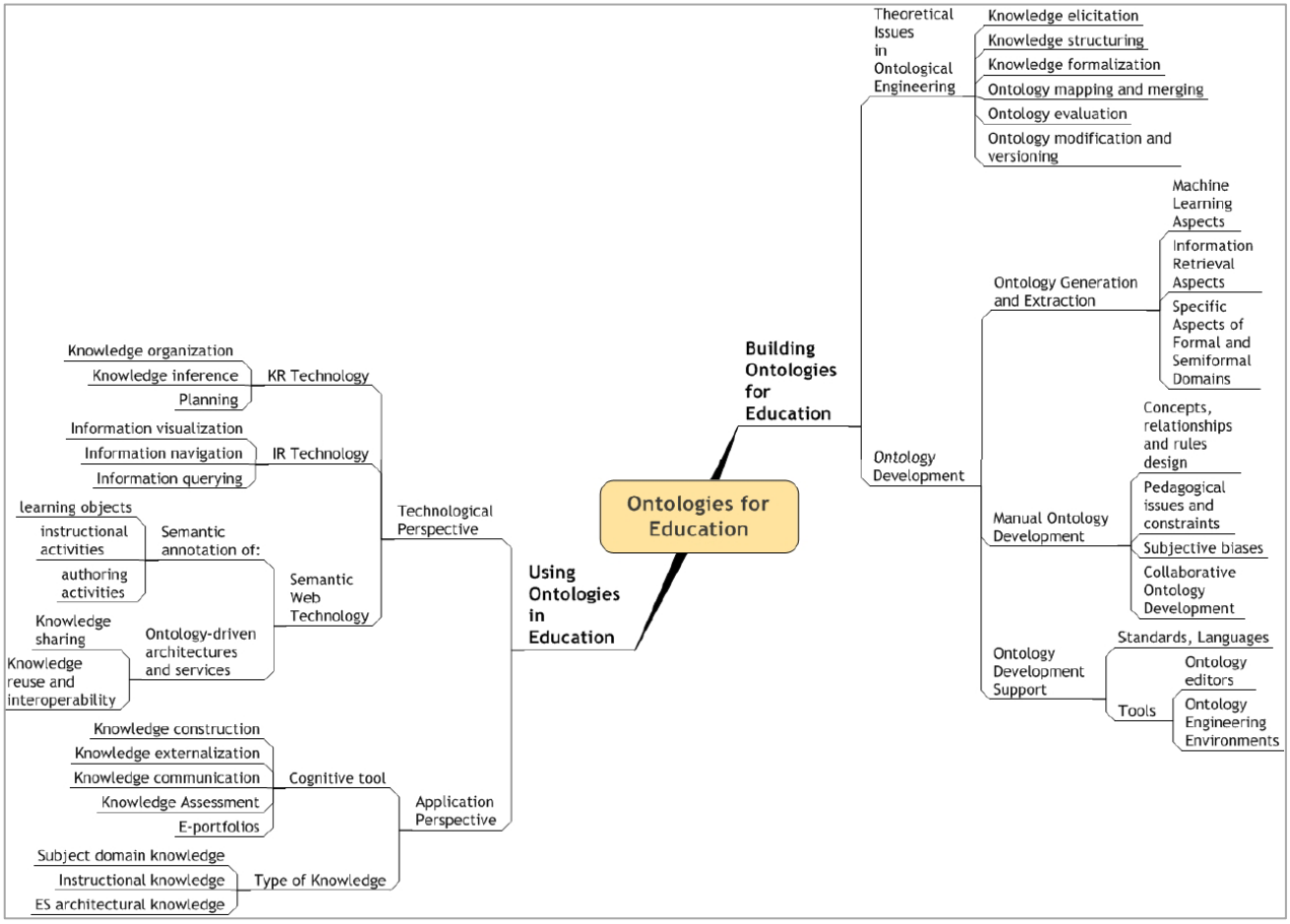


Figure 10: Ontology of Ontological Technologies in Education (source: *Dicheva et al.*[27])

The classification of the related works that studied for the purpose of this thesis, based in five main categories:

- **Knowledge Domain Ontologies**
- **Learning Tasks Ontologies**
- **Competency Ontologies**
- **Educational Metadata Ontologies**
- **Ontologies for e-Learning Applications**

The above classification and usage of ontologies is neither exhaustive nor exclusive. Many ontological models and applications that have studied are based on ontologies that can be classified to more than one of the above categories. Besides, different approaches have been found as well as in literature and in practical ontology development.

2.1.1. KNOWLEDGE DOMAIN ONTOLOGIES

Knowledge Domain Ontologies (or just Domain Ontologies) for educational purposes used for semantic annotation of educational content such as learning objects, instructional activities and other learning resources. Semantic annotation enables a framework for common knowledge base for share and reuse learning resources on a global scale. Furthermore, domain ontologies can be very useful for knowledge organization, automated knowledge inference, information visualization and navigation, information querying and other activities in the domain field.

A growing number of ontologies for representing the knowledge in a domain have been developed; some of the most important works are described below.

Some researchers, i.e. *Grandbastien & Huyinh Kim Bang* [28], *Van Assche* [29] and *Paquette* [30], described frameworks where domain ontologies used for annotating learning resources in order to enable the retrieval, reuse and interoperability of such resources. *Alberdi & Sleeman* [31] designed the ReTAX framework for the taxonomic revision in biological domain with an automated way.

Sosnovsky & Gavrilova [32] designed ontology for teaching/learning C programming to undergraduate students but it can be used in e-learning systems too.

Bianchi & Mastrodonato [33] introduced the use of Semantic Web services within AquaRing project, an EC-funded project that addresses the access and use to a large amount of knowledge and information on aquatic environment which collected from different resources all over Europe. In this project, there are seven ontologies covering different aspects of the domain that used for content annotation. *AquaRing ontology* integrates the seven ontologies and used for semantic services to the *AquaRing* portal.

In [34], *Oprea* supported that ontologies for University courses, in order to support ontology sharing and reusing, should include general terms suitable to any course as well as specific terms for the knowledge domain. The appropriate specific terms are added in the ontology and suitable properties are attached for a specific course that the ontology intended to be used. In the case of the designed ontology for a university e-Learning course about Artificial Intelligence, named *Univ_Edu_Onto* [34], the vocabulary includes general terms (such as *discipline*, *curriculum*, *objectives*, *number_of_credits*, *etc*) and specific terms (such as *knowledge*, *inference*, *heuristic*, *cognitive_system*, *etc*).

Mesaric & Dukic [35] presented the construction of domain ontologies for Economics curricula in Croatia. They used common vocabularies like SUMO [36] and SENSUS [37] and followed multilevel approach to dictionary creation. The resulted thesaurus is composed of four layers: (a) *Epistemic level* (general terms and relations), (b) *Domain level* (information structures specified for the domain), (c) *Level of Usage* (specific segments in domain) and (d) *Project level* (specific implementations). *Curriculum* can be developed in *Domain level* while *Courses* could be defined in the level of *Usage* and *Learning Objects* in the *Project level*.

An ontological approach for the conceptualization of knowledge structures in higher education presented by *Chung & Kim* in [38]. The designed learning ontology combines different kinds of educational entities organized on a layered structure. The integrated ontology consists of four sub-ontologies, *Curriculum ontology*, *Syllabus ontology*, *Subject ontology* and *Resource ontology*. Upper-level ontology is connected with lower-level ontology with one-to-many relationship. This ontology is also included in the set of existing ontologies close to our ontology and is further analyzed later in chapter 3.

2.1.2. LEARNING TASKS ONTOLOGIES

Apart from knowledge domain modeling, ontologies can be used for the semantic annotation and modeling of learning tasks and educational content, such as instructional activities, learning units, learning scenarios, learning paths, *etc*. The construction of such ontologies take into account well-known standards for learning planning in order to describe the learning entities, relations and learning paths of the learning process. The use of ontologies can result in more effective automatic or semi-automatic tools and services that increase the level of reusability, personalization and adaptation in Learning Management Systems (LMS).

Mizoguchi et al. [39] discussed how a task ontology contributes to characterizing Intelligent Educational Systems (IES). They designed a task ontology that describes concepts playing important role in IES and their preliminary taxonomy. The top-level concepts in the ontology are: (a) *Goals of education*, (b) *Learner's state*, (c) *System's functionality*, (d) *Learner-System interaction*, (e) *Teaching material knowledge*.

In [40], *Amorim et al.* presented a learning design ontology that describes all the concepts and their role in the design of a teaching-learning process based on the IMS-LD standard[19]. An ontology-based framework for bridging content about learning design and learning objects described by *Knight et al.* [41].

The use of ontology for annotation of learning scenarios proposed by *Rius et al.* in [42]. The designed ontology can contribute in the specification of existing learning scenarios as well as the creation and validation of new ones. *Barros et al.* [43] presented an ontology that integrates the learning elements and their relations in a collaborative environment that can be used to define new collaborative learning scenarios.

[19] IMS Learning Design Specification: www.imsglobal.org/learningdesign

2.1.3. COMPETENCY ONTOLOGIES

Although it has been in use for decades, the concept of *competence* has enjoyed increasing currency in the last few years in educational research and empirical studies, dealing with the development of human resources and the productivity of education. The globalization of labor and the international competition have resulted in growing requirements for knowledge and skills. Hence, the productivity of educational systems has become even more crucial for society, upgrading the role of *'outputs'* and *'outcomes'* at all levels of educational systems, from elementary and higher education up to vocational and adult education.

A useful definition for *competence* in educational context mentions[20]:

> "*A cluster of related abilities, commitments, knowledge, and skills that enable a person (or an organization) to act effectively in a job or situation. Competence indicates sufficiency of knowledge and skills that enable someone to act in a wide variety of situations. Because each level of responsibility has its own requirements, competence can occur in any period of a person's life or at any stage of his or her career*."

Research has shown the significant role of *competence modeling* for any kind of pedagogical planning about learning and training courses based on e-learning systems [30], [44], [45], [46].

According to *Sicilia* [46], ontologies offer the most functioning infrastructure for describing a knowledge domain and enabling competence management in intelligent knowledge management systems. In [47], *Vas* introduced an ontology-based model that connects knowledge and outcomes of a study program with employment opportunities.

To sum up, ontology-based models could establish integration of competency ontologies in learning software frameworks as an instructional tool, increasing the exchange and reusability of learning objectives, linking competencies with learning resources, enabling the

[20] Competence, definition in BusinessDictionary: www.businessdictionary.com/definition/competence.html

management of learner's knowledge profile and leading to higher competency-oriented achievements [29], [30], [45].

2.1.4. EDUCATIONAL METADATA ONTOLOGIES

The classic literal definition says that *Metadata* is "*data about data"*, means it is the data that provides information about various aspects of the data we interested in[21]. A more descriptive definition used by NISO [48] describes metadata as:

> "*Structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource*"

As far as the educational domain is concerned, there are standards and specifications for metadata relating to resources that can be used in learning process. An overview of existing standards and developments about metadata for learning materials has been carried out by *Barker & Campbell* [49].

Based on these standards, *ontologies for educational metadata* semantically annotates learning material by means of ontology terms, enabling that way the interoperability, exchangeability and reusability of learning resources among applications. Some approaches for describing educational content and learning material are coming next.

Valaski and her colleagues have tested the use of ontologies for automatic classification and recommendation of learning objects according to user's learning styles [50], [51], [52]. *Lee et al.* has shown the functionality of ontologies for organizing learning material of Java courses in an adaptive learning environment by developing the *JLOO* (Java Learning Object Ontology) [53].

The *educational ontology* that developed by *Bianchi & Mastrodonato* [33] within the *AquaRing* project has as central concept the learning *Resource* in a scheme with five main classes: (a) *Resource* (the learning resource), (b) *Resource_Feature* (resource's main

[21] Metadata (Wikipedia): en.wikipedia.org/wiki/Metadata

educational characteristics), (c) *Context* (the educational level/environment), (d) *Objective* (the cognitive learning outcomes) and (e) *User* (the target group).

Koutsomitropoulos and his colleagues [54], [55] introduced two ontologies that describe educational content and used for delivering Semantic Web services over educational resources in digital repositories and digital libraries systems. An ontology-based management system for DSpace repository - follows the Dublin Core (DC) metadata schema – is constructed in [54], which used in the University of Patras. Furthermore, because DC schema sometimes lacks of efficient characterization of educational metadata, authors implemented the mapping of some LOM elements that were not available in DC-Terms but were essential in their case. For more efficiently characterization of learning objects, the researchers developed a LOM ontology [55]. In [56], *Verbert & Duval* presented *ALOCOM*, a generic content model for learning objects that aims to align existing learning objects in order to enhance their interoperability. In the context of primary and secondary education, an ontological modeling of learning resources for teaching courses for Informatics in school is proposed in [57].

2.1.5. Ontologies for e-Learning Applications

Ontology technology is considered to play pivotal role in supporting intelligent learning systems [58], [59], [60], [61]. Apart from ontologies mentioned before, there are ontologies for describing the basic components and architecture of Learning Management System (*LMS*), enabling the exchangeability and reusability of educational content among Intelligent Instructional Systems (*IIS*) as well as the scalability and interoperability of these systems [58].

Several researchers support the fact that Learning Management Systems can benefit from semantic technologies and ontologies in order to provide personalized adaptive learning experience to users. *Kerkiri & Manitsaris* [62] recommended methods on how ontologies can facilitate e-learning systems to achieve personalization by using ontologies for handling collected metadata of the user's preferences and monitoring their learning progress. *Chung & Kim* [63] and *Gascueña, Fernández-Caballero & González* [64] also suggest the use of ontologies in e-learning environments for dealing with learning material and designing

learning paths as well as supporting the system to enhance the reusability of resources and the personalization of the learning process.

Despite the continuous progress during the last decades, *Mizoguchi & Bourdeau* [58] exposed common issues that need to be addressed in the research area of AI-ED (Artificial Intelligence and Education) and discussed directions to overcoming them. They analyzed the status and challenges in the four underlying concepts of *Intelligence*, *Conceptualization*, *Standardization* and *Theory-awareness* within the AI-ED research which consists of the three main research areas: *Artificial Intelligence* (AI), *Instructional Design Science* (IDS) and *Learning Science* (LS). Subsequently, they discussed about the importance of ontology-awareness in AI-ED research and how ontological engineering [65] can help to deal with such problems and the significant role that ontology-based architecture can play in authoring and tutoring environments.

Motivated by the tremendous growth of interest about *Semantic Web*, *Al-Yahya, George & Alfaries* [66], developed a comprehensive review of the literature about the use of semantic technologies and development of ontologies in e-Learning domain. They surveyed, analyzed and classified key contributions related to the current usage and promising future of ontologies in the field of e-Learning. *Figure 11* and *Figure 12* depict the increasing number of published papers relevant to the terms *'Ontologies'* and *'e-Learning'*.

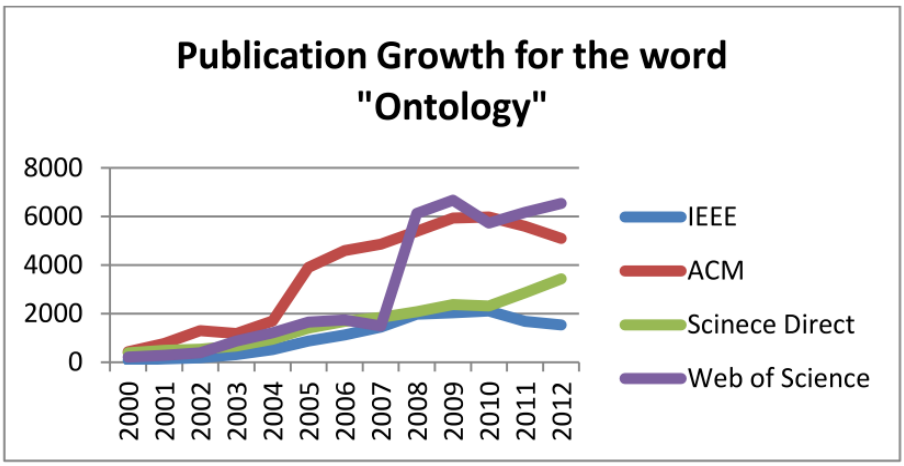


Figure 11: Publication Growth for the word *'Ontology'* (source: *Al-Yahya et al.* [66])

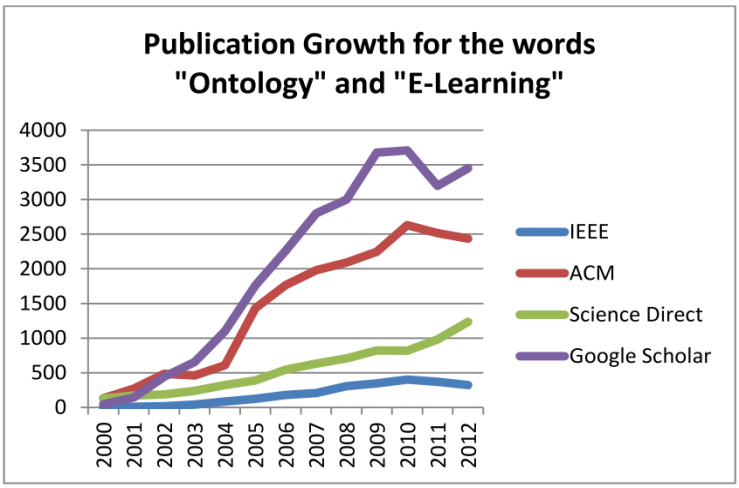


Figure 12: Publication Growth for the words 'Ontology' and *'e-Learning'*(source: *Al-Yahya et al.* [66])

Among several classification methods that presented in the literature, authors chose one that is based on the subject of the conceptualization [67] and classified ontologies in four categories: (i) *Domain-Task ontologies*, (ii) *Domain ontologies*, (iii) *Task ontologies* and (iv) *Application ontologies*. They reported that the majority of surveyed ontologies were classified as *Task ontologies*, with the summation to be over five times higher than in other three types.

In the context of e-Learning, authors used as key characteristic the e-Learning tasks that the ontology serves and classified the literature in this area as follows:

- *Curriculum Modeling and Management* where important curriculum processes may benefit from the semantic description of curriculum elements.
- *Describing Learning Domains* where two main groups exist, ontologies that describe *Subject domain* or either *Learning task.*
- *Describing learner data* for assessment and personalization.
- *Describing e-Learning services* as a shared vocabulary for interoperability and exchangeability.

Personalization of learning is pointed out as the most popular between these, due the modern tendencies of pedagogy, curriculum and learning environments that express the challenge of adaptation to learner needs and requirements (*Figure 13*).

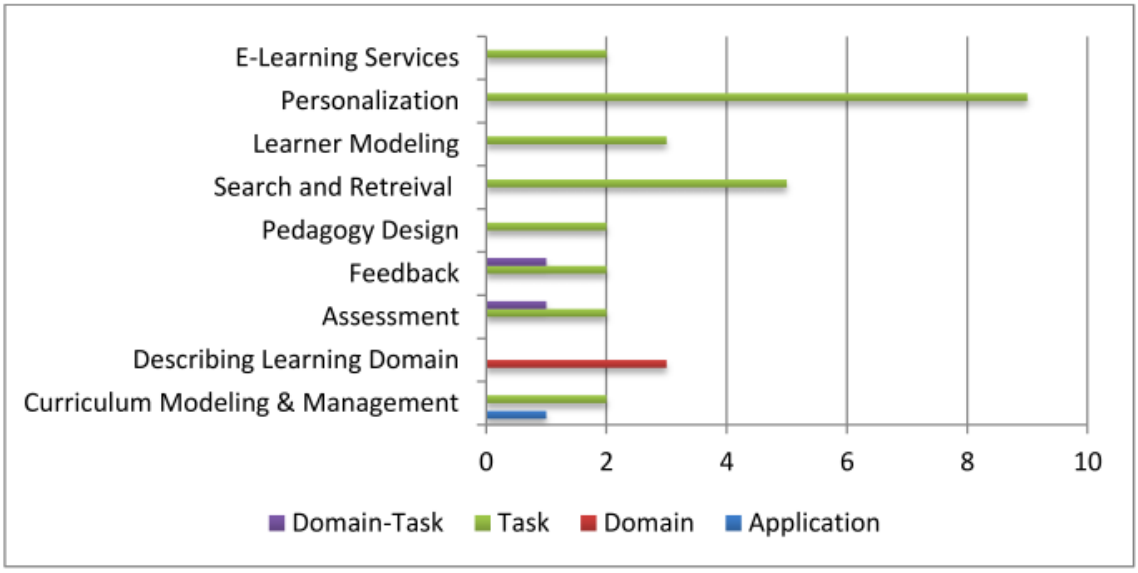


Figure 13: Ontology Classification according to E-Learning Task (source: *Al-Yahya et al.* [66])

A concept-based, courseware-authoring, web environment, with layered ontological support, is presented by *Aroyo, Dicheva & Cristea* [60]. The proposed architecture is based on three layers, *Course ontology*, *Subject domain ontology* and *Educational Metadata layer*. Its main target is to support authoring tasks of different instructors in collaborative building and cooperative reuse of authoring content and products. Ontologies are used for conceptualization of the course content and domain terminology allowing intelligent assistance of courseware authors.

Adorni, Battigelli & Brondo [68] introduced two Intelligent Instructors Systems, CADDIE and IWT, developed within research projects, that both count on ontologies and semantic technology in order to support instructional design and learning tasks.

2.2 CURRICULUM AND COURSE ONTOLOGIES

Turning the focus in the domains of *Curriculum* and *Course*, ontology engineering and Semantic Web technologies can play significant role in key curriculum tasks and course activities.

In this paragraph, we survey and analyze some notable works related to the subjects of Curriculum and Course in an academic setting.

2.2.1. CURRICULUM ONTOLOGIES

Curriculum is considered as top-level information of educational entities of an educational organization. It contains the list of courses of an educational period, while the sequencing denotes the appropriate order that these learning units need to be taught in order learners to be able to cover the subject areas and accomplish the learning goals [69].

The application of ontologies and semantic web in the context of curriculum can help fostering the transparency, convergence, collaboration, exchangeability and interoperability within the area of higher education [70].

Curriculum management and development can be improved through the use of ontologies in curriculum tasks like aligning, classification, comparing, and matching between universities, national educational systems or relevant disciplines. Semantic annotation and indexing of curriculum items help the administration in monitoring and managing the educational offerings of an institution, revealing useful information like course overlapping, uncovered or less covered areas, possible synergies between courses, *etc* [38], [71].

Competence management and human resources management in education can also benefit by supporting services, based on ontologies, that can facilitate student mobility, degree comparability, credits transferability, competitiveness, accreditation, evaluation, *etc* [44], [70], [72].

Furthermore, the development of ontology-based e-learning systems enhances the building of personalized and adaptive curriculum. Personalization of curriculum based on adaptive learning paths is the most challenging pedagogical service nowadays, as shown before in *Figure 13*. Learning Management Systems, supported by ontologies, are fostered to enable

flexibility and adaptability in curriculum and course construction process. This enhances adaptive and personalized learning, tailored to learner's needs and requirements. Whether the e-Learning system dynamically sequencing the appropriate learning objects and constructing learning paths [42], [69], [73] or the involvement of learners and teachers is allowed [63], the contribution of ontologies is crucial.

The Bowlogna Ontology

'*Bologna Declaration*' is a joint declaration of the European Ministers of Education convened in Bologna 1999, which aims to establish and promote worldwide the European system of higher education [74]. Its main purpose is to unify curricula and research across European universities in order to increase the comparability and competitiveness of European higher education as well as to foster the mobility of students and employees across Europe. Therefore, it proposes a number of policies and processes that need to attained such as the adoption of a system of degrees easily readable and comparable, the establishment of a system of credits and accreditation such as ECTS (*European Credit Transfer System*), the co-operation in quality assurance and the elimination of obstacles to free mobility [74].

Nowadays, after more than fifteen years, the Bologna process is well under way with many objectives already have been consolidated and helped in the standardization and interconnection of heterogeneous higher education systems across Europe. However, as stated by *Demartini, Enchev, Gapany & Cudre-Mauroux* in [75], there is still a long way to go until we have an integrated and coherent European higher education system. For example, although the consolidation of ECTS helps in certificate's recognition and student exchange across universities, it lacks in cohesion and automation.

Demartini and his colleagues [75] tried to address that issue by creating the *Bowlogna Ontology*, a standard and reusable ontology that models an academic setting as described in *'Bologna Declaration'*. *Bowlogna Ontology* aims to improve the standardization, communication and collaboration among universities across Europe.

In the design of the ontology, authors used transformation methods rather than standard ontological engineering methods [76]. They reviewed a linguistic lexicon derived from the Bologna reform and extracted all essential concepts and relations, which used to design the

ontology model. Finally, this resulted in an ontology that describes all important concepts involved in a Bologna academic environment, such as *students*, *professors*, *courses*, *study tracking*, *evaluations* and *ECTS credits*, among other elements (66 classes total).

Researchers provided the ontology with a multi-lingual character by defining all its concepts in four languages: English, French, Italian, and German. It is noted that it can be easily extended and cover even more European languages.

Another important feature is that the ontology can divided in two main sections, according to the information stored. The first one is the public section, where the information is available publicly and to other universities, such as the ECTS credits of a course. On the other hand, information that is required to be private (such as students' grades in exams) is stored in the private section of the ontology. The private section will be accessible only by certified users. However, such kind of security and access control mechanisms can be incorporated only with the contribution of applications, not by the ontology itself.

BBC Curricula Ontology

In 2013, BBC published the *Curriculum Ontology* describing the national curricula within the United Kingdom [77]. The ontology provides a core data model of the national curricula across the UK that also improves the organization and discovery of learning resources via the national curricula. The data model represented in a 3-dimensional space corresponding to the three key concepts which are the *Level*, *Topic* and *Field of Study*, as shown in *Figure 14*.

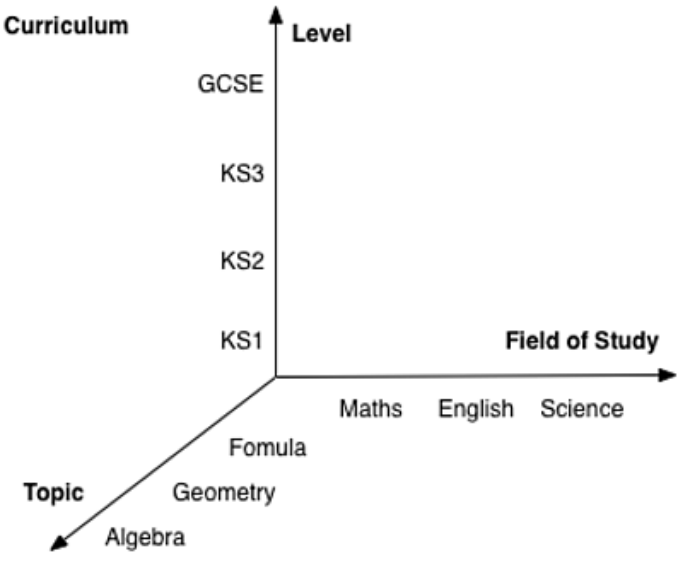


Figure 14: The 3-D model of BBC Curriculum Ontology (source: *BBC* [77])

Level expresses the stage of study (grade of school), *Field of Study* refers to the disciplines of a curriculum (e.g. Math) and *Topic* describes educational subjects more specifically than the Field of Study (e.g. Algebra in Math). Atomic and elementary units of a curriculum are defined by *TopicofStudy* which is the combination of the three dimensions of the data model.

The ontology also provides a vocabulary combined with mappings to Schema.org educational vocabulary. Top-level classes of Curriculum Ontology are sub-classing the Schema.org AlignmentObject[22], as shown in *Figure 15*. AlignmentObject describes an alignment between a learning resource and a node in an educational framework. The use of vocabularies enables easier discovery of learning resources by search engines, agents and institutions.

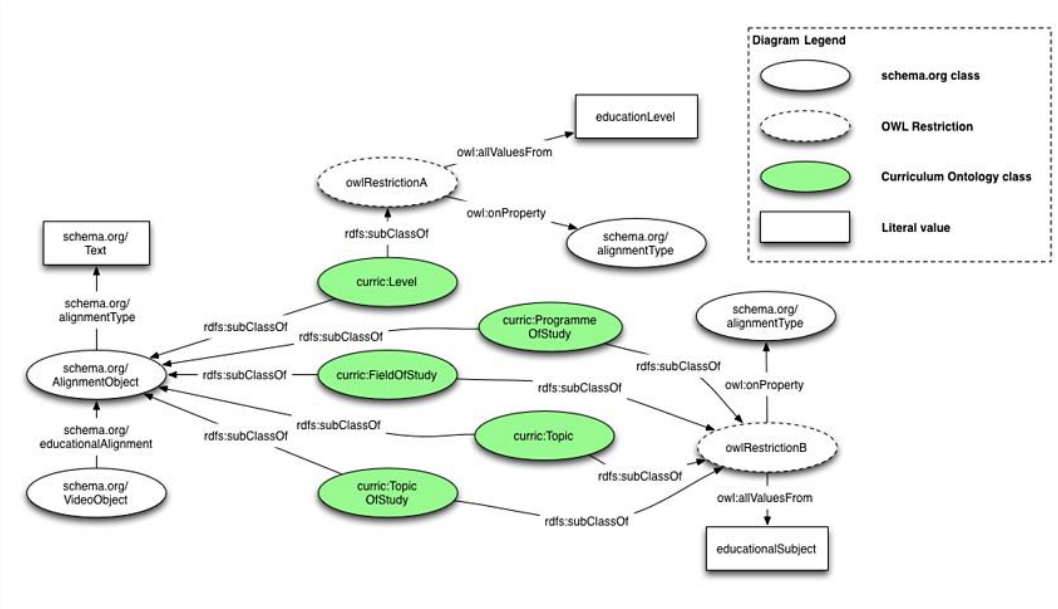


Figure 15: BBC Curriculum Ontology mappings to Schema.org AlignmentObject (source: *BBC* [77])

2.2.2. COURSE ONTOLOGIES

Course in education is a unit of teaching, usually in one subject, offered by an educational institute or organization to a group of students and led by one or more instructors. A course typically lasts one academic term and leads to a course degree or diploma.

[22] Schema.org AlignmentObject: http://schema.org/AlignmentObject

A list of courses with their sequence forms a university curriculum and outlines the subject areas that covered and the learning route that students can follow. Course planning is a pivotal pedagogical service. Two important structural tasks of *course planning* are *course scheduling* and *course sequencing* [69]. Course syllabus takes into consideration the need for course scheduling and determining useful information about the course such as instructors, learning topics and outcomes, grading and evaluation, learning material, classroom, calendar, *etc.* Course sequencing outlines learning paths for learners in order to accomplish course goals. Learning path defines the teaching units required and the appropriate content along with the order they must be offered. Creating adaptive learning paths that taking into consideration learner needs and preferences is one of the biggest challenges in education [78].

Semantic web technologies and ontology engineering have the prospect to contribute humans and intelligent systems in several course tasks such as [60], [79], :

- Course design and construction
- Course planning
- Development of course content
- Courseware authoring
- Course review and benchmarking
- Course assessment
- *etc*.

The development of structured machine-readable syllabuses support teaching and learning with intelligent services like [38], [80]:

- Syllabus recommendation
- Syllabus versioning
- Syllabus matching and interlinking
- Syllabus-based adaptive learning path
- Syllabus knowledge bases with semantic searching and browsing

In order to support the teaching of university courses, *Oprea* developed a course ontology whose vocabulary includes general terms and specific terms of a course [34]. Ontology's

general terms are suitable for any course, enhancing that way ontology sharing and reusing, while specific terms are course relevant. Based on the set of general terms, the ontology can be adapted to a variety of courses across domains and incorporated in university e-learning platforms. Ontologies can also be used for conceptual courseware authoring, searching and assembling learning objects [32], [53]. Even though these ontologies have a specific course subject, they can be used for sharing and reusing course's knowledge units in most courses in the subject of programming fundamentals. Using ontology for assembling the corresponding learning objects facilitates the adaptability of e-learning environments.

A simple course ontology that annotates courses offered by their university, created by *Ameen, Khan & Rani* in [81]. Ontology built for supporting students to discover and select courses.

Boyce & Pahl [79] underlined the lack of educational ontologies in numerous subjects and a reason for that, is the lack of standards and tools for the development of such ontologies. The contribution of domain experts is regarded as essential in addressing this issue. They stated that, with the establishment of standards and the availability of ontology tools, it would be easier for individuals to develop ontologies in their expertise area. This will increase the number of available ontologies and documents that are machine-processable, driving closer towards the Semantic Web [1]. To empower their proposal, authors presented a comprehensible method for domain experts, rather than ontology engineers, to develop course ontologies. More details about their method are following later, in chapter *Ontology Design*.

Subsequently, authors developed course ontology in order to define a minimal set of relationships that would be sufficient and clear to represent all possible relationships for building course ontologies. To evaluate the transferability of relationships across domains, a second ontology for a different subject from unrelated area has been designed. Finally, the sufficiency and reusability of the defined relationship types, proved very satisfying.

In conclusion, authors have pointed out that a well-designed ontology can be used to describe subjects and courses in one institution as well as being transferable and reusable to other schools too, a fact that gives great importance and research interest for the construction of ontologies in the context of curriculum and course by institutions and individuals.

The lack of ontologies with courseware content is a crucial issue for *Zeng, Zhu & Ding* too [82]. Authors stated that educational institutions should play greater role in the creation of educational Semantic Web. University should be responsible for the initial structure of a top-level ontology for university courses for describing general concepts and relationships within a course. The upper layer of course ontology, models the course's main concepts, namely *Curriculum, KnowledgeArea, BasicConcept, Topic, LearningGoal*. Afterwards, students and teachers of university could manage the development of ontologies for specific courses. For this purpose, in order students gain the necessary skills, an elective course named "*Knowledge Engineering and Ontology*" is suggested to be offered by the university [82].

In [63], *Chung & Kim* presented an ontology-based e-learning system which supports learners in building their own learning paths. The ontology model consists of three ontologies on separate layers, namely *Curriculum ontology*, *Syllabus ontology* and *Subject ontology*. Each layer includes the entities of that subject. Learners contribute in learning path building by constructing their subject ontologies, which merged with teacher-based in *Subject ontology*. During this process, learners motivated to study deeply the concepts of curriculum, course and syllabus, achieving active learning.

In [38], *Chung & Kim* again, presented an ontological approach for semantic modeling of multilevel knowledge structures in higher education area. The designed ontology follows layered approach to describe different kinds of educational entities. Each of the sub-ontologies includes the conceptualized entities of that layer and is linked to each other with relationships of classes. The integrated ontology consists of four layers as sub-ontologies: (a) *Curriculum* ontology, (b) *Syllabus* ontology, (c) *Subject* ontology and (d) *Resource* ontology.

More details about the ontologies described in [34], [38], [63], [79], [82] as well as other ontologies that are close to the subject of this thesis are following in next chapter *Ontology Design*.

SPECIAL PART
CHAPTER 3. ONTOLOGY DEVELOPMENT

The purpose of this thesis is the semantic annotation of concepts and entities within an academic institution aiming to model the core concepts of a tertiary education curriculum. An educational ontology has been developed that conceptualizes educational entities within curriculum and syllabus with sufficiency and richness. The ontology will be able to support rich services on top for improving curriculum management and development as well as enabling syllabus semantic searching, matching, interlinking and recommendation.

The ontology comprises of 41 concepts in a taxonomy, 9 of which are the top-level concepts. It also includes 54 objects properties for establishing relations between concepts and 76 data properties for describing concepts characteristics. All entities are enriched with extra annotation information in two languages. In addition, a large number of instances created.

In this chapter, the methodology and development process of the educational ontology are described in detail.

3.1 METHODOLOGY

The design and development of ontology usually encompasses several tasks. There is not one standard methodology for developing ontologies. In the literature, can be found methodologies that order these tasks differently but the general approach is not differs a lot.

An adequate methodology for ontology creation that adopted by many researchers includes the following seven steps [53], [79], [83]:

1. Determine the domain and scope of the ontology
2. Ascertain if there is related ontologies
3. Enumerate important terms in the domain
4. Define the key concepts as classes and class hierarchy
5. Describe the properties of classes
6. Attaching facets to the properties
7. Create instances

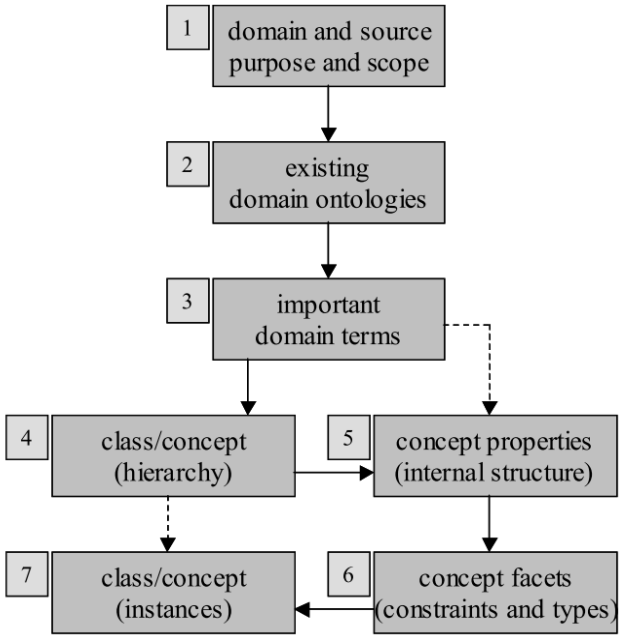


Figure 16: The development process of an ontology (source: *Boyce & Pahl* [79])

Another comprehensive methodology that deals with the whole development process, includes four steps with the second step to be distinguished in three sub-steps [84], [85], [86]:

1. Scope and purpose of the ontology
2. Ontology building
    a. Ontology capture
    b. Ontology encoding
    c. Ontology integration
3. Ontology evaluation
4. Ontology documentation.

The methodology that followed in this thesis is a combination of the above-mentioned two methodologies and is composed of four main stages, with the second stage to be distinguished in six sub-stages, as illustrated in *Figure 17*:

1. Domain and Purpose definition
2. Ontology building
    2.1. Survey existing related ontologies
    2.2. Enumerate important terms
    2.3. Define classes and class hierarchy
    2.4. Describe the properties of classes

2.5. Attaching facets to properties

2.6. Create instances

3. Evaluation
4. Documentation

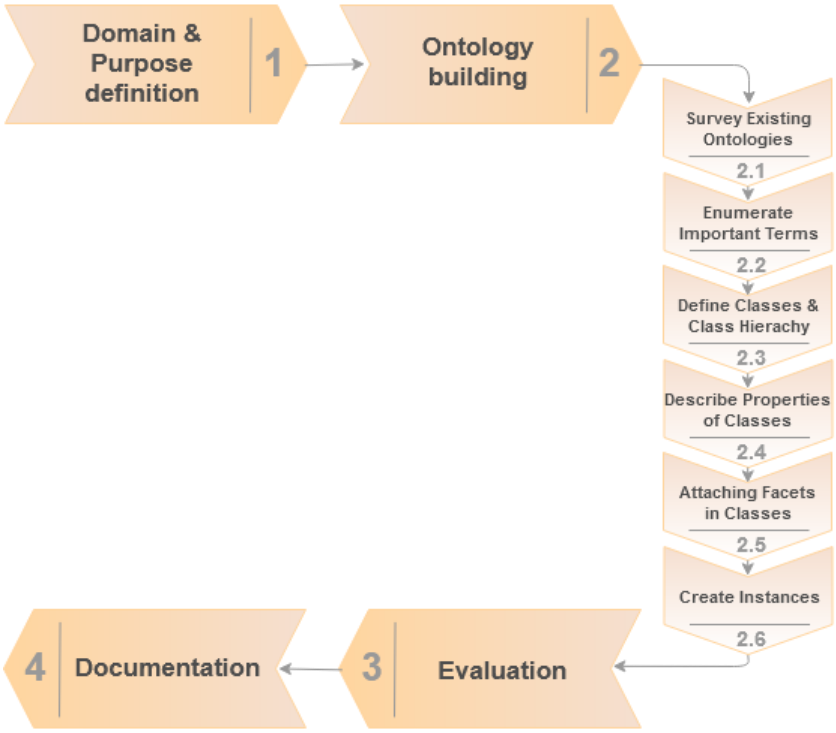


Figure 17: The Methodology of the Development of CCSO

It must be noted that the last two stages, Evaluation and Documentation, should be performed jointly with ontology development process in a iterative process [86]. Each stage of this process is presented and described in details in the next paragraphs.

3.2 Tools

One of the most fundamental concerns when building ontology is to choose a suitable ontology editor. In this thesis, the tool that used for building our ontology is *Protégé* Desktop [21].

Its integrated and intuitive interface makes it easier for non professionals in ontology engineering to build their own ontology while it offers sufficient options for professionals. In addition, the large number of available plug-ins extends its capabilities.

Protégé supports the most known modern ontology languages with OWL 2 - the ontology language that used in our project - among them. Furthermore, Protégé 5 comes with the *HermiT* OWL reasoner included that is considered as one of the best open-source OWL reasoners available.

Protégé is currently the most popular tool for ontology development and editing [26]. Thus, there is a growing community of users for collaboration and support.

It must be noted that, in newer versions of Protégé, the terms '*properties*' and '*individuals*' are used mostly instead of '*slot*' and '*instances*' for the corresponding concepts.

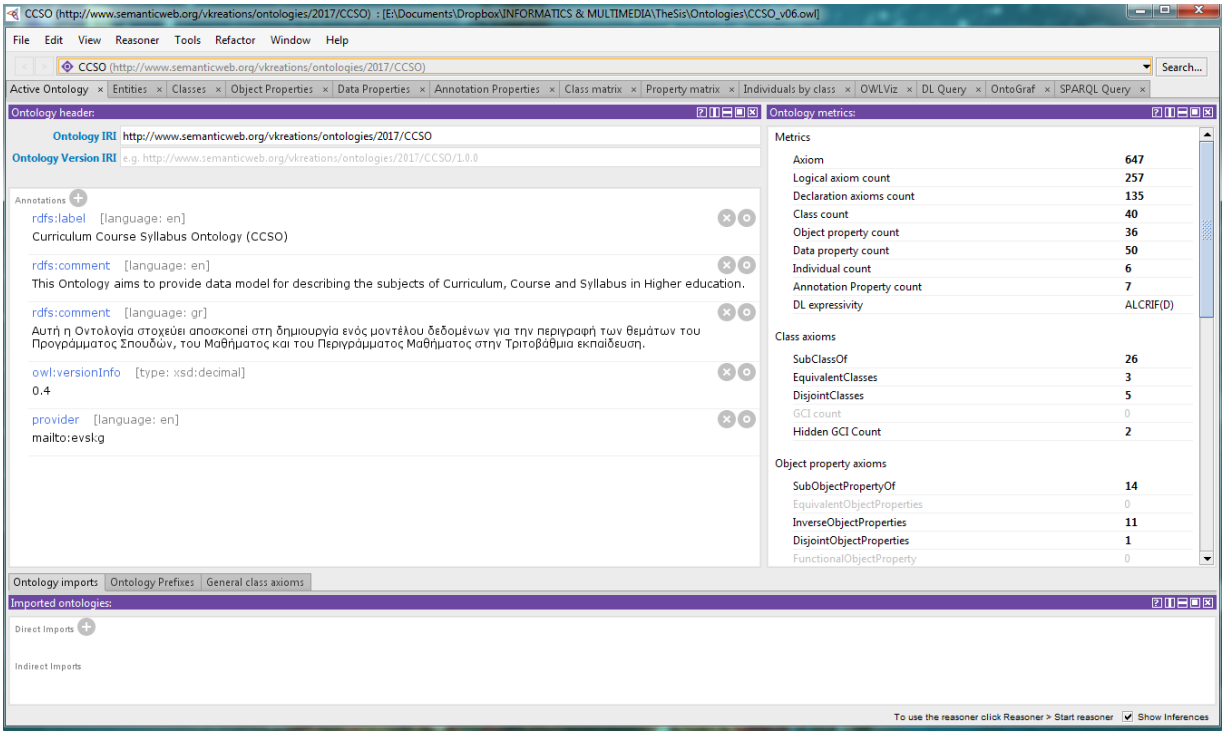


Figure 18: The environment of Protégé 5

OpenLink Virtuoso[23] is used as TripleStore for upload and storage of triples of the ontology (*Figure 19*). The storage of triples in a TripleStore makes the RDF data available to the Internet and facilitates the access and retrieval of RDF data through semantic queries in SPARQL.

[23] OpenLink Virtuoso: https://virtuoso.openlinksw.com

Figure 19: The environment of Virtuoso Conductor

3.3 DOMAIN AND PURPOSE

First of all, we need to determine the domain as well as the purpose and scope of the ontology. That is, answer some basic questions [79], [83]:

- What domain will the ontology cover?
- What is the purpose of the ontology?
- What type of questions the ontology should be able to answer?
- Who will use and maintain the ontology?

This work aims to provide a general ontology for the knowledge domain of the subjects of Curriculum, Course and Syllabus in Tertiary education, so that educational semantic web has a knowledge base.

The purpose is to design a generalized expression of the concepts, entities and theories that exist in the construction of a third-level curriculum. The ontology designed to be valid and compatible for any kind of curriculum and syllabus, and being easily adapted and reused from other schools or universities. The construction of any Curriculum or Course ontology for a specific domain could be based on the general ontology and enriched with particular appropriate classes, relations and instances of the domain.

An additional goal of this thesis is to support ontology based triple-store repository for hosting Curriculums and Course Syllabuses of different educational institutions (universities, high schools, secondary schools, primary schools, vocational training schools, *etc*).

The ontology can be used for inspecting, managing and monitoring a full study course and the repository would be useful for program reviews and curriculum improvement. Students, teachers as well as software agents could use the stored information for searching and comparing or constructing curriculums and course syllabuses.

Competency Questions

A good approach to determine the scope of the ontology is to define a list of questions that introduce some scenarios for the proposed ontology in terms of its applications. These questions are known as *Competency Questions* [87] and play two roles for our ontology. On the one hand, they outline the expectations the designed ontology should fulfill, and on the other hand they will be used later to evaluate the ontology achievements by examine the answers given to these questions.

In the knowledge domain of Curriculum and Syllabus in higher education, considering the possible scenarios and requirements students and instructors may have from the ontology, we defined the following competency questions, classified in four sub-sets according the related class or the purpose.

Questions related to Organization and Department

1. Which study programs does a Department provide?
2. Who are the alumni of a Department?
3. Who are the employees of a Department?
4. Who are the members of the Academic Staff of a Department?

Questions related to Person

5. Which degrees does a Person have?
6. Which Courses does a Professor teach?
7. Which Courses does an Assistant assist in?
8. Which Publications has a Person authored?
9. Which Courses has a Student successfully completed?

Questions related to Course and Syllabus

10. Who are the Instructors of a Course?

11. Who are the Students that attend a Course??
12. Which are the Syllabi of a Course?
13. Which are the Topics and Learning Outcomes of a Course?

Questions related to study planning

14. Which Courses are included in a study program?
15. Which are the core Courses in a study program?
16. Which Courses have prerequisites (and what are they) in a Study Program?
17. Which Courses cover a Knowledge Area?
18. How many credits has a Student got so far?
19. Can a Student enroll in a Course?
20. Can a Person register in a study program for a degree?

Considering this list of questions, the ontology should contain information about several properties of its core concepts. Speaking about the *Department*, it should include information about listed study programs, alumni and employees while the *ProgramofStudy* should inform about courses included, students and credits. As to the *Course* and *Syllabus*, it should include information about instructors and attendees of a course, requirements, credits, topic, as well as sufficient data for describing a course such as type of course, semester that it is offered, *etc*.

3.4 BUILD THE ONTOLOGY

Building domain ontology requires deep understanding of the domain of interest. The necessary additional knowledge can come from domain experts, textbooks and existing ontologies.

Domain experts offered meaningful help with their knowledge and experience in all phases of the development. Textbooks and documents were another important source of information, especially during the phase of ontology capture. International curricular guidelines, syllabus templates and school study programs have been taken into consideration. Existing ontologies can also offer powerful help in building the ontology, either with full integration or by importing some parts of them or as worthwhile case studies.

3.4.1. SURVEY EXISTING ONTOLOGIES

Survey and consider reusing existing ontologies should be one of the first actions in ontology design. Probably, there must be a number of ontologies already available that can be refined and extended for our particular needs. Even if it is not compatible enough to merge, it may be a good source of inspiration and guidance.

The survey for existing educational ontologies focusing in Curriculum and Course Syllabus included research in search engines and ontology repositories. We search for available ontologies using the following search engines:

- **Google** google.com

  Common search engines are designed to work unstructured text in natural languages. However, *Google* can be used for finding ontology documents by using the facility "filetype:" and limiting the results to ontology files (like RDF, OWL, *etc*).

- **Swoogle** http://swoogle.umbc.edu

  Swoogle is a search engine for the Semantic Web, introduced by *Li Ding* and his fellows [88]. It is a crawler-based indexing system for Semantic Web documents that also extracts metadata and inter-document relations from discovered documents. It allows users to query for ontologies that contain specific terms as Classes, Properties, Instances or Comments and to discover how documents are inter-connected (usage and imports).

- **OntoSearch** www.ontosearch.com

  OntoSearch is a tool that combines the Google search engine with ontology visualization capabilities [89]. OntoSearch based on Google APIs to search and retrieve RDFs files. The user can chose any of the returned documents to view their taxonomy visualized.

- **SemSearch** http://technologies.kmi.open.ac.uk/semsearch

  SemSearch is a keyword-base semantic search service that allows users to specify queries with subject and keywords. In the website someone can find useful information about the project. Unfortunately, the search service that is in page '*Applications*' seems to be unavailable.

- **Google Scholar** scholar.google.gr
  Google's search engine for academic publications. It searches and indexes academic works published in academic journal articles, books or thesis.

The search included also the following ontology repositories:

- DAML Ontology Library www.daml.org/ontologies
- Ontolingua ontolingua.stanford.edu
- SHOE www.cs.umd.edu/projects/plus/SHOE

We focused our research in ontologies which conceptualizes knowledge and learning concepts, i.e. *Curriculum*, *Course* and *Syllabus*. First and foremost are two seminal ontologies, *BBC Curriculum ontology* [77] that models UK Curriculum and *Bowlogna ontology* [75] which describes the concepts in an academic environment according to Bologna Process. These two, plus two ontologies designed by Chung & Kim [38], [63] were the most influential in the development of our ontology.

Ontologies designed by *Chung & Kim* in [38], [63] have similar core concepts. *Zeng et al.* [82] and *Oprea* [34] presented ontologies for university courses as well as *Boyce & Pahl* [79] who also described a method of constructing course ontologies. Another interesting ontology, is the SWRC ontology, developed by *Sure at al.* [90], that models research communities and related entities within. It must be noted that it wasn't possible to find the ontology files but only the works which presented the ontologies. The only exception, is the SWRC ontology whose files are available online [91].

Demartini et al., The Bowlogna Ontology [75]

Demartini and his fellows [75] developed the *Bowlogna Ontology* which describes the entities and relations involved in an academic institution according to Bologna Process [92]. The ontology aims to facilitate automated processes and workflows across European universities. Improving student mobility is one of the main goals of *Bologna process*, thus, *Bowlogna ontology* pays special attention in course's grading system and student's study tracking.

Figure 20 shows some classes with their relationships in the *Bowlogna Ontology*. The ontology contains 25 top-level classes (66 in total) and models an academic setting. Some of the key classes in this ontology are:

- Academic_Degree
- Professor
- Student
- Teaching_Unit
- Evaluation
- ECTS_Credits
- *etc.*

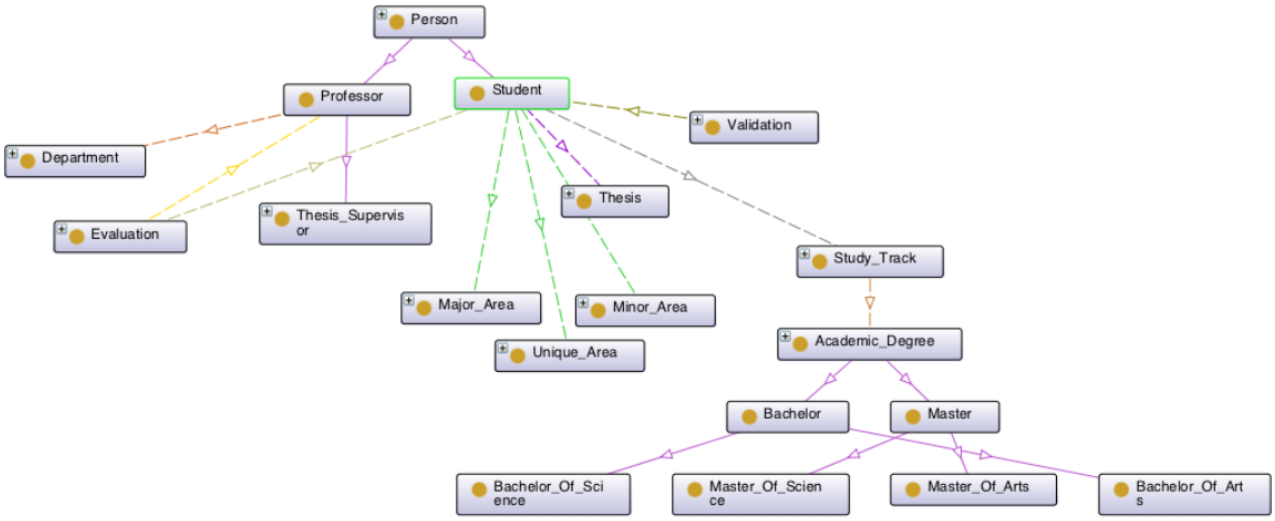


Figure 20: Some classes and their relationships in the Bowlogna Ontology (source: *Demartini et al.* [75])

Despite *Bowlogna* ontology is quite relevant to our ontology, it is used as a notable case study and reference instead of merge it into our ontology since it has different purpose and sets its focus in facilitating study tracking, student evaluation and degree transferability.

BBC Curriculum Ontology [77]

BBC's Ontology models the UK curriculum using three key concepts in a 3-dimensional space, i.e. *Levels*, *Field of Study* and *Topics*. The 3-D diagram is shown in *Figure 14* at the previous reference for the ontology.

Level denotes the stage of study (e.g. 1st grade of Elementary School) with four sub-classes corresponding to the four different ways to specify the level: *Age*, *KeyStage*, *Year* and *Phase*.

Field of Study refers to the disciplines of a curriculum such as Maths, History, English, etc. *Topic* defines educational subjects in a more specific way than the *Field of Study* (e.g. Geometry in Math, energy in Physics).

ProgrammeofStudy is the combination of these 3-dimensions *Level*, *Topic*, *FieldofStudy* and defines the atomic units of a curriculum. The diagram in *Figure 21* shows the classes and properties have been defined in the ontology.

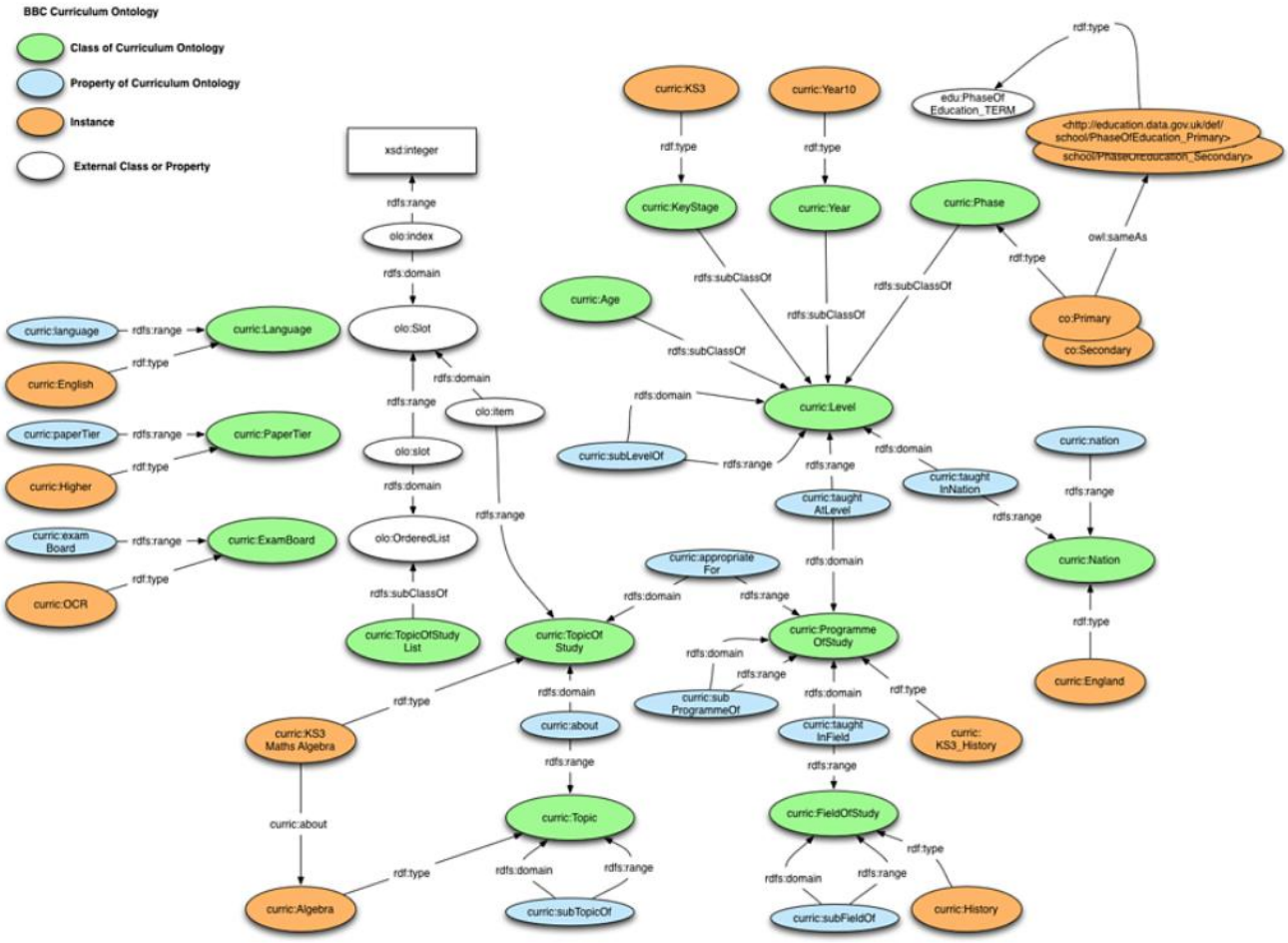


Figure 21: Classes and Properties in *BBC Curriculum Ontology* (source: *BBC* [77])

Although, BBC's data model may be applicable for describing curricula in other countries, we didn't merge it because it describes curriculum in a broader view incorporating information that would be superfluous in our curriculum ontology. BBC's ontology formally describes the national curricula; therefore it goes in deep analysis about topics and content mainly in compulsory education while our focus is after that stage of study.

However, it was quite helpful for studying the concepts of *ProgrammmeofStudy*, *FieldofStudy*, *Topic* and their relations, as they are also included in our ontology as presented later in *Table 4*.

Chung & Kim, Ontology Design for Creating Adaptive Learning Path in e-Learning Environment [63]

In their first work, *Chung & Kim* [63] designed an ontology-based e-learning system which could enhance active learning allowing the creation of adaptive learning paths by learners themselves. The proposed ontology model integrates multiple ontologies on different layers, i.e. *Curriculum ontology*, *Syllabus ontology*, and *Subject ontology*. Each ontology contains classes, properties and relations for describing the knowledge and concepts of the particular domain. *Figure 22* shows the ontologies and the relationships between them with some classes and properties.

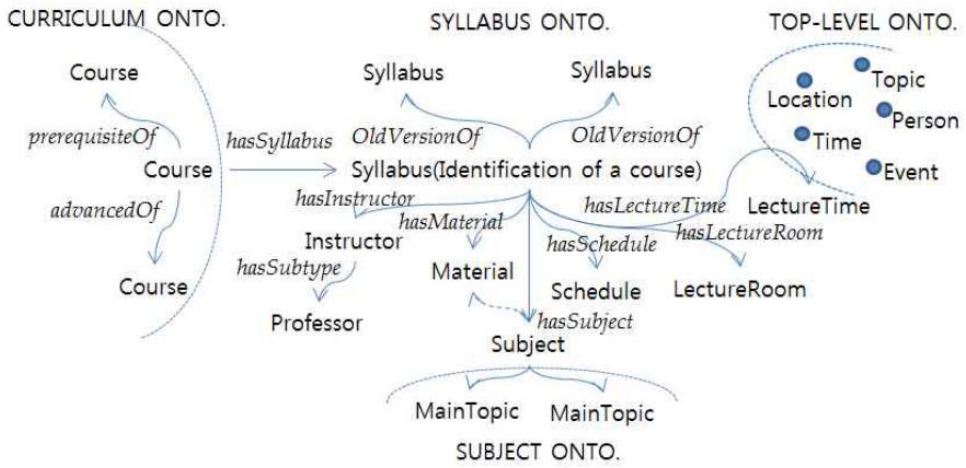


Figure 22: The ontologies and the relationships between them (source: *Chung and Kim* [63])

Curriculum ontology establishes connections with one or more syllabus ontologies, each of them contain one or more subject ontologies. Some classes that included in the three proposed ontologies are listed in *Table 1*.

Ontology model designed in this work have some similar parts with our project but it has different purpose. It is meant to support e-learning system where users will be able to build their own learning paths. Subject ontology plays significant role in building learning paths so it is described in depth with information that, currently, is unnecessary in our ontology. Moreover, some key concepts essential for our ontology are missing, such as

EducationalOrganization and *Discipline* which however are included to the next ontology by the same researchers.

Curriculum ontology	Syllabus ontology	Subject ontology
ProgramOfStudy	Syllabus	LearningConcept
Course	Instructor	FundamentalConcept
KeyConcept	Professor	AdvancedConcept
AttainmentGoal	Material	RelatedConcept
AttainmentLevel	Schedule	Example
etc.	LectureRoom	*etc.*
	etc.	

Table 1: Classes included in the three integrated ontologies (source: *Chung & Kim* [63])

Chung & Kim, An Ontological Approach for Semantic Modeling of Curriculum and Syllabus in Higher Education [38]

In this work, researchers kept the multilevel approach in the design of the ontology which has been extended further. The presented learning ontology conceptualizes different kinds of learning concepts based on a layered structure consist of four ontologies which are *Curriculum ontology*, *Syllabus ontology*, *Subject ontology* and *Resource ontology*. In this paper, Curriculum ontology and Syllabus ontology are described in detail. *Figure 23* shows the four layers of the ontology while its classes are listed in *Table 2*. Upper ontology can connect with one or more lower ontologies through relationships.

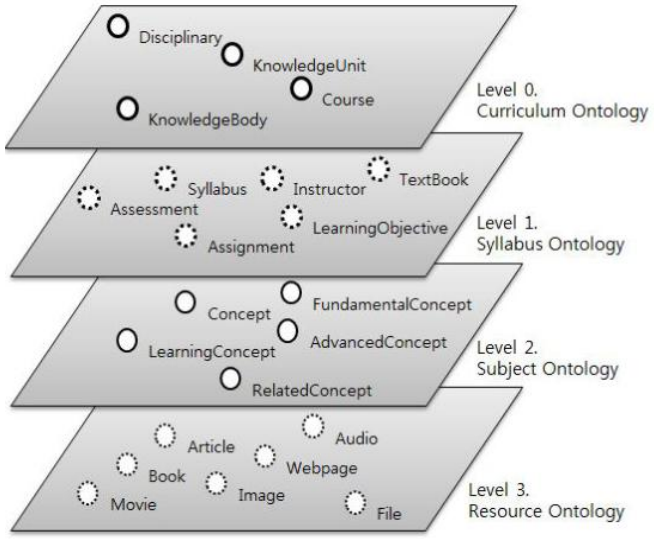


Figure 23: The four-layered integrated Learning Ontology (source: *Chung & Kim* [38])

Curriculum ontology	Syllabus ontology	Subject ontology	Resource ontology
Discipline	Syllabus	Concept	Article
BodyOfKnowledge	Instructor	LearningConcept	Book
KnowledgeArea	LearningOjective	FundamentalConcept	Movie
UnitOfKnowledge	LearningMaterial	AdvancedConcept	File
Competency	TeachingMethod	RelatedConcept	Webpage
University	LearningMethod	*etc.*	Image
College	Assignment		Audio
Department	ExamQuiz		*etc.*
Course	*etc.*		
MajorCourse, ElectiveCourse, DesignCourse			
etc.			

Table 2: Classes included in the three integrated ontologies (source: *Chung & Kim* [38])

The designed *Curriculum Ontology* provides classes and properties aiming to represent the knowledge structure of disciplines as well as the curriculum structure (*Figure 24*). It can be used by people in education for supporting curriculum building, comparison and progress.

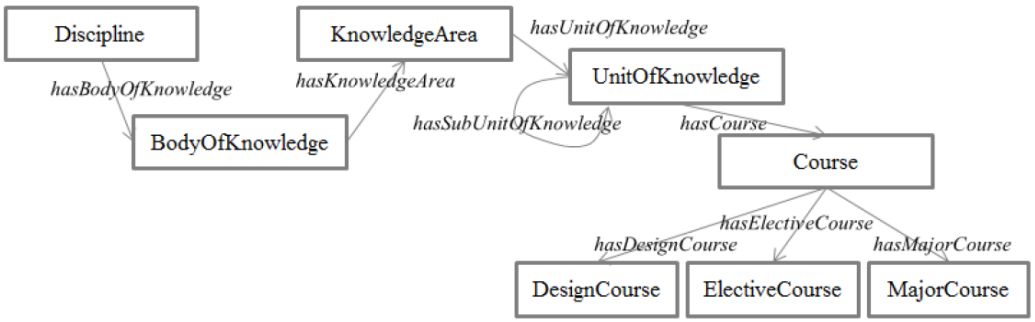


Figure 24: Classes and relationships in Curriculum ontology (source: *Chung & Kim* [38])

The goal of *Syllabus ontology* is to support people and machines in intelligent services about syllabus such as syllabus matching, syllabus versioning, adaptive learning paths construction with syllabus recommendations, *etc* (*Figure 25*).

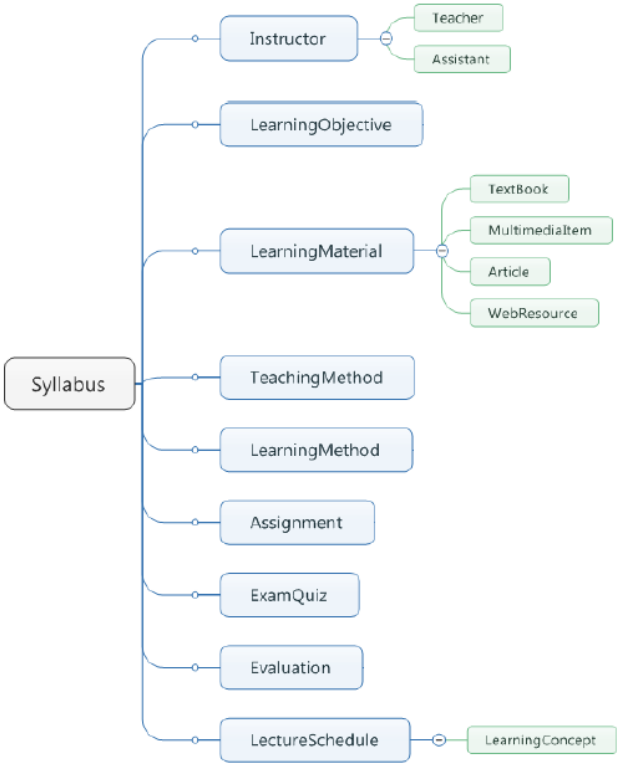


Figure 25: Class hierarchy of the Syllabus ontology (source: *Chung & Kim* [38])

Moreover, to measure the achievement of learning objectives from the students, researchers designed a learning objective class using a structured format of learning objectives based on Boom's taxonomy (*Figure 26*). In order to automatically calculate the achievement level, they created an achievement matrix. In addition, they specified methods for syllabus classification of heterogeneous syllabuses and syllabus integration.

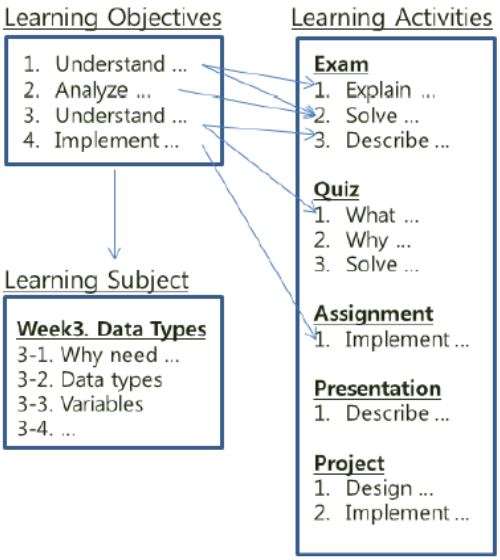


Figure 26: Classes for measure the learning achievement (source: *Chung & Kim* [38])

The previous two ontologies presented by *Chung & Kim* are quite close to our educational ontology, sharing some similar points and goals but there are differences too. Some of the goals and usages of their ontology are beyond our intentions without this to dispute their significance. However, the most important difficulty was that neither of them is available online.

The list below depicts the important terms of the integrated ontology which are also listed in the combined *Table 3* in next paragraph:

- Curriculum
- University
- College
- Department
- Discipline
- BodyOfKnowledge
- KnowledgeArea
- UnitOfKnowledge
- Competency
- ProgramofStudy
- Course
- Syllabus
- Instructor
- LearningOjective
- LearningMaterial
- TeachingMethod
- LearningMethod
- Concept
- LearningConcept
- FundamentalConcept
- AdvancedConcept
- RelatedConcept

Zeng, Zhu & Ding, Study on Construction of University Course Ontology: Content, Method and Process [82]

Ontology proposed by *Zeng, Zhu & Ding* [82] presented a top-level ontology that models the general concepts and relationships of a course. The main concepts described in their ontology are: *Curriculum*, *KnowledgeArea*, *BasicConcept*, *Topic* and *LearningGoal* (*Figure 27*). Researchers state that the top-level ontology should be the base for the development of any other specific course ontology.

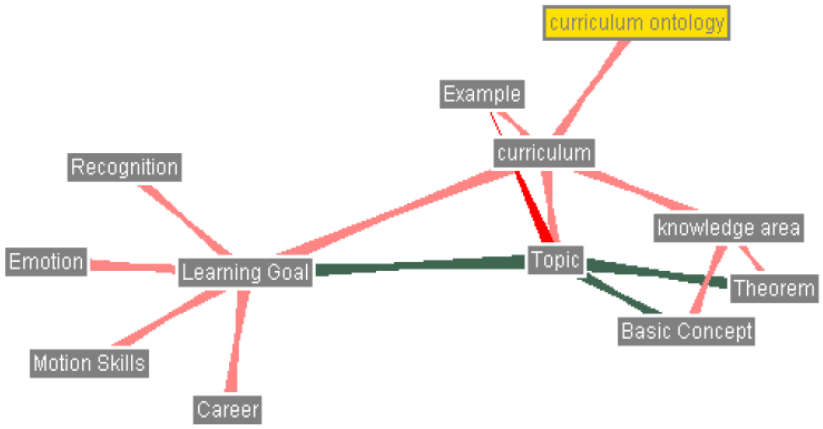


Figure 27: The top-level Course ontology (source: *Zeng et al.* [82])

In this work, the ontology is described shortly without many details and the classes that used are rather common in educational ontologies. The described process of the ontology development was helpful as a general guide, although it was also short.

Oprea, An Educational Ontology for Teaching University Courses [34]

Oprea developed an educational ontology for supporting a teaching system of university courses [34]. She developed a course ontology, named *Univ_Edu_Onto*, with a vocabulary consisted of the general terms of a university course such as *discipline*, *curriculum*, *objective*, *professor*, *audience*, *etc*. The ontology aims to become a top-level ontology for building specific courses ontologies with the addition of course's specific terms. *Figure 28* shows the general terms on the ontology and the specific terms for a course about Artificial Intelligence. The main relationships used in the ontology are *has*, *part_of*, *required_by* and *is_a*.

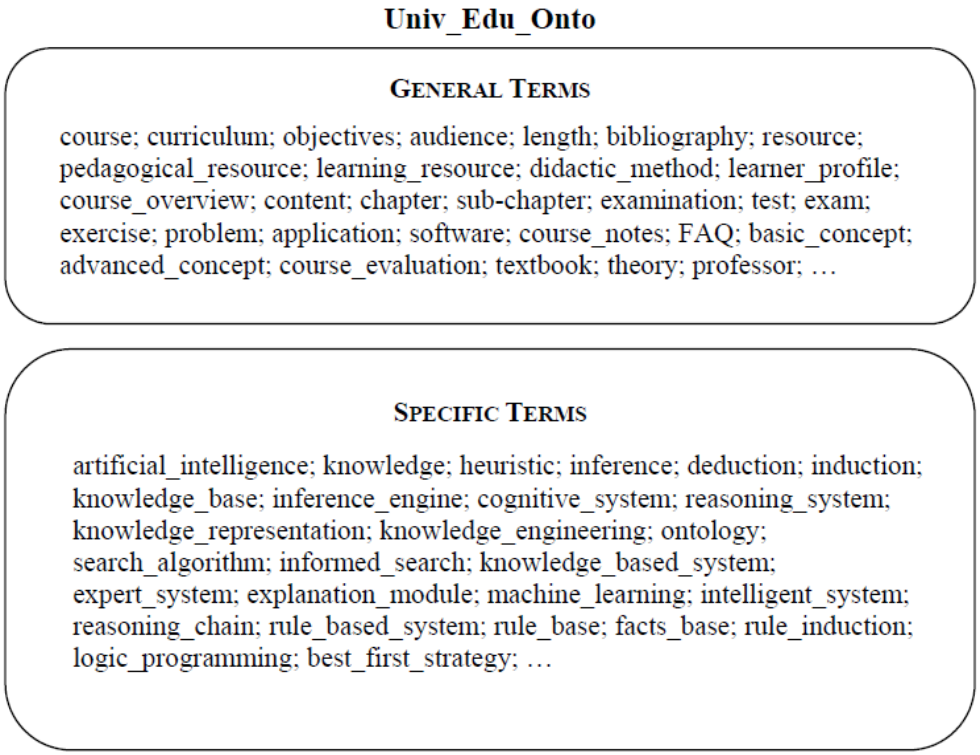


Figure 28: Terms of a university course ontology (source: Oprea [34]**)**

Below is the list of some of the general terms that have a meaning about our project, which are also listed in the combined *Table 4* in next paragraph.

- Course
- Curriculum
- Objective
- Resource
- Didactic_Method
- Course_Overview
- Content
- Examination
- Test
- Basic_Concept
- Advance_Concept
- Course_Evaluation
- Professor

Since Univ_Edu_Onto assists an e-learning system, Oprea describes entities within a course in depth including terms and information unneeded in our case.

Boyce & Pahl, Developing Domain Ontologies for Course Content [79]

Boyce & Pahl presented a systematic approach to Ontological Modeling of Domain Ontologies. Particularly, they focused in defining education-specific relationship types that can be used by domain experts to conceptualize rich domains adequately [79]. In order to

separate the knowledge structure from the associated content, they split the overall knowledge into two spaces, the *concept space* and the *educational content space*.

The relations defined in the *concept space* are *IsBasisOf*, *HasSubtype*, *HasPart*, *HasConstraint* and *HasFunction*. '*HasSubtype*' is the substitute of '*Is-A*' relation and that makes it the most common relationship type. Researchers noted that the use of '*Is-A*' relation often resulted in the creation of inappropriate relationships by being used to relate concepts with their definition or show synonyms rather than describe a generalization/specialization relation.

In the *educational content space*, the defined relationship types are *HasDefinition*, *HasSynonym*, *HasAsExample* and *HasFurtherExplanation*. The two sets of relationships have been defined during the design of ontology for course content in the database domain. The transferability of the relationships has been tested with the implementation of a second ontology in a different educational domain, i.e. enzymology, where only one relationship didn't used.

The two ontologies developed by *Boyce & Pahl*, have different target than our ontology describing the teaching content of the courses. In their work, researchers analyzed the ontology development process they followed (*Figure 16*), which presented by *Noy & McGuinness* [83] and influenced our work too, as shown before in. Finally, we used some of the relationships specified in the concept space, i.e. *hasSubtype* and *hasPart*.

Below we review ontologies related to the concepts of *Curriculum*, *Course* and *Syllabus* which have been found in our research in search engines and ontology repositories. Their ontology files are available on the web and can be reused

University Ontology - Department of Computer Science, University of Maryland

Source: www.cs.umd.edu/projects/plus/SHOE/onts/univ1.0.html

This ontology is written in SHOE language [93] and describes the organizational structure of a university and the activities occur in it. It includes concepts such as *faculty*, *student*, *course*, *research*, and *publication*. Its focus is mostly at the affiliation between a person (as

professor, researcher or student) and the organization. There is a class *Course* (sub-class of the class *Work*) which describes a course from the perspective of *work* for a member of *Faculty* class and the perspective of *event* (class *Schedule*) for an organization. There is some information useful for our project but important concepts, such as *Syllabus* and *Subject* do not exist at all. The classes and properties of the ontology are depicted in *Figure 29* as *Univ-CS ontology,* which illustrated there, is a small extension of the *University Ontology*.

Univ-CS - Department of Computer Science, University of Toronto

Source: www.cs.toronto.edu/semanticweb/maponto/MapontoExamples/univ-cs.owl

This ontology extends the, previous mentioned, *University Ontology* (SHOE) with some sub-classes, meaningful for a Computer Science Department. Like its base ontology, it lacks information about essential concepts for our project.

The SHOE version of the ontology can be found in the repository of SHOE language ontologies of the Department of Computer Science in the University of Maryland (www.cs.umd.edu/projects/plus/SHOE/onts/cs1.1.html). *Figure 29* presents the classes and the properties of the ontology in Protégé.

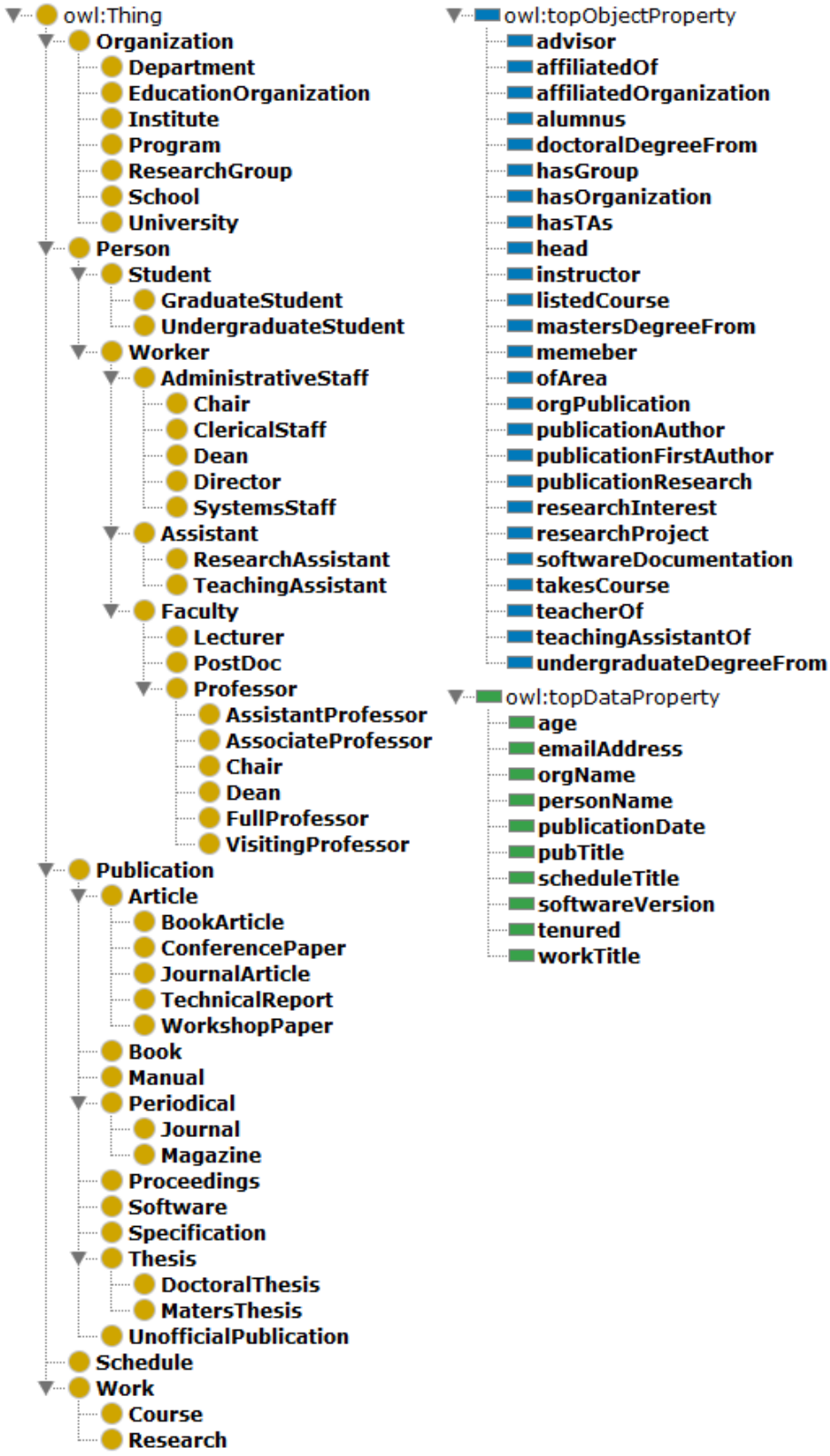


Figure 29: The Classes and Properties of Univ-CS ontology in Protégé

AIISO (Academic Institution Internal Structure Ontology) - Electrical Engineering and Computing, University of Zagreb

Source: vocab.org/aiiso/schema

AIISO ontology describes the internal organizational structure of an academic institution. It is a knowledge organization ontology which includes classes and properties for describing entities within an institution and can mainly be used internally the institution by people who work there (*Figure 30*). It was useful in our project for studying the internal organizational structure of an educational organization.

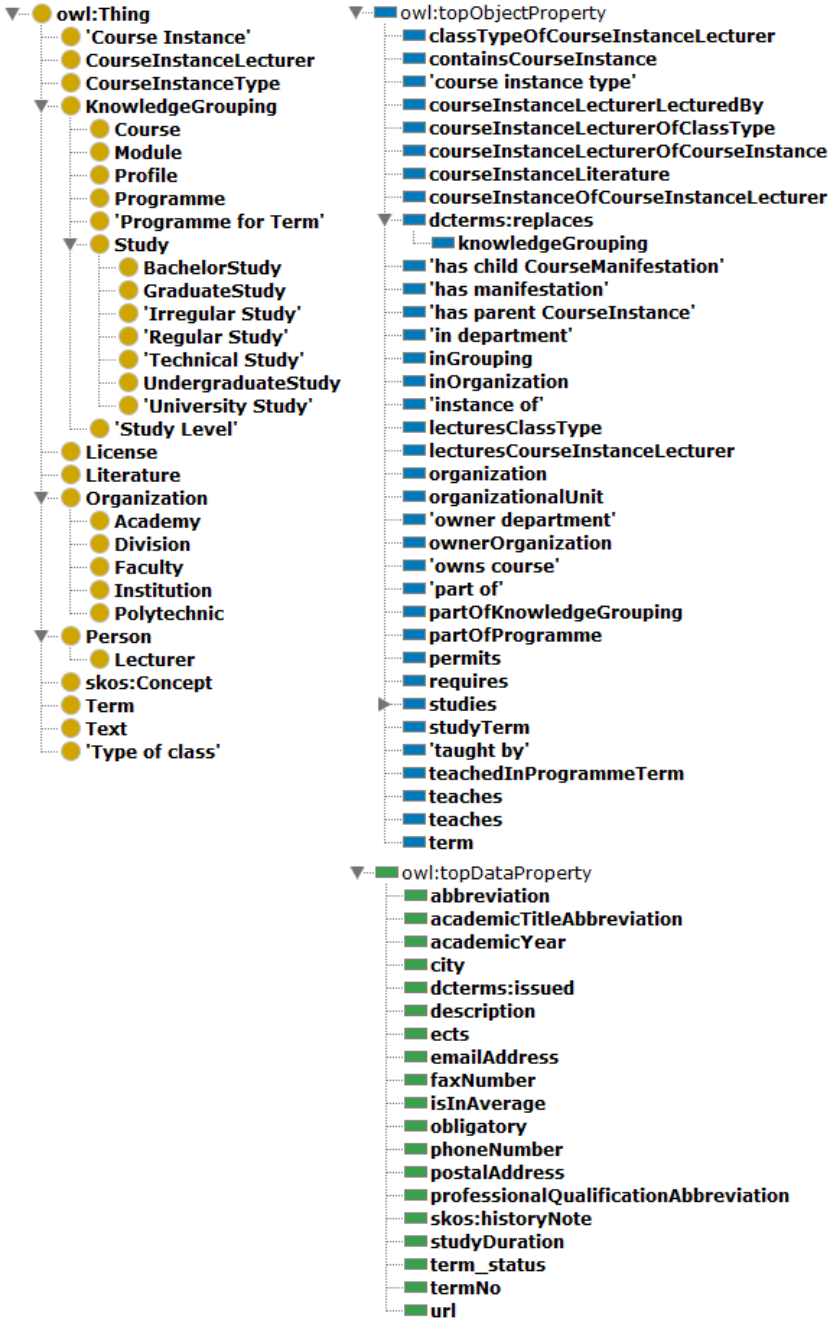


Figure 30: The Classes and Properties of AIISO ontology in Protégé

Univ-Bench - Department of Computer Science and Engineering, Lehigh University

Source: swat.cse.lehigh.edu/onto/univ-bench.owl

Univ-Bench is a university ontology developed for benchmark tests. Its main classes are: *organization*, *person*, *publication* and *work*. It gives a detailed view of the organizational structure of an institute and the kinds of affiliation between an organization and a person as employee or either as student (*Figure 31*).

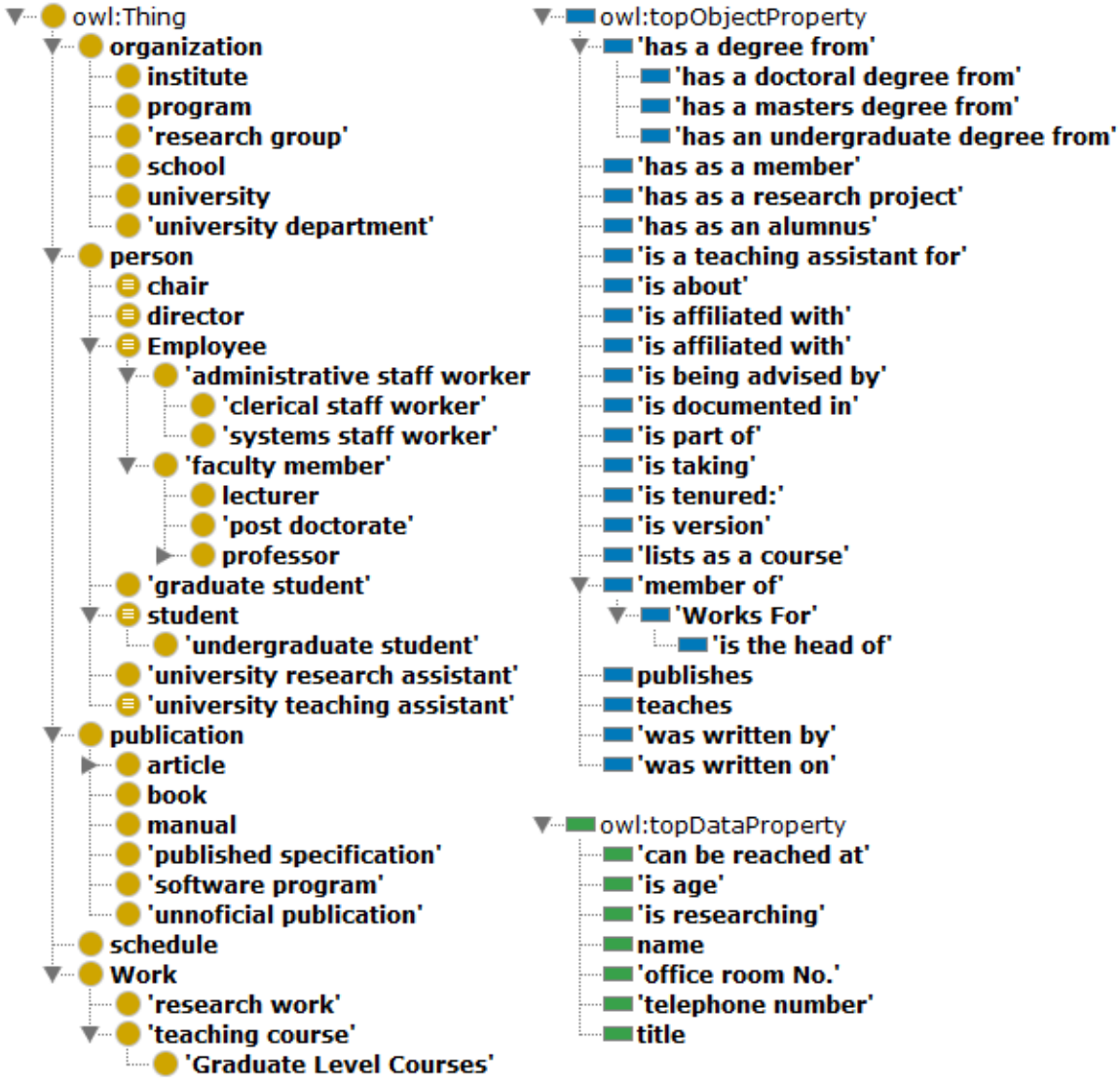


Figure 31: The Classes and Properties of Univ-Bench ontology

SWRC Ontology (Semantic Web for Research Communities)

Source: ontoware.org/projects/swrc

SWRC ontology models key entities within research communities [90]. The ontology comprises of six top level concepts, named *Publication*, *Person*, *Organization*, *Event*, *Topic* and *Project* (*Figure 32*). It contains classes and properties for describing typical research communities and the relationships between them and is used by some portals and applications for the semantic annotation of their resources and publications [90].

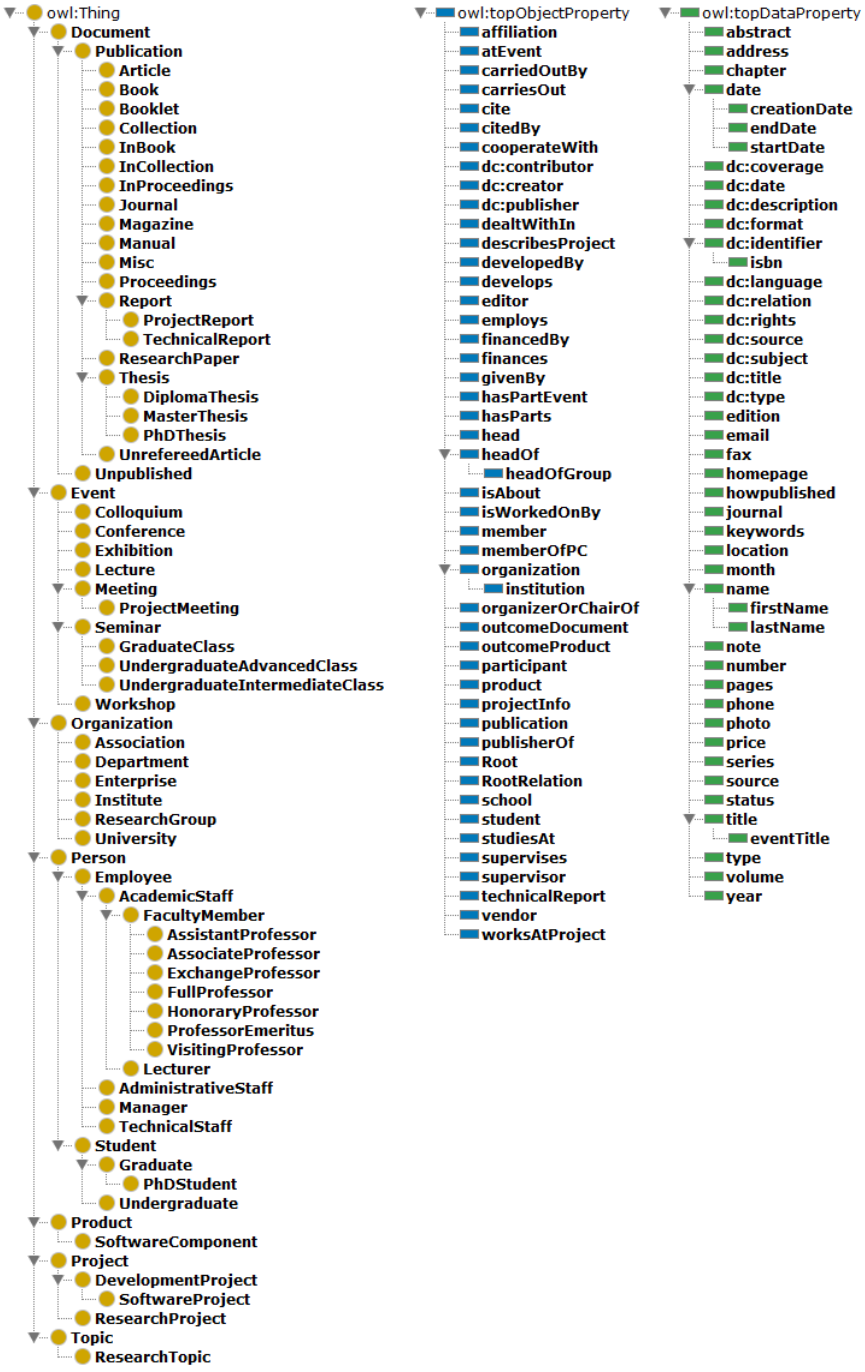


Figure 32: The Classes and Properties of SWRC ontology in Protégé

instOntology - Indian Statistical Institute

Source: www.isibang.ac.in/~bisu/ontology/instOntology.owl

InstOntology is an institutional ontology that provides classes and properties for describing the teaching entities in the institute. It includes five main classes named *Centre*, *Person*, *Course_Name*, *Event* and *Publication*. The classes and properties used with the existed instances, shows that the ontology aims to describe events and activities relevant to the teachers and students that occur inside the particular institute and used by employees or applications inside there (*Figure 33*). It can be adjusted and reused by other institutes because of its typical structure.

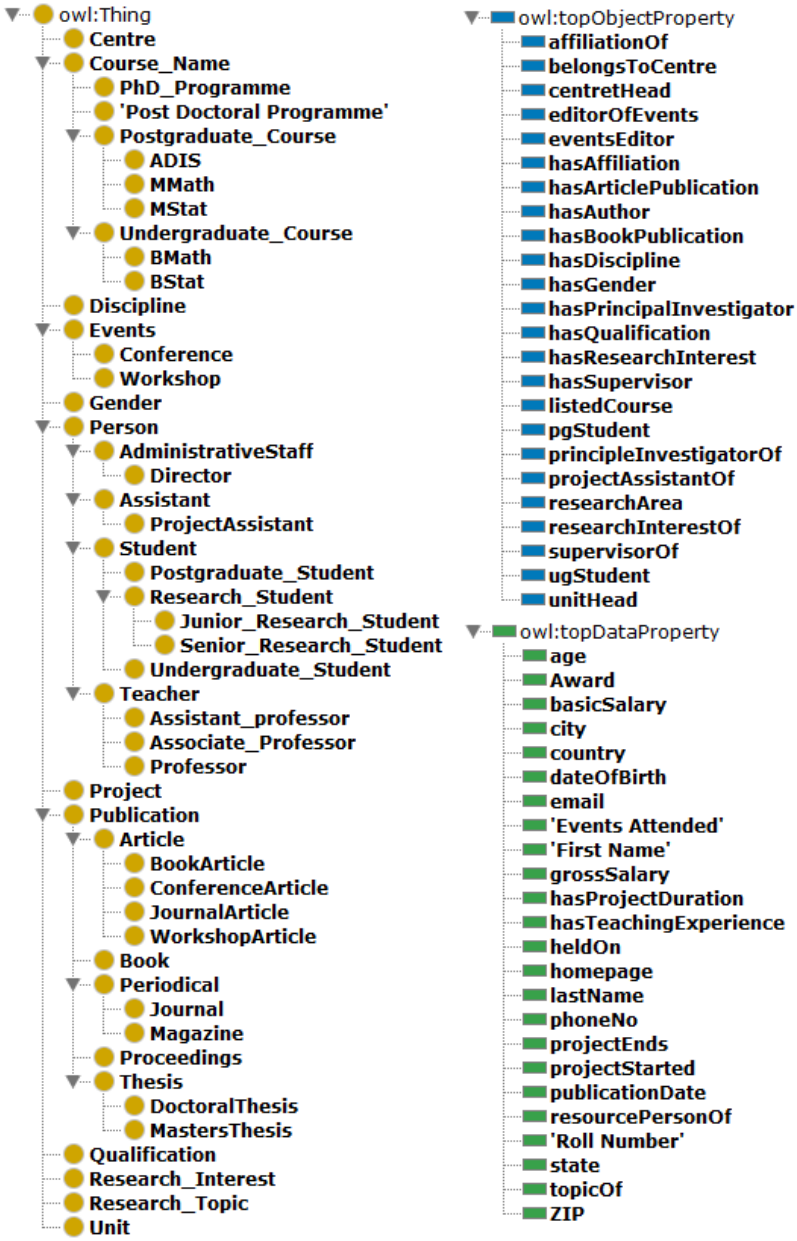


Figure 33: The Classes and Properties of instOntology in Protégé

Karma-Syllabus Ontology

Source: github.com/szeke/karma-tutorial/blob/master/ontology/syllabus.owl

This ontology was the most relevant result when searching for ontology files with the word '*syllabus*' in the filename. It built during the development of a software testing tool named *Karma Tutorial*[24]. It contains few classes and properties and describes teaching and evaluating events happen in a course (*Figure 34*).

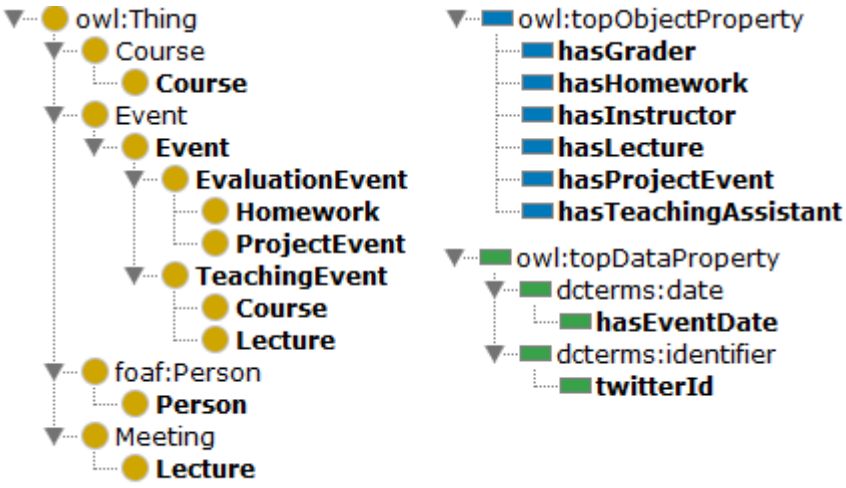


Figure 34: Classes and Properties of Karma-Syllabus Ontology in Protégé

Summary of the evaluation of existing ontologies

Briefly, we can say that most of the previous mentioned approaches presented, have similarities and complete each other for describing the subjects of educational curriculum, course and syllabus. However, there are points that are very general or either very detailed for our ontology and are not adopted, points that are modified or extended as well as points that needs to be added. Ontologies that found with online research, lack in documentation which makes it difficult to understand and making the most of them. As to the ontologies presented in literature [34], [38], [75], [79], [82] there are not any ontology files publicly available. Probably some were either only design proposals while others haven't been published. The only exceptions were *BBC Curriculum Ontology* [77] and *SWRC Ontology*

[24] Official Karma website: http://karma-runner.github.io/0.12/index.html

[90] with ontology files and documentation available. We used and adjust some parts of available existing ontologies in composing and building our ontology as well as reputable references and noteworthy case studies.

3.4.2. ENUMERATE IMPORTANT TERMS

Before describing concepts, classes and relations, it is a good practice to write down a list of all possible terms we would like either to describe or to make statements about. At this step, it is important to make a complete list of terms without worrying about concept overlapping, relations or properties they may have. This list will probably change during the next two steps with the definition of the classes and hierarchy and the description of the properties.

Initially, we inspected carefully the existing related ontologies that mentioned in the previous paragraph and extracted the important terms that could be included in our list. *Table 3* lists the important terms in existing ontologies. Some of these terms may overlap or correspond to the same concept (such as *ProgrammeofStudy* and *AcademicDegree*) but the goal is to guide us in making our own list.

BBC Curriculum ontology [77]
Level
Field of Study
Topic
Topic of Study
Programme of Study
Bowlogna Ontology [75]
Curriculum
Field of Studies
Academic Degree
Department
Educational Module
Subject
Study Track
Learning Activity
Learning Outcome
Teaching Unit
Professor
Student
Thesis

Learning Ontologies by Chung & Kim [38],[63]	
Discipline	BodyOfKnowledge
KnowledgeArea	UnitOfKnowledge
Curriculum	
Competency	
University	College
Department	
ProgramOfStudy	
Course	
KeyConcept	
AttainmentGoal	AttainmentLevel
Syllabus	
Instructor	Professor
Schedule	LectureRoom
LearningOjective	LearningMaterial
TeachingMethod	LearningMethod
Concept	
LearningConcept	FundamentalConcept
AdvancedConcept	RelatedConcept

Course Ontology by Zeng, Zhu & Ding
Curriculum
Topic
Knowloedge Area
Theorem
Example
Learning Goal

SWRC Ontology [90]
Publication
Person
Organization
Event
Topic
Project

Univ_Edu_Onto by Oprea [34]
Course
Curriculum
Objective
Resource
Didactic_Method
Course_Overview
Content
Examination
Test
Basic_Concept
Advance_Concept
Course_Evaluation
Professor

Univ-CS
Organization
Student
Faculty
Publication
Work

AIISO
Course
Module
Programme
Study
Organization
Person

Univ-Bench
Organization
Person
Employee
Student
Publication
Work

instOntology
Centre
Person
Course_Name
Event
Publication

Table 3: Important terms in existing ontologies

Furthermore, we analyzed a wide range of textual syllabuses available on the web and advised templates included in the Accreditation Template of Hellenic Quality Assurance & Accreditation Agency (HQA) [94] that establishes the procedure of academic accreditation of programs of study in Higher education in Greece, according to Law 4009/11. We also referenced the *Computer Science Curricula 2013* [95], a report from the ACM/IEEE-CS joint task force which provides important guidance on curricular structure and development for undergraduate programs in the area of Computer Science.

Finally, we concluded in a comprehensive list of all important terms within the subjects we are interested in. *Table 4* shows the list of terms divided in two sets.

Curriculum	**Syllabus**
Curriculum	Course
Educational Organization	Topic
University	Concept
Department	Syllabus
Discipline	Teaching Unit (Module)
Knowledge Body	Event
Knowledge Area	Instructor (Professor)
Knowledge Unit	Student
Learning Competency	Learning Outcome
ProgramofStudy (Academic Degree)	Teaching Method
Person	Learning Method
Course	Learning Material
Publication	Evaluation

Table 4: List of important terms within the subjects of Curriculum, Course and Syllabus of CCSO

3.4.3. DEFINE CLASSES AND CLASS HIERARCHY

This step and the next one - *defining classes and hierarchy* and *describing properties of classes* - are probably the most important in building process as well as closely intertwined.

In order to define the classes, we referenced the list of terms we built before (*Table 4*) and underline the main concepts of these terms. Main concepts may represent the super-classes in our ontology (e.g. *Discipline*, *EducationalOrganization*, *ProgramofStudy*, *Person*, *Course*, *Syllabus*, *Event*) with the remainder to be sub-classes or standalone classes but this will be more clear during hierarchy definition. We also took into consideration some tips for avoiding errors during class definition process [83]: classes represent concepts so despite the synonyms or several names in multiple languages they represent the same class; a class can also model abstract/mental concepts if needed; keep a balance and decide wisely whether a distinction is modeled as a class or an attribute (property).

Three formal approaches about developing a class hierarchy have been found in literature [83]:

1. The **top-down** approach goes from the "Whole" to the "Part". The development starts with the definition of the general concepts in the domain and subsequent specialization of remainder concepts.
2. The **bottom-up** approach goes from the "Part" to the "Whole". The development starts with the definition of the most specific concepts and subsequently grouping these into more general concepts.
3. Also there is a **combination** development process of the two approaches (*top-down* and *bottom-up*). It starts with the definition of the most salient concepts and subsequently specializing or generalizing the related concepts.

In this thesis, we followed the top-down development process to organize concepts, starting from the "Whole" to the "Part", going from the "root" to the "leaves". We adopted the multilevel design approach in order to achieve better understanding and distinct view of the ontology's entities and the taxonomy. The conceptualized concepts are organized into 3 layers, which are named as *Curriculum*, *Syllabus* and *Topic*. Each layer includes the semantic entities - classes, properties, individuals and relations - defined in that level. Ontology in upper layer establishes connections with one or more lower layer ontologies, i.e. curriculum ontology with one or more syllabus ontologies, syllabus ontology with one or more topic ontologies.

This ontology aims to be used from learners and educators as a tool for search and navigation as well as from information systems as knowledge component and so it should be able to describe in detail the concepts and entities that can be found in the specified subjects without growth or multiplied unnecessarily. In addition, ontology mappings have been established for the alignment of three main concepts with relevant terms in international ontologies and vocabularies in order to enable easier discovery of content by other organizations and search engines. During the taxonomy design we tried to follow the

principle of good shape and its criteria, i.e. *Law of Good shape* (*Law of Pragnanz*[25]) and *Ockham's razor principle* (*Law of Parsimony*[26]) [27], and avoid errors such as *circularity error, semantic inconsistency error, redundancy error* or *incompleteness of taxonomy* [83].

Some researchers noticed that it's easy for teachers to identify the concepts of a course but it's difficult to explicitly express the relationships within [82]. Therefore, they propose to keep as minimal as possible the set of conception relationships that created for use. *Dicheva & Dichev* [96] suggested to establish five types of relationships: '*superclass-sub-class*', '*class-instance*', '*super-sub*', '*relevant to*' and '*mentioned by*'. *Boyce & Pahl* in their work created two small sets of relationships [79]. The first set is used for representing the concept space linking the concepts between each other: '*Is-a'*, '*HasSubtype*', '*HasPart*', '*IsBasisOf*', '*HasConstraint*' and '*HasFunction*'. The second set includes relations that used for linking a concept to some content: '*HasDefinition*', '*HasSynonym*', '*HasAsExample*' and '*HasFurtherExplanation*'.

The kinds of relationships that used in our ontology are:

- '`superclass`': Predefined property that is used as a predicate in a statement to declare that a class is a generalization of other more specialized class or classes. A superclass can be a generalization of multiple sub-classes.
- '`sub-classOf`': Predefined property that is used as a predicate in a statement to declare that a class is a specialization of another more general class. A sub-class can be a specialization of multiple superclasses.
- '`hasPart`': indicates that a concept comprises a number of concepts in a 'Part-Whole' relation.
- '`partOf`': the inverse relation of '*hasPart*', indicates that a concept is part of a broader or bigger concept. '`belongsTo`' relation expresses also this kind of relationship.

[25] Law of Pragnanz: 'Good' means regular, complete, balanced and symmetrical.
[26] Law of Parsimony: entities should not be multiplied unnecessarily.

- ‘`hasSubtype`’: the inverse relation of ‘*Is-A*’ that suggested to avoid it due to the danger of pitfalls [79].

Class hierarchy of Curriculum Ontology

As noted above, we started the hierarchy design process with advising the list of important terms (*Table 4*) and underlining the basic concepts of these terms. Basic concepts may represent the super-classes in our ontology and will be the root of their own sub-trees. When describing the properties of classes, in the next step, these sub-trees will link to each other forming that way the taxonomy of Curriculum Ontology.

The main concepts of the important terms within the subject of Curriculum (*Table 4*) supposed to be:

- Field of Study
- Curriculum
- Educational Organization
- Program of Study
- Person
- Course

Taxonomy of FieldofStudy sub-tree

In Computer Curricula 2013 the organization of knowledge structure is adequately defined [95]. *Figure 35* presents the taxonomy of the sub-tree composed of `FieldofStudy` superclass, `Discipline` class and the remainder classes, followed by information about the classes.

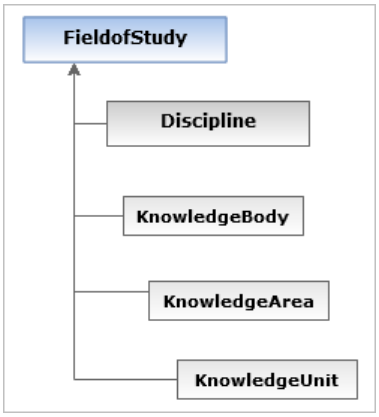


Figure 35: Taxonomy of FieldofStudy sub-tree

FieldofStudy class represents the root of the sub-tree, the root-class in the structure of knowledge.

Discipline class represents academic discipline. An *Academic Discipline* is a branch of knowledge or a field of study. The branches of science are commonly mentioned as the scientific disciplines (*e.g. Computer Science*, *Mathematics*, *Physics*, *etc*)[27].

KnowledgeBody class defines the main knowledge body of a Discipline, a specification of the content to be covered in a curriculum and establishes a '*partOf*' relationship with *Discipline* class. In CS2013 is presented as a set of *Knowledge Areas*.

KnowledgeArea class defines the subjects of a *Body of Knowledge*, it establishes a '*partOf*' relationship with *KnowledgeBody* class and is further organized into a set of *Units of Knowledge*. It is possible the topics within a Knowledge Area to organized into different courses with different approaches in different institutions.

KnowledgeUnit class defines the part of a *Knowledge Area*, so it establishes a '*partOf*' relationship with *KnowledgeArea* class.

For example, a specific course titled as '*Java Programming I*' has a subsumption relation with an instance of `KnowledgeUnit` class as '*Object-Oriented Programming*' which has a subsumption relation with an instance of `KnowledgeArea` class as '*PL/Programming Languages*' which has a subsumption relation with an instance of `KnowledgeBody` class as '*Computer Science & Software Engineering*'.

Taxonomy of EducationalOrganization sub-tree

Ontology mapping with terms or elements in shared international vocabularies can help the alignment and classification of the ontology as well as the reuse and dissemination of data.

EducationalOrganization has the role of the root-class in its own sub-tree and defines an organization designed to provide educational services and learning environments. This

[27] Outline of Academic Disciplines (Wikipedia): en.wikipedia.org/wiki/Outline_of_academic_disciplines

class establishes an ontology alignment with '`EducationalOrganization`'[28] type from Schema.org vocabulary, a specialized subtype of '`Organization`'[29] type for easier discovery and reusing. Properties of the shared type can be useful in the description of the sub-classes in the sub-tree as they inherit these properties from the parent class, i.e. `identifier`, `name`, `logo`, `address`, `location`, `workPhone`, `faxNumber`, `email`, `employee`, `numberOfEmployees`, `alumni`, *etc.* *Figure 36* presents the taxonomy of the sub-tree composed of `EducationalOrganization` superclass and its sub-classes. The sub-classes of the sub-tree are linked directly to '*EducationalOrganization*' class. The relationships between the sub-classes of the sub-tree are modeled using the '*belongsTo*' relation instead of '*rdfs:sub-classOf*' which would not be right. The hierarchy relationship '*rdfs:sub-classOf*' represents a '*is-A*' relation*: a class A is a sub-class of B if every instance of A is also an instance of B* [30].

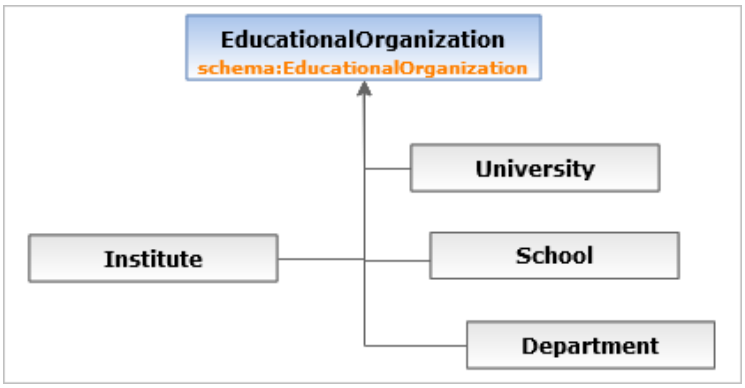


Figure 36: Taxonomy of EducationalOrganization sub-tree

`University`[31] class defines an institution of higher education and research which awards academic degrees in a set of academic disciplines. It presented as sub-class of '*EducationalOrganization*' class.

`School`[32] is sub-class of '*EducationalOrganization*' class and defines a division within a university comprising one subject area, or a number of related subject areas. We followed the

[28] *EducationalOrganization* type in Schema.org: schema.org/EducationalOrganization
[29] *Organization* type in Schema.org: schema.org/Organization
[30] Property *rdfs:subClassOf*: www.w3.org/TR/rdf-schema/#ch_subclassof
[31] University (Wikipedia): en.wikipedia.org/wiki/University

U.S. terminology where such divisions are referred to as *Colleges* or *Schools* while the term *Faculty*, which is used in some other countries, defined as synonym to the term *Academic staff* of a University. We used the relationship '*belongsTo*' to model the relation between School and University, i.e. `School`-*belongsTo*-`University`.

`Department`[33] is sub-class of '*EducationalOrganization*' class and defines a division of a School (or College) devoted to a particular academic degree, thus it establishes a '*belongsTo*' relation with *School* class (`Department`-*belongsTo*-`School`).

`Institute` is sub-class of '*EducationalOrganization*' class and used to describe an autonomous educational institution which provides learning services in areas of Non-Formal Education such as *Lifelong Education*, *Vocational Education*, *etc.*

Taxonomy of ProgramofStudy sub-tree

The **`ProgramofStudy`** sub-tree enables the classification of the degrees awarded to the students upon a successful completion of a course of study in an educational institution. The sub-tree includes five sub-classes as presented in *Figure 37*. The sub-classes of *ProgramofStudy* are disjoint to each other (e.g. a Bachelor degree cannot be a Master degree).

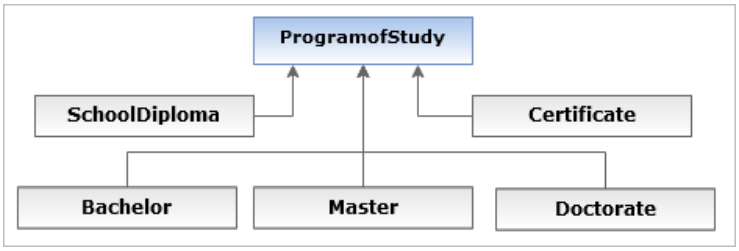


Figure 37: Taxonomy of ProgramofStudy sub-tree

- **`SchoolDiploma`** class expresses graduation diplomas in Compulsory Education, provided by Primary and Secondary schools (Primary School, Secondary School and High School).

---

[32] Faculty (Wikipedia): en.wikipedia.org/wiki/Faculty_(division)
[33] Wikipedia: en.wikipedia.org/wiki/Academic_department

- **`Certificate`** class expresses the diplomas that cannot be classified to the other sub-classes such as Technical/Vocational Diploma, Teaching Diploma, Language Diploma, *etc.*
- **`Bachelor`** class expresses the undergraduate academic degree awarded by universities and colleges, known as Bachelor's degree.
- **`Master`** class expresses the postgraduate academic degree, known as Master's degree, awarded by universities or colleges upon completion of a course of study demonstrating mastery of a specific field of study or area of professional practice.
- **`Doctorate`** class expresses the academic and research degree, also known as Doctor's degree, awarded by universities that certifies the person as an expert of art or sciences in a field and qualifies to teach at the university level.

Taxonomy of Person sub-tree

The *Person* sub-tree represents the several types of persons affiliated with an educational organization. We focus on persons associated with teaching and learning concepts (i.e. professors, students, assistants). *Figure 38* depicts the taxonomy of the sub-tree composed of `Person` superclass and its sub-classes.

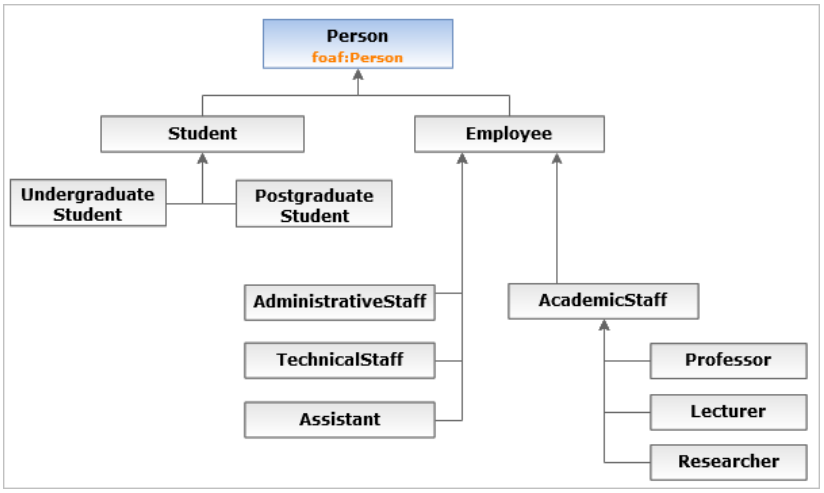


Figure 38: Taxonomy for Person sub-tree

Person class is the root-class of this sub-tree and, in our project, represents a human being who is either employee or student in an institution. We established an alignment to shared type '`Person`'[34] in international FOAF vocabulary for easier discovery and reusing by others. The properties of the shared type were used for the description of the sub-classes and their instances (e.g. *firstName*, *lastName*, *familyName*, *title*, and *gender*) while some additional properties has also defined (e.g. *personID*, *profilePic*).

Employee class is sub-class of '*Person*', represents all kinds of employees of an organization, e.g. Professors, Officers, Technicians, *etc*.

AcademicStaff[35] class is sub-class of '*Employee*', used to describe the academic staff of an institution, a school or a university such as professors of various ranks, lecturers and/or researchers. In North America the term *Faculty* is used instead, thus we defined it as synonym using the annotation property `skos:altLabel`.

Professor class is sub-class of '*AcademicStaff*' class, used to describe an instructor that is an expert of art or sciences and participates in a program of study of an educational institution. There are several ranks[36] of professors in a university such as *Assistant professor*, *Associate professor*, *Visiting professor*, (full) *Professor*, *etc*. We chose to model this distinction as an attribute value of class `Professor` rather than sub-classes since the rank distinction does not has any particular implications in the relationships with other entities. If this characteristic gets higher importance, in future use of the ontology, then it could be introduced as separate class.

Lecturer class is sub-class of '*AcademicStaff*' class too, used to describe an academic person at an early career stage with open-ended or tenured position in a university who teaches and conducts research.

Researcher class is sub-class of '*AcademicStaff*' class too, used to describe an academic person that conducts research, participates in research projects or leads research groups.

[34] Person type in FOAF: http://xmlns.com/foaf/spec/#term_Person
[35] Wikipedia: https://en.wikipedia.org/wiki/Faculty_(academic_staff)
[36] List of Academic ranks in Wikipedia: https://en.wikipedia.org/wiki/List_of_academic_ranks#Greece

There are several ranks of researchers like *Research assistant*, *Research associate*, *Postdoctoral researcher*, *etc* but such detail is unnecessary in our project.

`Student` class is sub-class of '*Person*' class and describes a learner who attends an educational program. The label '*Pupil*' is used as synonym. The class has two sub-classes: '**`UndergraduateStudent`**' for students who attend university programs up to the level to the Bachelor's degree and '**`PostgraduateStudent`**' for those who own a Bachelor's degree and studying for an academic or professional certificate.

`AdministrativeStaff` class is sub-class of '*Person*' class and describes an employee who conducts of the administrating of an educational organization.

`Assistant` class is sub-class of '*Person*' class and describes a person, usually a postgraduate student, who helps by assuming duties in lessons (Teaching Assistant), projects (Project Assistant), research (Research Assistant), *etc.*

`TechnicalStaff` class is sub-class of '*Person*' class and represents an employee who is specialized and related to provide practical, mechanical and applied services (e.g. electrician, plumber, *etc*).

It must be noted, that some classes, i.e. `AdministrativeStaff`, `TechnicalStaff`, `Lecturer` and `Researcher`, are included mainly for classification reasons as well as for future extension of the ontology.

The Course class

In Higher/Tertiary Education, a course is an institutional unit of teaching that lasts one academic term with a specific subject, usually is led by one or more instructors and has a fixed group of students. A course can be defined as a set of syllabi and covers a variety of KnowledgeUnits (KUs) and Topics within different KnowledgeAreas (KAs).

`Course` class is a top-level class without sub-classes since in our approach the distinction of the course types (*DesignCourse*, *MajorCourse*, *CoreTier1*, *CoreTier2*, *ElectiveCourse*) is modeled as an attribute value rather than as sub-class since it sufficient for answering the related competency questions and course distinction does not has any particular implications

with other entities. An instance of Course class can establish connection with one or more instances of Syllabus class (1:n cardinality).

Below you can see at a glance the list of classes of Curriculum ontology and their hierarchy. *Figure 39* shows the taxonomy of classes of Curriculum ontology and *Table 5* lists the classes of Curriculum ontology.

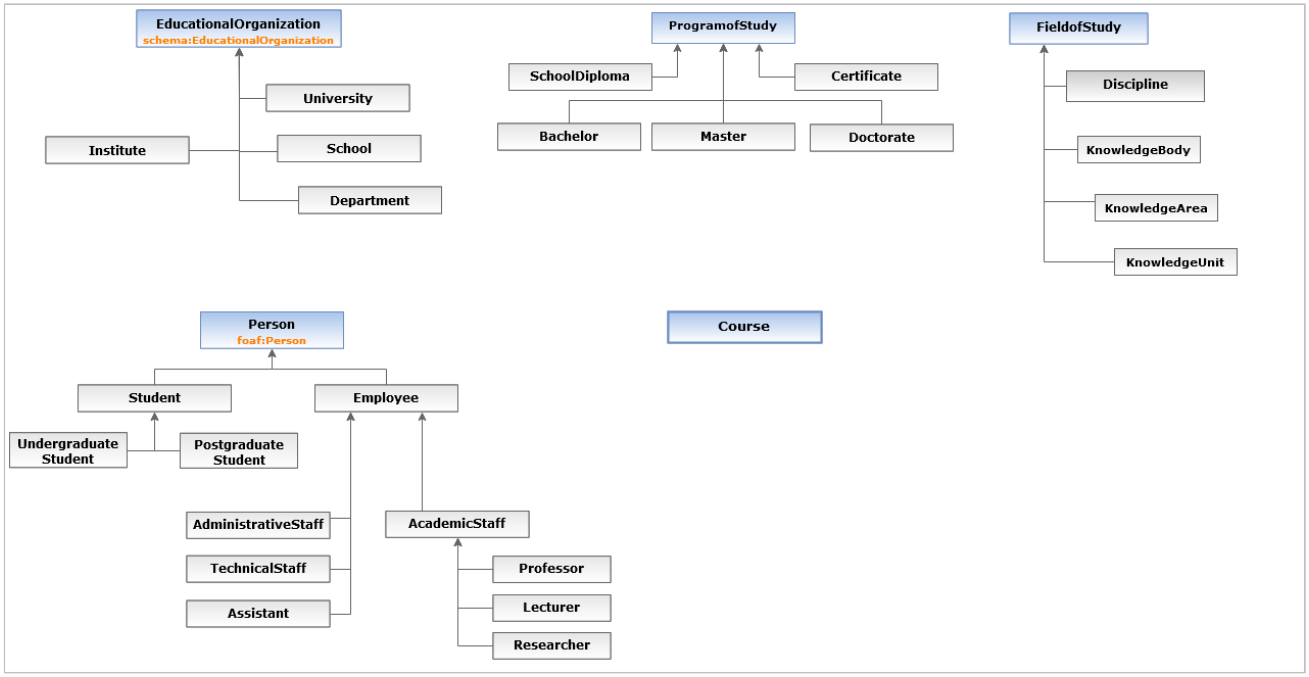


Figure 39: Class hierarchy of Curriculum ontology

Class	SuperClass	Sub-class	Definition
FieldofStudy	-	Discipline	Fields of Study, Academic Disciplines
Discipline	FieldofStudy	-	An Academic Discipline
KnowledgeBody	FieldofStudy	-	Main Knowledge body of a Discipline
KnowledgeArea	FieldofStudy	-	Subjects of a Body of Knowledge
KnowledgeUnit	FieldofStudy	-	Category of a Knowledge Area
EducationalOrganization= schema.org: EducationalOrganization	-	University, Institute, School, Department	An organization designed to provide educational services and learning environments.
University	EducationalOrganization	-	An institution of higher education and research.
School	EducationalOrganization	-	A division within a university comprising one subject area, or a number of related subject areas.

Department	EducationalOrganization	-	A division of a College (or School) devoted to a particular academic degree.
Institute	EducationalOrganization	-	An autonomous educational institution which provides learning services in areas of Non-Formal Education.
ProgramofStudy	-	SchoolDiploma, Certificate, Bachelor, Master, Doctorate	The degrees awarded to the students upon a successful completion of a course of study in an educational institution.
SchoolDiploma	ProgramofStudy	-	The graduation degrees provided by Primary and Secondary schools (Primary School, Secondary School and High School).
Certificate	ProgramofStudy	-	The diplomas that cannot be classified to the other sub-classes such as Vocation diploma, Teaching diploma, Languages diploma.
Bachelor	ProgramofStudy	-	The undergraduate academic degree awarded by universities and colleges.
Master	ProgramofStudy	-	The postgraduate academic degree, awarded by universities or colleges upon completion of a course of study demonstrating mastery of a specific field of study or area of professional practice.
Doctorate	ProgramofStudy	-	The academic and research degree, awarded by universities that certifies the person as an expert of art or sciences in a field and qualifies to teach at the university level.
Person= FOAF:Person	FOAF:Person	Employee, Student	A human being who is either employee or student in an institution.
Employee	Person	AcademicStaff, AdministrativeStaff, Assistant, TechnicalStaff	Represents all kinds of employees of an organization.
AcademicStaff	Employee	Professor, Lecturer, Researcher	Academic staff of an Institute

Professor	AcademicStaff	-	An instructor that is an expert of art or sciences and participates in a programme of study.
Lecturer	AcademicStaff	-	An academic person at an early career stage.
Researcher	AcademicStaff	-	An academic person that conducts research, participates in research projects or leads research groups.
Student	Person	UndergraduateStudent, PostgraduateStudent	A learner who attends an educational programme.
UndergraduateStudent	Student	-	Students who attend university programs up to the level to the Bachelor's degree.
PostgraduateStudent	Student	-	Students who own a Bachelor's degree and studying for an academic or professional certificate.
AdministrativeStaff	Employee	-	An employee who conducts of the administrating of an educational organization.
Assistant	Employee	-	A person who helps by assuming duties in lessons, projects, research, etc.
TechnicalStaff	Employee	-	Technical staff of an Organization.
Course	-	-	An institutional unit of teaching that lasts one academic term with a specific subject, usually is led by one or more instructors and has a fixed group of students.

Table 5: The Classes of Curriculum ontology

When the definition of classes and class taxonomy is completed, we start building our ontology in Protégé.

First we create the top-level classes, the superclasses of the ontology (*FieldofStudy*, *EducationalOrganization*, *Person*, *ProgramofStudy*, *Course*) started from Protégé's root class *Thing*. Subsequently, we created the sub-classes of each superclass's sub-tree. Every class is provided with comments using available annotation properties *rdfs:comment*, *skos:altLabel*, *rdfs:Label*, *etc*, as shown in *Figure 40*. All annotation properties of classes are written in two languages, Greek and English, giving the ontology a bilingual character.

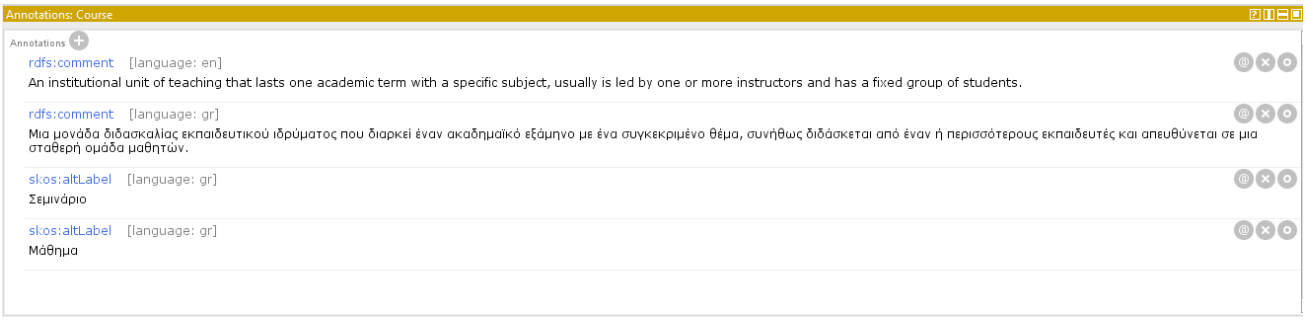


Figure 40: Annotation properties of Course class

Class hierarchy of Syllabus Ontology

The syllabus is a kind of blueprint that addresses course information, expectations and responsibilities to students. It is considered to be the most appropriate way to introduce a course to students; a kind of "*contract between faculty members and students*" designed to inform them about a course. A syllabus contains important information about a course like learning topics and objectives, teaching methods, outcomes, evaluation procedures, grading policy, bibliography, study resources, *etc*. It is usually created by course designers or instructors using predefine formats or templates.

Our designed Syllabus ontology, conceptualizes the internal and external entities associated with a course as described in Curriculum ontology. The ontology is designed to provide high reusability and adaptability by covering a variety of courses as well as searchability and exchangeability of syllabuses among heterogeneous software systems in different institutions. The target is to provide a syllabus ontology which can be reused by a variety of institutions for their courses. A list of important terms within the Syllabus subject has already presented before in *Table 4*.

`Syllabus` is the core class of which all other classes are defined. Syllabus class establishes connections with classes of Curriculum ontology (e.g. *Course*, *Professor*, *Assistant*, and *Student*) as well as Topic ontology (e.g. *Event*, *Publication*). A large amount of the available information about a course is attached to syllabus class with its data properties.

`Event` is a superclass that meant to model all events that happen in a class during a course. Event class has two sub-classes, `TeachingEvent` and `EvaluationEvent`. **`TeachingEvent`** class expresses all events that happen in a class which are related to teaching. **`EvaluationEvent`** class expresses all events included in the evaluation procedure of the

students. Moreover, it can be expanded in order to include more types of events that taking place in an institution, like lecture or conference.

TeachingEvaluationMethod class describes the different kinds of teaching methods and learning activities within a course, such as lecture, seminar, assessment, project, laboratory practice, fieldwork, study and analysis of bibliography, essay writing, tutorials, placements, art workshop, interactive teaching, educational visits, examination, *etc*. Some of these methods are also being used in evaluation procedure (e.g. project, assessment, examination, *etc*).

Topic class describes a subject of study, a small section from Body of Knowledge that could be organized into course and covered during lessons. A Topic may be covered in several Courses included in different instances of ProgramofStudy.

LearningOutcome class describes the learning outcomes, specific knowledge, skills and competences, which the students will acquire with the successful completion of the course. In other implementations, the term "*Learning Objective*" is also used instead.

Resource class models teaching and learning resources that are being used during a course, as well as publications that have instances of *Person* class as authors and in many cases can also be used as teaching/learning material. Books, academic articles, notes as well as webpages, videos, images, audios and other stuff with educational interest for a course can be used as teaching/learning material in lessons.

When definition of classes and taxonomy of Syllabus ontology had completed, the building our ontology continued in Protégé with the creation of new classes and their hierarchy.

Figure 41 below, shows the class hierarchy of Syllabus ontology while *Table 6* lists the classes of Syllabus ontology.

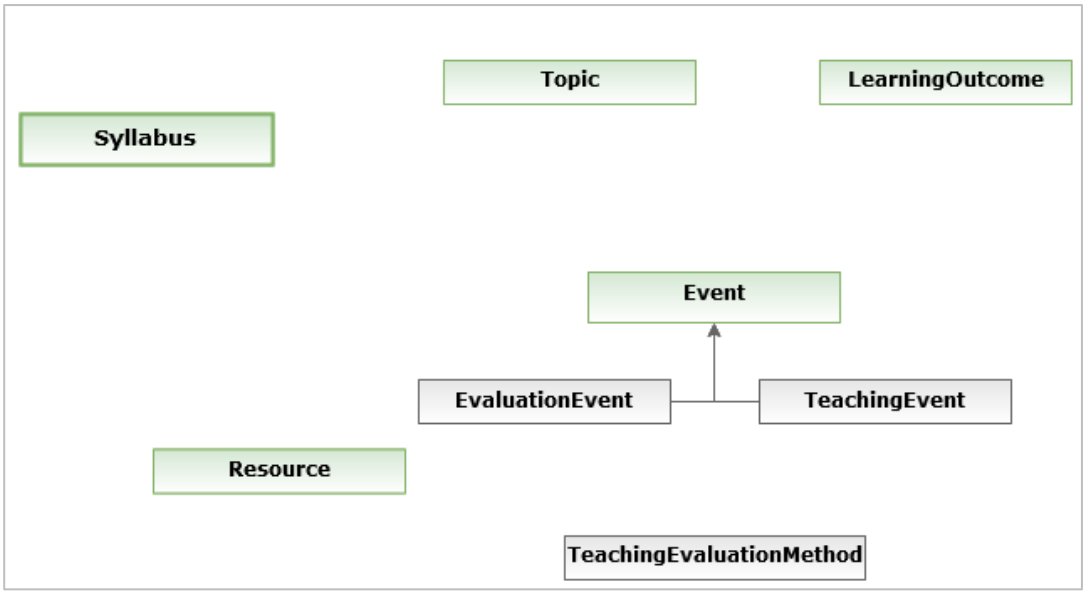


Figure 41: Class hierarchy of Syllabus ontology

Class	SuperClass	Sub-class	Definition
Syllabus	-	-	A document that addresses course information, expectations and responsibilities to students.
Event	-	TeachingEvent, EvaluationEvent	All events that happen in a class within a course.
TeachingEvent	Event	-	Events that happen in a class which are related to teaching.
EvaluationEvent	Event	-	Events included in the evaluation procedure.
TeachingEvaluation-Method	-	-	The manner and the different kinds of teaching methods and learning activities within a course.
Topic	-	-	A subject of study, a small section from Body of Knowledge that could be organized into course and covered during lessons.
LearningOutcome	-	-	Learning outcomes, specific knowledge, skills and competences, which the students will acquire with the successful completion of the course.
Resource	-	-	Teaching and learning resources that are being used during a course. Also, Publications of a Person.

Table 6: Classes of Syllabus ontology

The integrated CCSO ontology

Finally, the integrated educational ontology currently includes the classes without any object or data properties, which they will be defined in the next steps, as shown in *Figure 42*. The name that has given is CCSO, which are the first letters of its core concepts, i.e. *Curriculum-Course-Syllabus-Ontology*.

CCSO comprises of 41 concepts in a taxonomy, 9 of which are the top-level concepts of the ontology, namely *FieldofStudy*, *EducationalOrganization*, *ProgramofStudy*, *Person*, *Course*, *Syllabus*, *Event*, *Topic* and *Resource*. *Figure 43* depicts the classes and their hierarchy in Protégé.

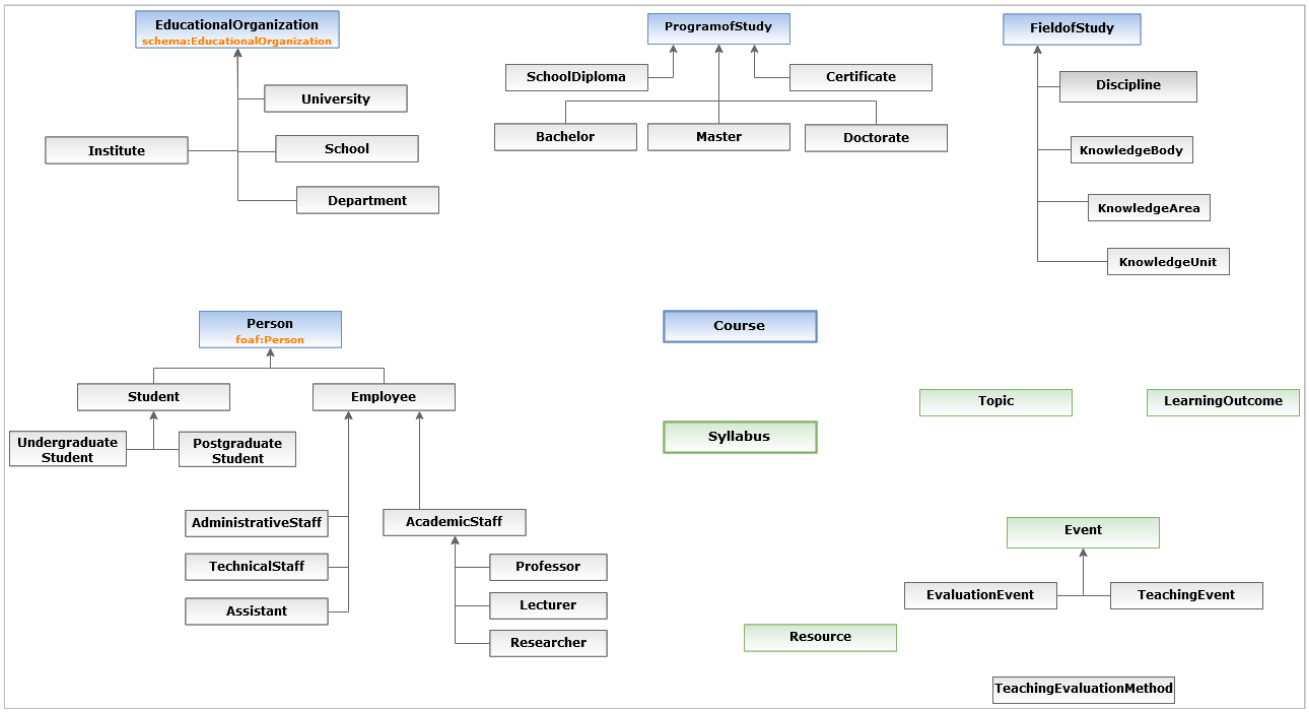


Figure 42: Classes and class hierarchy of CCSO ontology

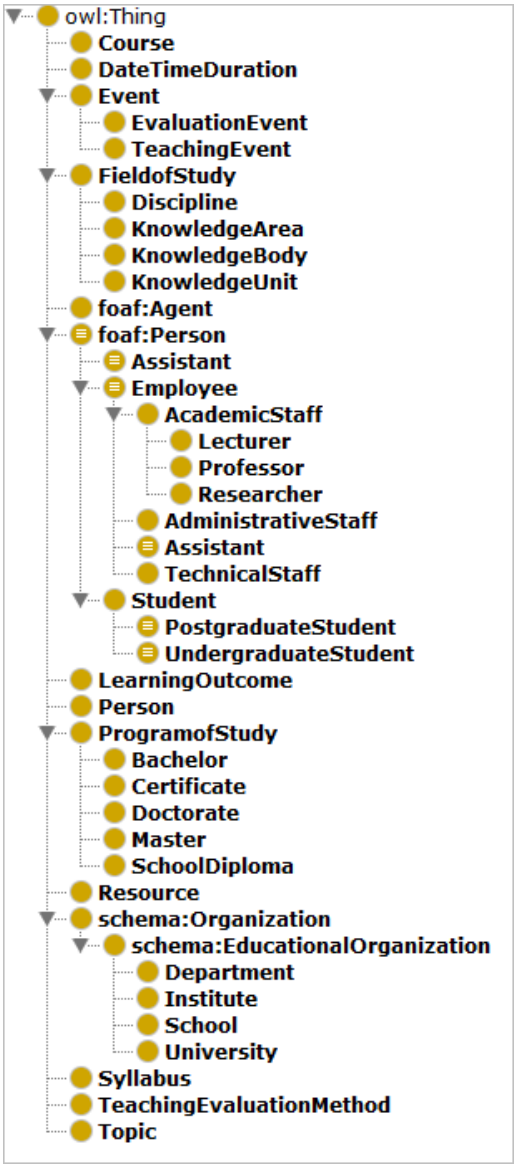


Figure 43: Classes and their hierarchy of CCSO ontology in Protégé

3.4.4. Describe the Properties of Classes

Once we have defined the classes, the next step is to describe the internal structure of ontology taxonomy and specify the properties of classes. The properties that describe a class also called *slots* of that class. Protégé defines properties of classes in two categories: *Object properties* and *Data properties*.

- **Object property** describes a relationship to other individuals which can be instances of the same class or other classes. Thus, an object property also constitutes the relationship with other classes. For example, the class `Course` has an *object property* named `hasSyllabus` which establishes relations with one or more Syllabus classes.

Thus, its values are instances of the Syllabus class and and is decrared in the field *Range* of the object property, in Protégé.

- **Data property** contains information about an individual, an instance of the class, without any relation to others. For example, in class `Syllabus`, the value of the property `csName` is string that expresses the name of the syllabus while the property `weeklyHours` expresses the required hours in a week for all the activities of the course which has the syllabus.

Considering the class inheritance, all sub-classes of a class **inherit the properties from** the parent class. Thus, each property is attached to the most general class which can have that property. For example, all the sub-classes of class `FieldofStudy` will inherit the data property `fsName`, as shown in *Figure 44*.

In the next pages, the sub-trees of the top-level concepts are presented one-by-one, enriched with the properties of classes. Each class represented as a shape that contains three boxes: the first one with the class' name, the second box contains the Object properties of the class and the third box contains the Data properties of the class. The inherited properties are written in parenthesis. When a class has no properties, then the box has left empty.

Properties for classes of FieldofStudy sub-tree

In class **`FieldofStudy`** one object property and two data properties have been added, namely `partOf`, `fsName` and `abbreviation` which are inherited to all of its sub-classes.

In class `KnowledgeArea`, the object property `areaOfCourse` has been added that describes the relation "*Knowledge Area is area of a Course*" and it's inverse property of `coversKA`. In class `KnowledgeArea` the object property `lists` has been added that describes the relation "*KU lists a set of Topics*" and "*KU lists a set of Learning Outcomes*". It's inverse property of `listedInKU`.

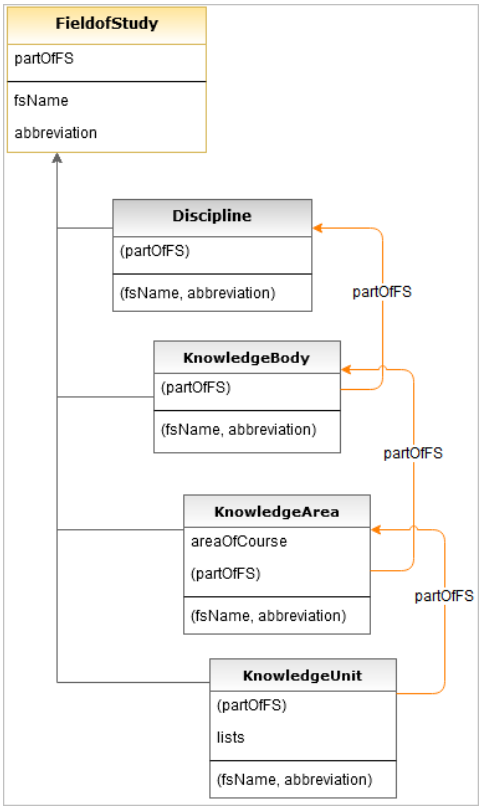


Figure 44: Class hierarchy and properties of FieldofStudy sub-tree

Properties for classes of EducationalOrganization sub-tree

`EducationalOrganization`, the root-class in this sub-tree, inherits the necessary data properties from the shared type '`schema.org:EducationalOrganization`'[37], namely `identifier`, `legalName`, `logo`, `address`, `location`, `workPhone`, `faxNumber`, `email` and `url` as shown in *Figure 45*. We attach three object properties in this class:

- `belongsTo` establishes the relations "*School belongsTo University*" and "*Department belongsTo School*".
- `providesProgram` describes the relations "*EducationalOrganization provides StudyProgram*", "*School provides StudyProgram*" and "*Department provides StudyProgram*". It's inverse property of `providedBy`.

[37] *EducationalOrganization* type in Schema.org: http://schema.org/EducationalOrganization

- `hasEmployee` describes the relation "*EducationalOrganization has Employee*" and it's inverse property of `worksFor`.

The rest classes of the sub-tree inherit their data properties from the root-class. The class `Department` has four object properties, the inherited one and three additional properties:

- `hasAcademicStaff` describes the relation "*Department has Academic Staff*" and it's inverse property of `memberOf`.
- `offersCourse` describes the relation "*Department offers Course(s)*" and it's inverse property of `offeredBy`.
- `hasAlumnus` describes the relation "*Department has alumnus*" and it's inverse property of `alumnusOf`.

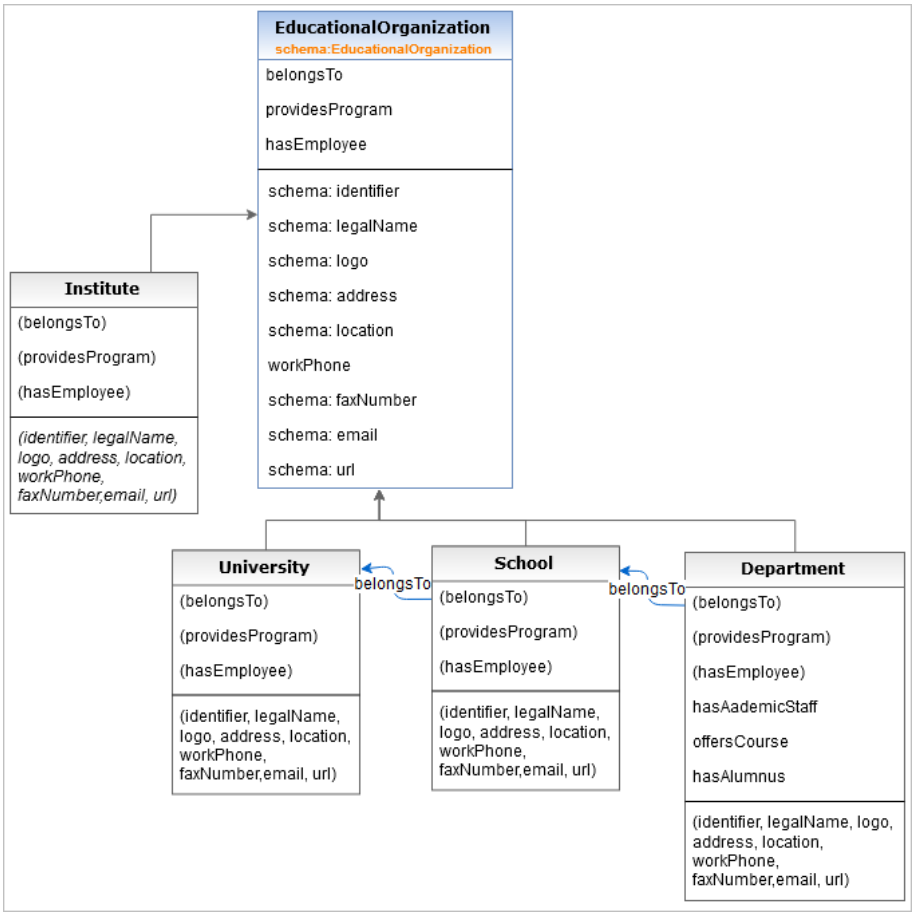


Figure 45: Class hierarchy and class properties of EducationalOrganization sub-tree

Properties for classes of ProgramofStudy sub-tree

`ProgramofStudy` class has enriched with seven data properties and five object properties which express relationships with other classes, as shown in *Figure 46*. The data properties

are: `code`, `psName`, `degreeTitle`, `degreeType`, `duration`, `degreeURL` and `creditsECTS` while the object properties are:

- `providedBy`, describes the relation "*Study Program is provided by an Educational Organization*" and it's inverse property of `providesProgram`.
- `focusIn`, describes the relation "*Study Program is focused in a Body of Knowledge*".
- `includesCourse`, describes the relation "*Study Program includes Course*" and it's inverse property of `includedIn`.
- `hasRegistered`, describes the relation "*Study Program has registered Student*" and it's inverse property of `enrolledIn`.
- `requiresProgram`, describes the relation "*Study Program requires Study Program*" enable to model prerequisites between the various types of degrees, such as "*a Master program requires a Bachelor degree*", "*a Doctorate program requires a Master degree*".

There is no need to add any more properties to any of the sub-classes which inherit the object and data properties from the root-class of the sub-tree.

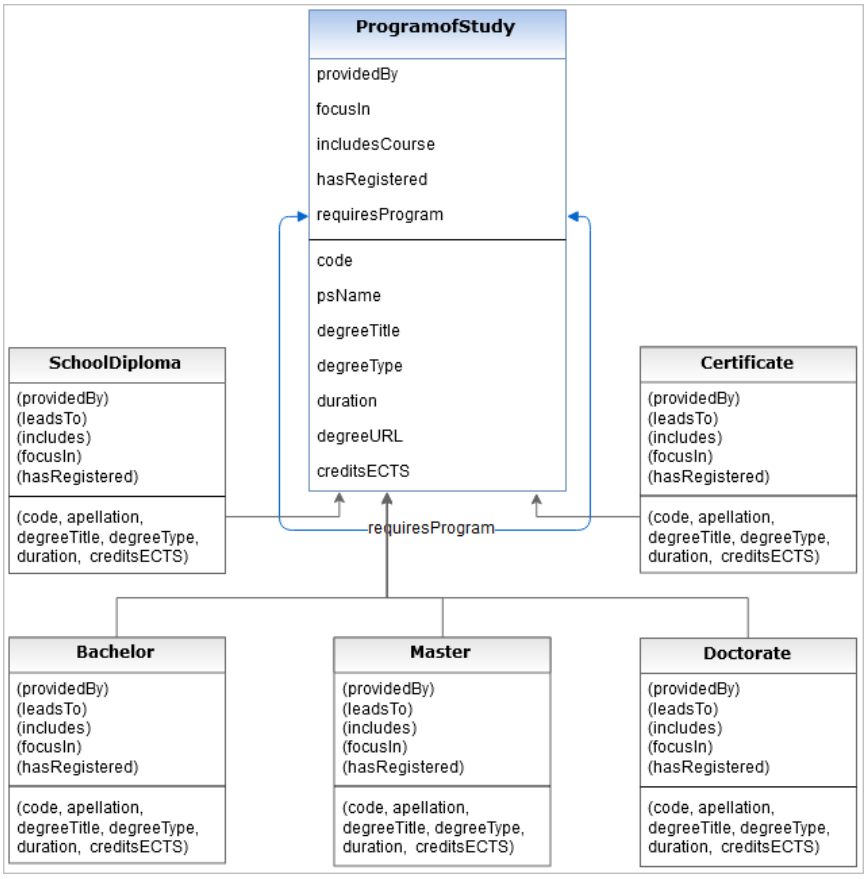


Figure 46: Class hierarchy and class properties of ProgramofStudy sub-tree

Properties for classes of Person sub-tree

`Person` is the root-class in this sub-tree and inherits some useful data properties from the shared type '`foaf:Person`'[38], namely `title`, `firstName`, `lastName` and `familyName`. In addition to these, some more data properties have been added: `personID`, `profilePic`, `gender`, `homePhone`, `workPhone`, `mobilePhone`, `faxNumber`, `email`, `workEmail` and `url`. As to the object properties, it has enriched with the properties:

- `hasDegree` describes the relation "*Person has Degree from a ProgramofStudy*".
- `alumnusOf` describes the relation "*Person is alumnusOf a Department*" and it's inverse property of `hasAlumnus`.
- `hasPublication` describes the relation "*Person has Publication*" with the class `Resource`.

[38] Person type in FOAF: xmlns.com/foaf/spec/#term_Person

All properties of Person are inherited to its sub-classes. In addition, in class Employee we add the object property worksFor (inverse property of hasEmployee) to describe the relation '*Employee works for an Organization*' and four data properties, namely staffSpecialization, staffPosition, officeNo and officeHours. *Figure 47* illustrates the class hierarchy and class properties of Person sub-tree.

Classes AdministrativeStaff and TechnicalStaff are included for classification reason and for future use, so it isn't described any relation with other entities.

In class Assistant two extra object properties has been added:

- assistsInCourse describes the relation "*An Assistant is (Teaching) Assistant in a Course*" and it's inverse property of hasCourseAssistant.
- assistsInEvent describes the relation "*An Assistant may assists in a teaching event*" or "*An Assistant may assists in an evaluation event*". It's inverse property of hasEventAssistant.

The class AcademicStaff has attached the object property memberOf which describes the relation "*Academic person that is member of the academic staff of a department*", it's inverse property of hasAcademicStaff and is inherited to its sub-classes, Professor, Researcher and Lecturer.

In class Professor two extra object properties have been attached:

- supervisorOf describes the relation "*Professor supervises a Course*" " and it's inverse property of hasSupervisor.
- instructorOf describes the relation "*Professor is instructor of a Course*" " and it's inverse property of hasInstructor.
- teachesIn, describes the relation "*Professor teaches in Teaching Events*" and it's inverse property of hasTeacher.

Class Student has attached two extra data properties, namely enrollmentDate and graduationDate, and three object properties:

- enrolledIn, describes the relation "*Student is enrolled in a Program of Study*" and it's inverse property of hasRegistered.

- `attendsCourse`, describes the relation "*Student attends Courses*" and it's inverse property of `hasAttender`.
- `hasCompleted`, describes the relation "*Student has successfully completed a Program of Study or a Course*" and it's inverse property of `hasCompleted`.

All classes, class hierarchy and class properties of `Person` sub-tree are illustrated in *Figure 47* below.

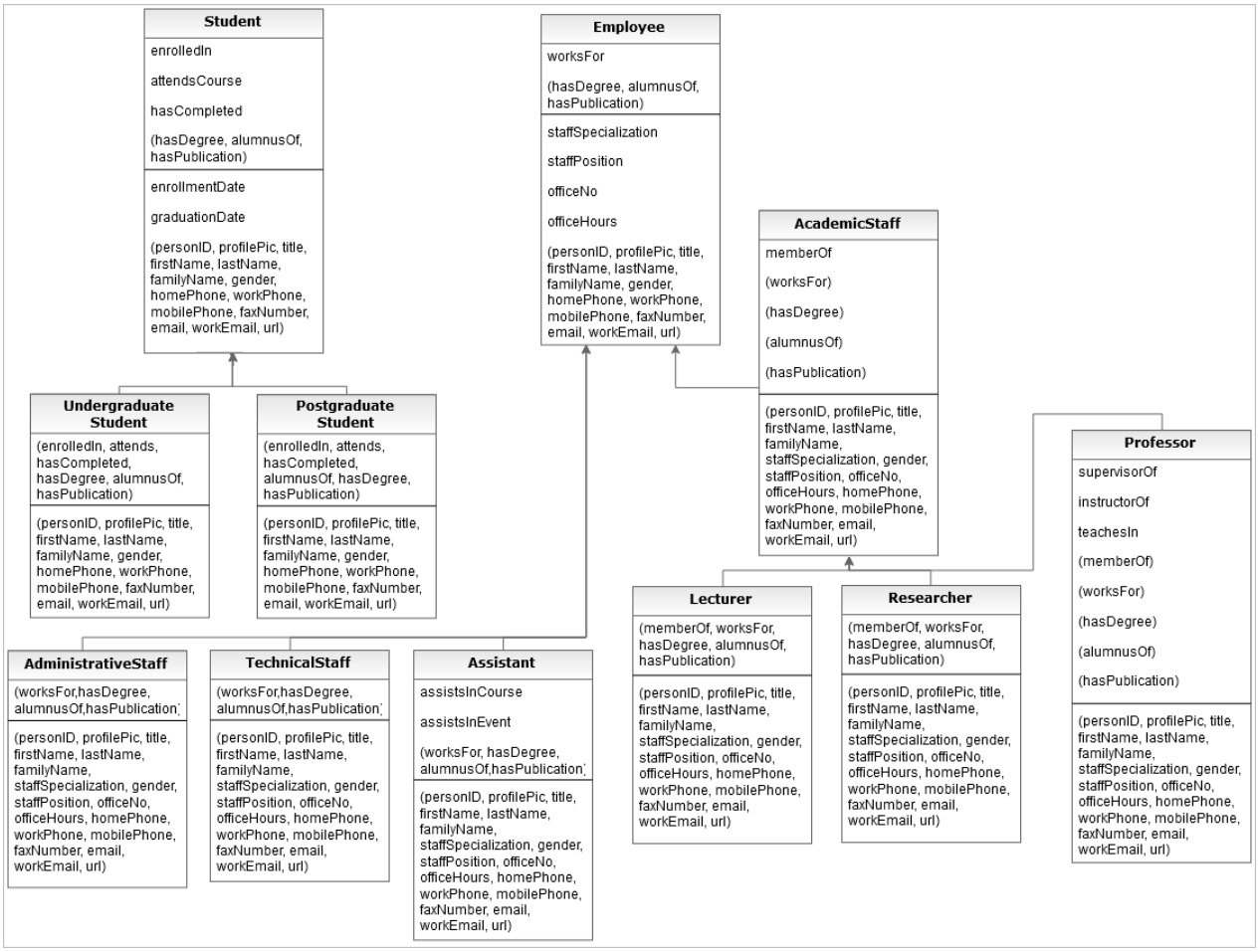


Figure 47: Class hierarchy and class properties of Person sub-tree

Properties for Course and Syllabus classes

The *Course* and *Syllabus* classes are both core classes in our ontology. In order to make these two concepts well described, we advised a variety of textual syllabuses available on the web and templates included in the Accreditation Template of Hellenic Quality Assurance & Accreditation Agency (HQA) [94].

In **`Course`** class two data properties has been attached, namely `code`, `csName` while in `Syllabus` class a large number of properties has been attached for describing a course in

detail: `code`, `csName`, `courseType`, `semesterinProgram`, `academicYear`, `language`, `offeredtoErasmus`, `startDate`, `endDate`, `duration`, `weeklyHours`, `weeklyClassSchedule`, `weeklyWorkload`, `creditsECTS`, `courseURL`, `eClassURL`, `versionNo`, `evaluationInfo` and `description`. *Figure 48* shows the data and object properties of *Course* and *Syllabus* classes.

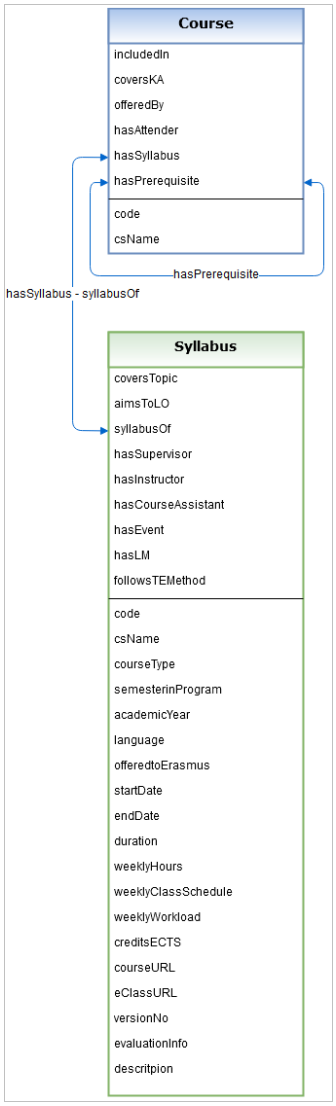


Figure 48: The properties of Course, Syllabus classes

Since they are core classes in our ontology, several object properties have been attached to them. In **`Course`** class has been attached the following:

- `includedIn` describes the relation "*Course is included in a Study Program*" and it's inverse property of `includesCourse`.

- `coversKA` describes the relation "*Course covers part of a Knowledge Area*" and it's inverse property of `areaOfCourse`.
- `offeredBy` describes the relation "*Course is offered by Department*" and it's inverse property of `offersCourse`.
- `hasAttender` describes the relation "*Course has Student as Attender*" and it's inverse property of `attendsCourse`.
- `hasSyllabus`, describes the relation "*Course has Syllabus*" and it's inverse property of `syllabusOf`.
- `hasPrerequisite`, describes the relation "*Course requires knowledge from other Course*".

In **`Syllabus`** class we add the following:

- `coversTopic` describes the relation "*Course covers Topic*" and it's inverse property of `coveredInCourse`.
- `aimsToLO` describes the relation "*Course aims to Learning Outcome*" and it's inverse property of `acquiredInCourse`.
- `syllabusOf` describes the relation "*It is syllabus of a Course*" and it's inverse property of `hasSyllabus`.
- `hasSupervisor` describes the relation "*Course has a Professor as Supervisor*" and it's inverse property of `supervisorOf`.
- `hasInstructor` describes the relation "*Course has a Professor as Instructor*" and it's inverse property of `instructorOf`.
- `hasCourseAssistant` describes the relation "*A Course has Assistant*" and it's inverse property of `assistsInCourse`.
- `hasEvent` describes the relation "*Course has Event*". A course is composed of some events, teaching events and evaluation events.
- `hasLM` describes the relation "*Course uses Resource as Learning Material*".
- `followsTEMethod` describes the relation "*Course follows Teaching Method*" and "*Course follows Evaluation Method*".

Properties of classes in Event sub-tree

In order to describe all events that taking place in a class within a course, the following data properties have been attached in Event class as the root-class in this sub-tree: `label`, `eventDateTime`, `eventWeekNo`, `duration`, `language`, `eventClassroom`, `workload` and `description`.

`Event` class has attached two object properties:

- `hasEventAssistant` describes the relation "*Event has Assistant*" and it's inverse property of `assistsInEvent`.
- `hasLM` describes the relation "*Event uses Resource as Learning Material*".

The two sub-classes of `Event` inherit its properties. *Figure 49* shows the object and data properties of class `Event` and its sub-classes and the `TeachingEvaluationMethod` class.

In class **`TeachingEvent`** four extra object properties have been attached:

- `hasTopic` describes the relation "*Teaching Event has Topic*".
- `expectsLO` describes the relation "*Teaching Event expects some Learning Outcomes for students*".
- `hasTeacher` describes the relation "*Teaching Events has Professor as Instructor*" and it's inverse property of `teachesIn`.
- `followsTEMethod`, describes the relation "*Event in classroom follows Teaching Method*".

In class **`EvaluationEvent`** two extra object properties have been attached:

- `hasGrader`, describes the relation "*Evaluation Event has Professor as Grader*".
- `followsTEMethod`, describes the relation "*Event in classroom follows Evaluation Method*".

In class **`TeachingEvaluationMethod`** three data properties have been attached, namely `methodName`, `description` and `mediaUsed`.

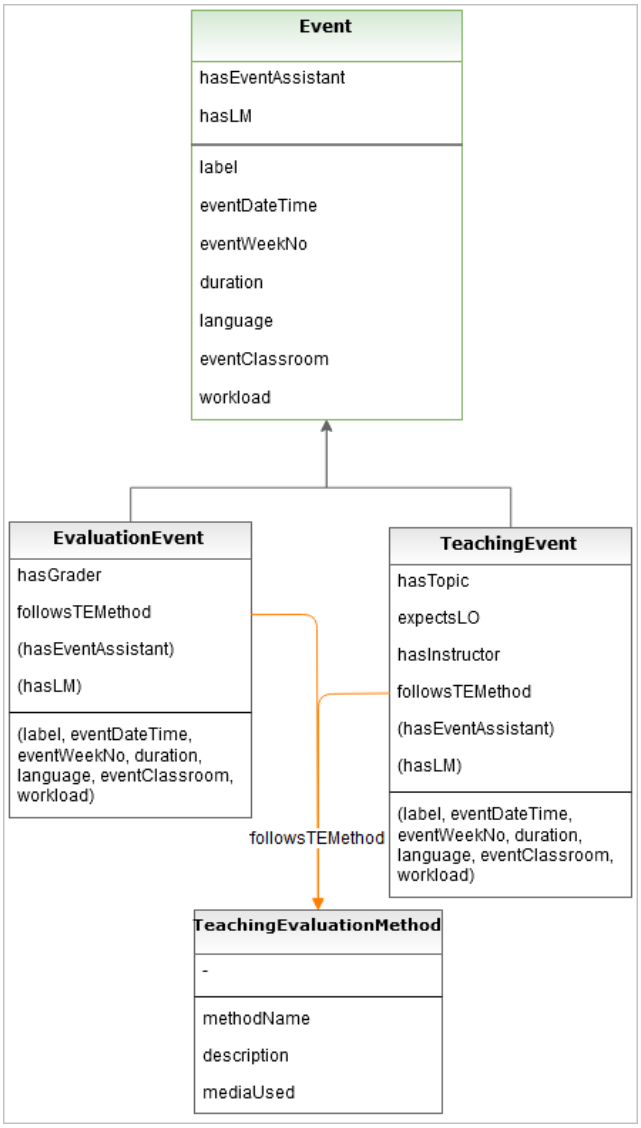


Figure 49: Object and Data properties of classes Event, TeachingEvent, EvaluationEvent and TeachingEvaluationMethod

Properties of classes LearningOutcome, Topic, Resource

In `Topic` class three data properties have been attached, namely `label`, `type`, `duration` and two in class `LearningOutcome`, namely `label` and `levelofMastery`. Data property `levelofMastery` expresses the *level of mastery* students are expected to achieve with respect to a topic.

Two object properties have been attached to each class, as shown in *Figure 50*.

In **`Topic`** class have been attached the following:

- `listedInKU` describes the relation "*Topic is listed by KnowledgeUnit*" and it's inverse property of `lists`.

- `coveredInCourse` describes the relation "*Topic is covered in Course*" and it's inverse property of `coversTopic`.

In **`LearningOutcome`** class have been attached the following:

- `listedInKU` describes the relation "*LearningOutcome is listed by KnowledgeUnit*" and it's inverse property of `lists`.
- `acquiredInCourse` describes the relation "*LearningOutcome is expected to be acquired in Courses*" and it's inverse property of `aimsToLO`.

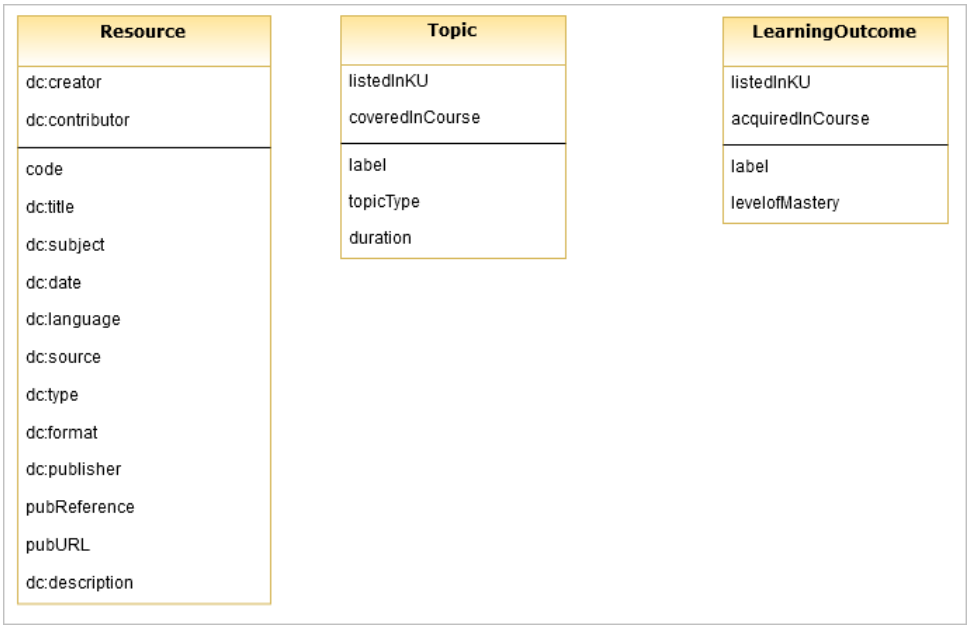


Figure 50: Object and Data properties of classes Topic, LearningOutcome and Resource

Properties of Resource class

The class `Resource` models the teaching and learning resources that are being used during a course. Books, academic articles, notes as well as webpages, videos, images, audios and other stuff with educational interest for a course can be used as teaching/learning material in a course.

The resource could be an official Publication that is a work of a person which its content is available to the general public or an unpublished work. The content may vary; usually it refers to text, images, audio or visual content, software programs, etc. The text includes papers and articles in newspapers, magazines, books, journals, *etc*.

The properties of this class have been aligned with *DC:elements*[39] for describing metadata of resources in order to enable better discovery and classification by search engines and applications. The object property `dc:creator` describes the relation "*An entity primarily responsible for making the resource*" while the property `dc:contributor` describes the relation "*An entity responsible for making contributions to the resource* ". As to the data properties, several properties have been attached for describing important information about a resource: `code`, `dc:title`, `dc:subject`, `dc:date`, `dc:language`, `dc:source`, `dc:type`, `dc:format`, `dc:publisher`, `resourceCitation`, `URL`, `dc:description`, as illustrated before in *Figure 50*.

Rules

Some rules have also been added to the ontology, so that it can be dynamically enriched with new relationships. These relationships, which are created automatically by reasoner, are used to answer queries of competency questions. *Figure 51* illustrates that rules from *Rules* tab in Protégé:

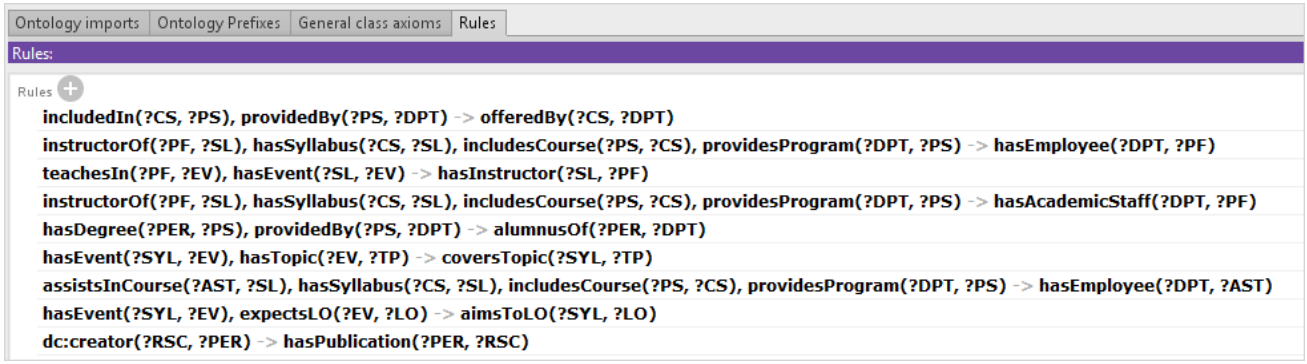


Figure 51: Rules tab in Protégé

The following list are the rules that have been added:

1. `includedIn(?CS, ?PS), providedBy(?PS, ?DPT) -> offeredBy(?CS, ?DPT)`
   A Course (CS) that is included in a ProgramofStudy (PS) which is provided by a Department (DPT) IS offered by that Department (DPT)

[39] Dublin Core Metadata Element Set, version 1.1: http://dublincore.org/documents/dces

2. `instructorOf(?PF, ?SL), hasSyllabus(?CS, ?SL), includesCourse(?PS, ?CS), providesProgram(?DPT, ?PS) -> hasEmployee(?DPT, ?PF)`
   A Professor (PF) that is instructor in a course (CS) which is included in a program of study (PS) which is provided by a Department (DPT) THEN the Department (DPT) has employee the Professor (PF)
3. `teachesIn(?PF, ?EV), hasEvent(?SL, ?EV) -> hasInstructor(?SL, ?PF)`
   A Professor (PF) that teaches in an Event (EV) for a course (CS) which is included in a program of study (PS) which is provided by a Department (DPT) THEN the Department (DPT) has Academic Staff that Professor (PF)
4. `instructorOf(?PF, ?SL), hasSyllabus(?CS, ?SL), includesCourse(?PS, ?CS), providesProgram(?DPT, ?PS) -> hasAcademicStaff(?DPT, ?PF)`
   A Professor (PF) that is instructor in a course (CS) which is included in a program of study (PS) which is provided by a Department (DPT) THEN that Professor (PF) is member of the Academic Staff of the Department (DPT)
5. `hasDegree(?PER, ?PS), providedBy(?PS, ?DPT) -> alumnusOf(?PER, ?DPT)`
   A Person (PER), who has Degree (PS) that provided by a Department (DPT) IS Alumnus of that Department (DPT)
6. `hasEvent(?SYL, ?EV), hasTopic(?EV, ?TP) -> coversTopic(?SYL, ?TP)`
   When a Syllabus (Course)(SYL) has an Event (EV) that covers a Topic (TP) THEN the Syllabus (Course)(SYL) covers that Topic (TP)
7. `assistsInCourse(?AST, ?SL), hasSyllabus(?CS, ?SL), includesCourse(?PS, ?CS), providesProgram(?DPT, ?PS) -> hasEmployee(?DPT, ?AST)`
   An Assistant (AST) that is instructor in a course (CS) which is included in a program of study (PS) which is provided by a Department (DPT) THEN the Department (DPT) has employee the Professor (PF)
8. `hasEvent(?SYL, ?EV), expectsLO(?EV, ?LO) -> aimsToLO(?SYL, ?LO)`
   When a Syllabus (Course)(SYL) has an Event (EV) that expects a LearningOutcome (LO) THEN the Syllabus (Course)(SYL) aims to that LearningOutcome (LO)
9. `dc:creator(?RSC, ?PER) -> hasPublication(?PER, ?RSC)`
   When a Resource (RSC) that has a Person (PER) as Creator, then this Person hasPublication that Resource.

Overview of the ontology

The final developed ontology comprises of 41 concepts in a taxonomy, 9 of which are the top-level concepts of the ontology, as shown in *Figure 52*. It also includes 54 objects properties for establishing relations between concepts, with 20 pairs of inverse object properties, and 76 data properties for describing concepts characteristics in detail.

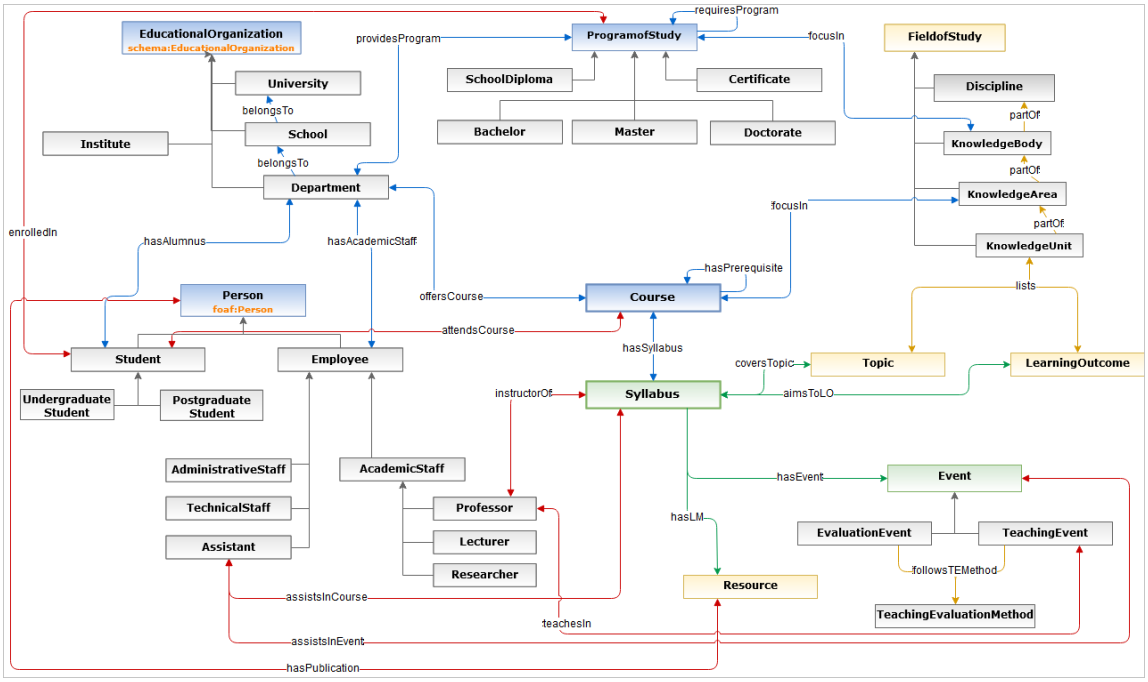


Figure 52: Classes, class hierarchy and their relations of CCSO

3.4.5. Attaching Facets to Properties

A property (slot in older versions of Protégé) can have several facets describing the characteristics of the property. In *data properties*, we attach facets that express the value type, the cardinality, allowed values and other specifications for the values that the property can take. In case of an *object property*, the values that a property can take are instances of other related classes which are defined in the field *Range* of the class' property. Furthermore, we can define which properties are inversed of another or disjoint with. In addition, OWL allows the meaning of object properties to be enriched with *properties characteristics*.

The characteristics that object properties may have are the following:

- Functional
- Inverse functional
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

More information about the above characteristics and their meaning can be found in [97].

An example is illustrated in *Figure 53*, where in class `Syllabus`, the value of the property `csName` is an `xsd:string` while the property `weeklyHours` can take values with `xsd:duration` value type. As to the object property `hasInstructor`, its values should be instances of class `Professor`, its inversed property is the `instructorOf` and it disjoints with properties `hasAttender` and `hasCourseAssistant`.

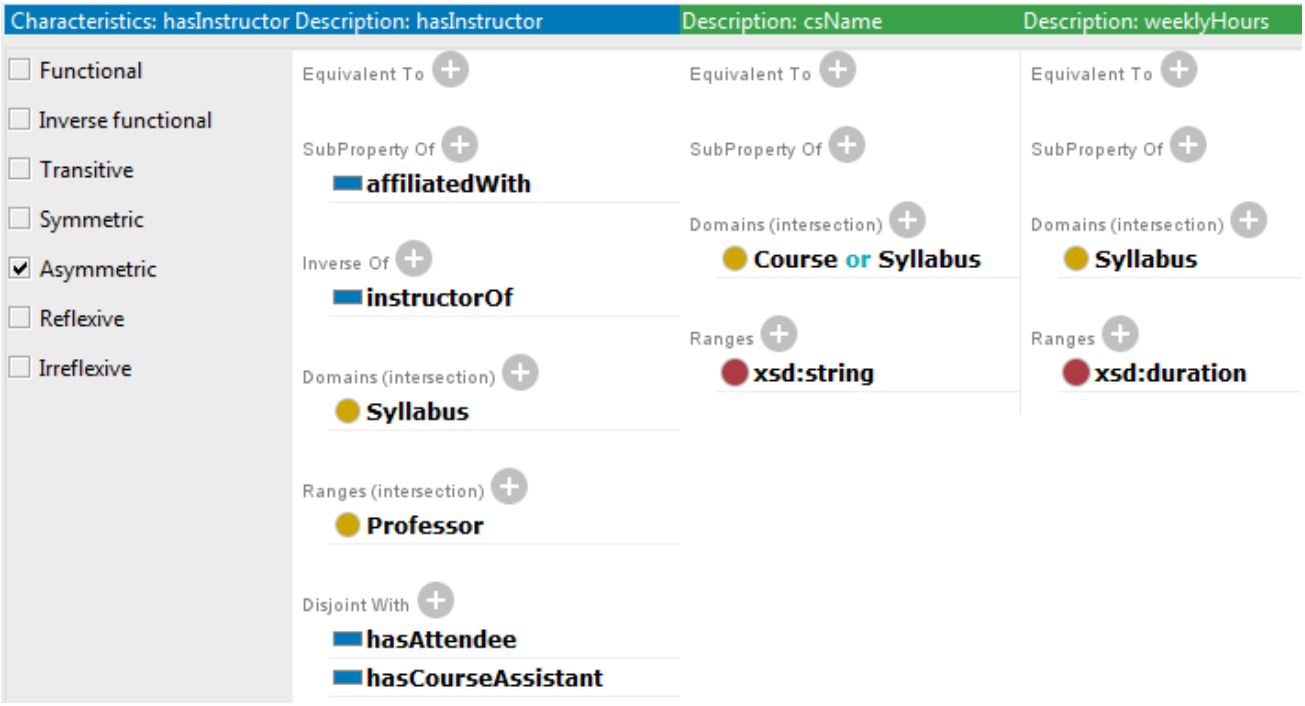


Figure 53: Some facets of Syllabus class in Protégé

In the next pages, all important information about the classes of the ontology is being presented, one-by one, including definitions of class, definition of its properties, alternate names and synonyms, facets of properties as well as the relationships with other classes.

Classes of FieldofStudy sub-tree

Class	**`FieldofStudy`**
Description (En)	The root-class of knowledge structure. It refers to the Academic Disciplines.

Description (Gr)	Η κλάση-ρίζα της οργανωτικής δομής με τις έννοιες που αφορούν τη γνώση και αφορά όλες τις Ακαδημαϊκές επιστήμες.
AltNames, Synonyms	Πεδίο Γνώσης
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)
has Sub-classes	`KnowledgeBody`
Disjoint	EducationalOrganization, Person

Object property	**Range**	**Relation it describes**
partOfFS	FieldofStudy	A field of study may be part of a broader field of study. It is used for describes relations between the sub-classes of the sub-tree. Transitive property.

Data property	**Declared as**	**Definition of Property**
fsName	String	Name of a field of study.
abbreviation	String	The shorten form of the Name of a field of study.

Class	**`Discipline`**
Description (En)	An ‘Academic Discipline' (here 'Discipline') is a branch of knowledge or a field of study.
Description (Gr)	Μία Επιστήμη ή 'Ακαδημαϊκή Επιστήμη' αντιπροσωπεύει ένα κλάδο γνώσης ή πεδίο σπουδών.
AltNames, Synonyms	Επιστήμη
is Sub-class of	`FieldofStudy`
has Sub-classes	-

Object property	**Range**	**Relation it describes**
(partOfFS)	FieldofStudy	Discipline is part of the FieldofStudy.

Data property	**Declared as**	**Definition of Property**
(fsName)	String	Name of a Discipline *(inherited property)*.
(abbreviation)	String	The shorten form of the Name of a Discipline *(inherited property)*.

Class	**`KnowledgeBody`**
Description (En)	Main knowledge body of a Discipline, a specification of the content to be covered in a curriculum.
Description (Gr)	Το κύριο σώμα γνώσης μιας Επιστήμης, μια προδιαγραφή του περιεχομένου που πρέπει να καλύπτεται σε ένα πρόγραμμα σπουδών.
AltNames, Synonyms	Body of Knowledge, Κύριο Σώμα Γνώσης
is Sub-class of	`FieldofStudy`

has Sub-classes	-

Object property	**Range**	**Relation it describes**
(partOfFS)	Discipline	A Knowledge Body is part of a Discipline.

Data property	**Declared as**	**Definition of Property**
(fsName)	String	Name of Body of Knowledge *(inherited property)*.
(abbreviation)	String	The shorten form of the Name of Body of Knowledge *(inherited property)*.

Class	**KnowledgeArea**
Description (En)	Subjects of a Body of Knowledge.
Description (Gr)	Θέματα ενός Σώματος Γνώσης (KnowledgeBody).
AltNames, Synonyms	Area of Knowledge, Περιοχή Γνώσης
is Sub-class of	FieldofStudy
has Sub-classes	-

Object property	**Range**	**Relation it describes**
areaOfCourse	Course	Knowledge Area of a Course. Inverse of: ***coversKA***
(partOfFS)	KnowledgeBody	A Knowledge Area is part of a Knowledge Body.

Data property	**Declared as**	**Definition of Property**
(fsName)	String	Name of *Area of Knowledge (inherited property)*.
(abbreviation)	String	The shorten form of the Name of *Area of Knowledge (inherited property)*.

Class	**KnowledgeUnit**
Description (En)	Part of a Knowledge Area.
Description (Gr)	Μέρος μιας Περιοχής Γνώσης (KnowledgeArea).
AltNames, Synonyms	Unit of Knowledge, Μέρος Γνώσης μιας Περιοχής Γνώσης
is Sub-class of	FieldofStudy
has Sub-classes	-

Object property	**Range**	**Relation it describes**
lists	Topic or LearningOutcome	Each KU lists a set of Topics that could be teached in a Teaching Event. Also, each KU lists a set of Learning Outcomes students are expected to achieve. Inverse property: ***listedInKU***
(partOfFS)	KnowledgeArea	A Knowledge Unit is part of a Knowledge Area.

Data property	Declared as	Definition of Property
(fsName)	String	Name of *Unit of Knowledge (inherited property).*
(abbreviation)	String	The shorten form of the Name of *Unit of Knowledge (inherited property).*
(fsTopic)	String	Topic of *Unit of Knowledge (inherited property).*

Classes of EducationalOrganization sub-tree

Class	**EducationalOrganization** shared type 'schema.org:EducationalOrganization'
Description (En)	An Organization designed to provide educational services and learning environments.
Description (Gr)	Ένας Οργανισμός που έχει σχεδιαστεί για την παροχή εκπαιδευτικών υπηρεσιών και μαθησιακών περιβαλλόντων.
AltNames, Synonyms	Εκπαιδευτικός Οργανισμός
is Sub-class of	schema.org:EducationalOrganization, Thing (root-class, by default the superclass of all classes)
has Sub-classes	University, Institute, School, Department

Object property	**Range**	**Relation it describes**
belongsTo	Educational Organization	It is used for describes relations between the sub-classes of the sub-tree. Transitive property.
providesProgram	ProgramofStudy	An Educational Organization provides a variety of Study Programs in many Disciplines and Knowledge Bodies. Inverse property: ***providedBy***
hasEmployee	Employee	EducationalOrganization has Employees. Inverse property: ***worksFor***

Data property	**Declared as**	**Definition of Property**
schema:identifier	String	The organization identifier that uniquely identifies the legal entity of the organization.
schema:legalName	String	The official name of the organization.
schema:logo	anyURI	An associated image as the logo of an Organization.
schema:address	String	The physical address of an Organization.
schema:location	String	The location an organization is located.
workPhone	String	The main phone number of an Organization or at the work of a Person.
schema:faxNumber	String	The fax number of an Organization or a Person.
schema:email	anyURI	The eMail address of an Organization or a Person.
schema:url	anyURI	The address of the webpage of an Organization or a Person.

In order to group some similar data properties, we use the data property phone as a super-property of homePhone, workPhone, mobilePhone and faxNumber and eContactInfo as super-property of email, workEmail and url, as shown in *Figure 54*.

Figure 54: The Super-properties phone and eContactInfo

Class	Institute	
Description (En)	An autonomous educational institution which provides learning services in areas of Non-Formal Education.	
Description (Gr)	Ένα αυτόνομο εκπαιδευτικό ίδρυμα που παρέχει υπηρεσίες μάθησης σε περιοχές της Μη-Τυπικής Εκπαίδευσης.	
AltNames, Synonyms	Ινστιτούτο	
is Sub-class of	EducationalOrganization	
has Sub-classes	–	
Object property	**Range**	**Relation it describes**
(belongsTo)	-	Object property is inherited from parent class (EducationalOrganization).
(providesProgram)	ProgramofStudy	Object property is inherited from parent class (EducationalOrganization).
(hasEmployee)	Employee	Object property is inherited from parent class (EducationalOrganization).
Data property	**Declared as**	**Definition of Property**
(identifier, legalName, logo, address, location, workPhone, faxNumber, email, url)	String, anyURI	Class inherits all its Data properties from the root-class of the sub-tree (EducationalOrganization).

Class	University	
Description (En)	An institution of Higher Education and research which awards academic degrees in a set of academic disciplines.	
Description (Gr)	Ένα ίδρυμα Τριτοβάθμιας Εκπαίδευσης και Έρευνας που απονέμει ακαδημαϊκούς τίτλους σε ένα σύνολο ακαδημαϊκών κλάδων.	
AltNames, Synonyms	Ανώτατο Εκπαιδευτικό Ίδρυμα (Α.Ε.Ι.), Πανεπιστήμιο, Ανώτατο Τεχνολογικό Εκπαιδευτικό Ίδρυμα (Α.Τ.Ε.Ι.)	
is Sub-class of	EducationalOrganization	
has Sub-classes	–	
Object property	**Range**	**Relation it describes**
(belongsTo)	-	The Object property is inherited from parent class (EducationalOrganization).
(providesProgram)	ProgramofStudy	The Object property is inherited from parent class (EducationalOrganization).
(hasEmployee)	Employee	University has Employee.

Data property	Declared as	Definition of Property
(identifier, legalName, logo, address, location, workPhone, faxNumber, email, url)	String, anyURI	The class inherits all its Data properties from the root-class of the sub-tree (EducationalOrganization).

Class	School
Description (En)	A division within a university comprising one subject area, or a number of related subject areas. The American usage is followed where such divisions are referred to as Schools or Colleges rather than using the term Faculty which is used in other countries in order to avoid misunderstandings because the same term (Faculty) refers to the Academic staff of a University in American usage.
Description (Gr)	Ένα τμήμα ενός πανεπιστημίου που περιλαμβάνει ένα θεματικό πεδίο ή έναν αριθμό σχετικών θεματικών πεδίων. Ακολουθήθηκε η αμερικάνικη προσέγγιση όπου τέτοια τμήματα αναφέρονται ως Κολλέγια ή Σχολεία αντί να χρησιμοποιηθεί ο όρος Faculty που χρησιμοποιείται σε άλλες χώρες για να αποφευχθούν παρεξηγήσεις, επειδή ο ίδιος όρος (Faculty) στην Αμερική αναφέρεται στο Ακαδημαϊκό Προσωπικό ενός Πανεπιστημίου.
AltNames, Synonyms	Σχολή, Κολλέγιο
is Sub-class of	–
has Sub-classes	–

Object property	Range	Relation it describes
(belongsTo)	University	School belongs to University.
(providesProgram)	ProgramofStudy	The Object property is inherited from the root-class of the sub-tree (EducationalOrganization).
(hasEmployee)	Employee	The Object property is inherited from the root-class of the sub-tree (EducationalOrganization).

Data property	Declared as	Definition of Property
(identifier, legalName, logo, address, location, workPhone, faxNumber, email, url)	String, anyURI	The class inherits all its Data properties from the parent class (University).

Class	`Department`	
Description (En)	A division of a College (or School) devoted to a particular academic degree.	
Description (Gr)	Ένα τμήμα ενός Κολλεγίου (ή Σχολής) που προσφέρει ένα συγκεκριμένο ακαδημαϊκό πτυχίο.	
AltNames, Synonyms	Τμήμα	
is Sub-class of	–	
has Sub-classes	–	
Object property	**Range**	**Relation it describes**
(belongsTo)	School	Department belongs to School.
(providesProgram)	ProgramofStudy	Department provides Program of Study.
(hasEmployee)	Employee	The Object property is inherited from the root-class of the sub-tree (EducationalOrganization).
hasAademicStaff	AcademicStaff	A Department has staff that is member of the Academic staff like Professors, Researchers and Lecturers. Inverse property: ***memberOf***
offersCourse	Course	A Department offers some Courses available to Students. Inverse property: ***offeredBy***
hasAlumnus	Person	A Department has alumnus, a person who completed successfully a Program of Study. Inverse property: ***alumnusOf***
Data property	**Declared as**	**Definition of Property**
(identifier, legalName, logo, address, location, workPhone, faxNumber, email, url)	String, anyURI	The class inherits all its Data properties from the root-class of the sub-tree (EducationalOrganization).

Classes of ProgramofStudy sub-tree

Class	**`ProgramofStudy`**
Description (En)	The degrees awarded to students upon a successful completion of a course of study in an educational institution.
Description (Gr)	Τα πτυχία που απονέμονται στους φοιτητές μετά την επιτυχή ολοκλήρωση ενός κύκλου σπουδών σε ένα εκπαιδευτικό ίδρυμα.
AltNames, Synonyms	InstitutionalDegree, Program of Study, Programme of Study, Τίτλος Προγράμματος Σπουδών
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)
has Sub-classes	`SchoolDiploma`, `Certificate`, `Bachelor`, `Master`, `Doctorate`

Object property	**Range**	**Relation it describes**
providedBy	Educational Organization	A Study Program is provided by an Educational Organization. Inverse property: ***providesProgram***
requiresProgram	ProgramofStudy	Study Program requires a Study Program. The property characterized as Asymmetric. *For example, a Master program requires the student has a Bachelor degree.*
includesCourse	Course	A Study Program can include many Courses. Inverse property: ***includedIn***
focusIn	KnowledgeBody	A Study Program is focused in a Body of Knowledge, at least.
hasRegistered	Student	A Study Program can have a lot of Students registered. Inverse property: ***enrolledIn***

Data property	**Declared as**	**Definition of Property**
code	Literal	The program identifier that uniquely identifies a Study Program in the national curriculum.
psName	String	The official name of a Study Program. The name could be different than official degree title.
degreeTitle	String	The official title of the Degree that awarded to the students upon a successful completion of Study Program in an educational institution.

degreeType	String	The various types of studies and degrees. A specific list of allowed values[40] in this property can also be used: · Bachelor of Science · Bachelor of Arts · Bachelor of Business Administration · Master of Science · Master of Arts · Master of Business Administration · Doctor of Philosophy
duration	duration	The type represents the duration of time of a ProgramofStudy. It is expressed as a number of years, months, days, hours, minutes, and seconds (e.g. P4Y=4 years, P6M=6 months).
degreeURL	anyURI	The URL of the official webpage with information about the Degree.
creditsECTS	decimal	The credits that corresponds to a Study Program, according to European Credit Transfer and Accumulation System (ECTS).

The rest classes of the sub-tree inherit all their *Object* and *Data properties* from the root-class (`ProgramofStudy`).

Class	**`SchoolDiploma`**
Description (En)	The graduation degrees provided by Primary and Secondary schools (Primary School, Secondary School and High School).
Description (Gr)	Τα πτυχία αποφοίτησης που δίδονται από σχολεία της Πρωτοβάθμιας και Δευτεροβάθμιας εκπαίδευσης (Δημοτικό, Γυμνάσιο και Λύκειο).
AltNames, Synonyms	Πτυχίο Αποφοίτησης
is Sub-class of	`ProgramofStudy`
has Sub-classes	–
Disjoint with	`Bachelor`, `Master`, `Doctorate`, `Certificate`

Object property	**Range**	**Relation it describes**
(providedBy)	EducationalOrganization	The class inherits its Object properties from the parent class (ProgramofStudy).
(requiresProgram)	-	
(includesCourse)	Course	
(focusIn)	KnowledgeBody	
(hasRegistered)	Student	

[40] Classification of Academic Degrees (Wikipedia): en.wikipedia.org/wiki/Classification_of_Academic_Degrees

Data property	Declared as	Definition of Property
(code, psName, degreeTitle, degreeType, duration, degreeURL, creditsECTS)	String, short, duration, decimal	The class inherits its Data properties from the parent class (ProgramofStudy).

Class	`Certificate`
Description (En)	The diplomas that cannot be classified to the other sub-classes such as Vocation diploma, Teaching diploma, Languages diploma, etc.
Description (Gr)	Τα διπλώματα που δεν μπορούν να ταξινομηθούν στις άλλες υποκατηγορίες, όπως το δίπλωμα Επαγγελματικής Κατάρτισης, το δίπλωμα Διδακτικής Επάρκειας, το δίπλωμα Ξένων Γλωσσών, κ.α.
AltNames, Synonyms	Πιστοποιητικό Σπουδών, Πιστοποιητικό Γνώσεων, Πιστοποιητικό Ικανοτήτων
is Sub-class of	`ProgramofStudy`
has Sub-classes	–
Disjoint with	`Bachelor`, `Master`, `Doctorate`, `SchoolDiploma`

Object property	Range	Relation it describes
(providedBy)	EducationalOrganization	The class inherits its Object properties from the parent class (ProgramofStudy).
(requiresProgram)	(SchoolDiploma)	A Program of Study in an Institution requires the student has a SchoolDiploma.
(includesCourse)	Course	
(focusIn)	KnowledgeBody	
(hasRegistered)	Student	

Data property	Declared as	Definition of Property
(code, psName, degreeTitle, degreeType, duration, degreeURL, creditsECTS)	String, short, duration, decimal	The class inherits its Data properties from the parent class (ProgramofStudy).

Class	`Bachelor`
Description (En)	The undergraduate academic degree awarded by universities and colleges, known as Bachelor's degree.
Description (Gr)	Το προπτυχιακό ακαδημαϊκό δίπλωμα που απονέμεται από πανεπιστήμια και κολέγια.
AltNames, Synonyms	Προπτυχιακό Ακαδημαϊκό Δίπλωμα, Πτυχίο

is Sub-class of	ProgramofStudy
has Sub-classes	–
Disjoint with	Master, Doctorate, Certificate, SchoolDiploma

Object property	Range	Relation it describes
(providedBy)	EducationalOrganization	The class inherits its Object properties from the parent class (ProgramofStudy).
(requiresProgram)	(SchoolDiploma)	A Program of Study for Bachelor degree requires the student has a SchoolDiploma.
(includesCourse)	Course	
(focusIn)	KnowledgeBody	
(hasRegistered)	Student	

Data property	Declared as	Definition of Property
(code, psName, degreeTitle, degreeType, duration, degreeURL, creditsECTS)	String, short, duration, decimal	The class inherits its Data properties from the parent class (ProgramofStudy).

Class	Master
Description (En)	The postgraduate academic degree, known as Master's degree, awarded by universities or colleges upon completion of a course of study demonstrating mastery of a specific field of study or area of professional practice.
Description (Gr)	Το μεταπτυχιακό ακαδημαϊκό δίπλωμα που απονέμεται από πανεπιστήμια ή κολέγια μετά την ολοκλήρωση ενός κύκλου σπουδών που αποδεικνύει την απόκτηση ειδικού πεδίου σπουδών ή τομέα επαγγελματικής πρακτικής.
AltNames, Synonyms	Μεταπτυχιακό Ακαδημαϊκό Δίπλωμα, Μεταπτυχιακό
is Sub-class of	ProgramofStudy
has Sub-classes	–
Disjoint with	Doctorate, Bachelor, Certificate, SchoolDiploma

Object property	Range	Relation it describes
(providedBy)	EducationalOrganization	The class inherits its Object properties from the parent class (ProgramofStudy).
(requiresProgram)	(Bachelor)	A Program of Study for Master degree requires the student has a Bachelor degree.
(includesCourse)	Course	
(focusIn)	KnowledgeBody	
(hasRegistered)	Student	

Data property	Declared as	Definition of Property
(code, psName, degreeTitle, degreeType, duration, degreeURL, creditsECTS)	String, short, duration, decimal	The class inherits its Data properties from the parent class (ProgramofStudy).

Class	Doctorate
Description (En)	The academic and research degree, also known as Doctor's degree, awarded by universities that certifies the person as an expert of art or sciences in a field and qualifies to teach at the university level.
Description (Gr)	Ο ακαδημαϊκός και ερευνητικός τίτλος, επίσης γνωστός ως Διδακτορικό δίπλωμα, απονέμεται από πανεπιστήμια που πιστοποιούν το άτομο ως ειδικός της τέχνης ή των επιστημών σε έναν τομέα και είναι ικανός να διδάσκει σε πανεπιστημιακό επίπεδο.
AltNames, Synonyms	Διδακτορικό Ακαδημαϊκό Δίπλωμα, Διδακτορικό
is Sub-class of	ProgramofStudy
has Sub-classes	–
Disjoint with	Master, Bachelor, Certificate, SchoolDiploma

Object property	Range	Relation it describes
(providedBy)	EducationalOrganization	The class inherits its Object properties from the parent class (ProgramofStudy).
(requiresProgram)	Master	A Program of Study for Doctorate degree may requires the student has a Master degree.
(includesCourse)	Course	
(focusIn)	KnowledgeBody	
(hasRegistered)	Student	

Data property	Declared as	Definition of Property
(code, psName, degreeTitle, degreeType, duration, degreeURL, creditsECTS)	String, short, duration, decimal	The class inherits its Data properties from the parent class (ProgramofStudy).

Classes of Person sub-tree

Class	**Person** shared type 'foaf:Person'
Description (En)	A person, a human being.
Description (Gr)	Ένας άνθρωπος.
AltNames, Synonyms	Άνθρωπος, Πρόσωπο
is Sub-class of	xmlns.com/foaf/0.1/Person, Thing (root-class, by default the superclass of all classes)
has Sub-classes	Employee, Student
Disjoint with	Discipline, FieldofStudy

Object property	**Range**	**Relation it describes**
hasDegree	ProgramofStudy	Person has some Degrees from some Departments (e.g. Bachelor Degree, Master Degree, etc) Inverse property: ***enrolledIn***
alumnusOf	Department	A Person is Alumnus of a Department. Inverse property: ***hasAlumnus***
hasPublication	Resource	A person creates and publishes a work.

Data property	**Declared as**	**Definition of Property**
personID	String	Person's identifier that uniquely identifies a person.
profilePic	anyURI	An associated profile picture of a person.
foaf: title	String	The title of a person, could an official title like Sir, Dr or Mr, Mrs, Ms, etc.
foaf: firstName	String	The first name of a person.
foaf: lastName	String	The last name of a person
foaf: familyName	String	The family name of a person
foaf: gender	String	The gender of this person, typically 'male' or 'female'.
homePhone	String	The phone number at the home of a person.
workPhone	String	The main phone number of an Organization or at the work of a Person.
mobilePhone	String	The mobile's phone number of a person.
schema: faxNumber	String	The fax number of an Organization or a Person.
schema: email	anyURI	The personal eMail address of a person.
workEmail	anyURI	The work eMail address of a person.
schema: url	anyURI	The address of the webpage of an Organization or a Person.

A person can have a home phone, a work phone, a fax number and a mobile phone as well as email, work email and homepage. In order to group some data properties that stores

similar information we use the relationship *rdfs:subPropertyOf*. We use the data property `phone` as a super-property of `homePhone`, `workPhone`, `mobilePhone` and `faxNumber` and the property `eContactInfo` as super-property of `email`, `workEmail` and `url`, as shown in *Figure 54* before.

Class	**`Employee`**	
Description (En)	Employee of an organization.	
Description (Gr)	Εργαζόμενος σε κάποιον οργανισμό.	
AltNames, Synonyms	Εργαζόμενος	
is Sub-class of	`Person`	
has Sub-classes	`AcademicStaff`, `AdministrativeStaff`, `TechnicalStaff`, `Assistant`	
Object property	**Range**	**Relation it describes**
worksFor	Educational Organization	An employee works for an organization. Inverse property: ***hasEmployee***
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**
staffSpecialization	String	The subject that a person is specialist and practices in.
staffPosition	String	The position where a person is being employed in the organization.
officeNo	String	The number and building of employee's office.
officeHours	String	Day and hours that the employee is at the office.
(personID, profilePic, title, firstName, lastName, familyName, gender, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits these Data properties from the parent class (Person).

Class	**`AdministrativeStaff`**
Description (En)	An employee who conducts of the administrating of an educational organization.
Description (Gr)	Ένας εργαζόμενος που ασκεί τη διοίκηση ενός εκπαιδευτικού οργανισμού.

AltNames, Synonyms	Administrative Staff, Διοικητικό Προσωπικό
is Sub-class of	`Employee`
has Sub-classes	–
Disjoint with	Student, TechicalStaff, Assistant

Object property	**Range**	**Relation it describes**
(worksFor)	EducationalOrganization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).

Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits these Data properties from the parent class (Person).

Class	**`TechnicalStaff`**
Description (En)	An employee who is specialized and related to provide practical, mechanical and applied services (e.g. electrician, plumber, etc).
Description (Gr)	Ο εργαζόμενος ο οποίος είναι ειδικευμένος στην παροχή πρακτικών, μηχανικών και εφαρμοσμένων υπηρεσιών (π.χ. ηλεκτρολόγος, υδραυλικός κ.λπ.).
AltNames, Synonyms	Technical Staff, Τεχνικό Προσωπικό, Τεχνικός
is Sub-class of	`Employee`
has Sub-classes	–
Disjoint with	AcademicStaff, AdministrativeStaff, Assistant

Object property	**Range**	**Relation it describes**
(worksFor)	EducationalOrganization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).

Data property	**Declared as**	**Definition of Property**

(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	The class inherits these Data properties from the parent class (Person).

Class	**Assistant**	
Description (En)	A person, usually an postgraduate student, who helps by assuming duties in lessons (TeachingAssistant), projects (ProjectAssistant), research (ResearchAssistant), *etc*.	
Description (Gr)	Κάποιο άτομο, συνήθως ένας μεταπτυχιακός φοιτητής, ο οποίος βοηθά με την ανάληψη καθηκόντων σε μαθήματα (βοηθός διδασκαλίας), σε έργα (βοηθός έργου), σε έρευνα (βοηθός έρευνας) κλπ.	
AltNames, Synonyms	Βοηθός	
is Sub-class of	Employee	
has Sub-classes	–	
Disjoint with	UndegraduateStudent, AcademicStaff, AdministrativeStaff, TechicalStaff	
Object property	**Range**	**Relation it describes**
assistsInCourse	Syllabus	An Assistant is (Teaching) Assistant in a Course. Inverse property: ***hasCourseAssistant*** Disjoint with: ***supervisorOf***, ***attendsCourse***, ***instructorOf***
assistsInEvent	Event	An Assistant may assists in a teaching event, an evaluation event or some other stuff and events. Inverse property: ***hasEventAssistant***
(worksFor)	EducationalOrganization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**

(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits these Data properties from the parent class (Person).

Class	**`AcademicStaff`**	
Description (En)	The academic staff of an institution, a school or a university such as professors of various ranks, lecturers and/or researchers. In North America the term Faculty is used instead.	
Description (Gr)	Το ακαδημαϊκό/εκπαιδευτικό προσωπικό ενός ιδρύματος, ενός σχολείου ή ενός πανεπιστημίου, όπως καθηγητές διαφόρων τάξεων, λέκτορες ή / και ερευνητές	
AltNames, Synonyms	Faculty, Ακαδημαϊκό προσωπικό, Εκπαιδευτικό προσωπικό	
is Sub-class of	`Employee`	
has Sub-classes	`Professor`, `Lecturer`, `Researcher`	
Disjoint with	TechicalStaff, Assistant, Student	
Object property	**Range**	**Relation it describes**
memberOf	Department	An academic person that is member of the faculty of a department. Inverse property: ***hasAcademicStaff***
(worksFor)	EducationalOrganization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits these Data properties from the parent class (Person).

Class	**Professor**
Description (En)	An instructor that is an expert of art or sciences and participates in a program of study of an educational institution.
Description (Gr)	Ο εκπαιδευτής που είναι ειδικός της τέχνης ή της επιστήμης και συμμετέχει σε πρόγραμμα σπουδών ενός εκπαιδευτικού ιδρύματος.
AltNames, Synonyms	Καθηγητής
is Sub-class of	AcademicStaff
has Sub-classes	–

Object property	**Range**	**Relation it describes**
supervisorOf	Syllabus	A Professor can supervise and manage all the activities within a Course such as the design, the implementation, the lessons, the outcomes, etc. Inverse property: ***hasSupervisor***
instructorOf	Syllabus	Professor is Instructor in some Courses Inverse property: ***hasInstructor***
teachesIn	TeachingEvent	A Professor teaches in some Teaching Events within a course. Inverse property: ***hasTeacher***
(memberOf)	Department	The Object property is inherited from parent class (AcademicStaff).
(worksFor)	EducationalOrganization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).

Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits its Data properties from the parent class (AcademicStaff).

Class	**Lecturer**

Description (En)	An academic person at an early career stage with open-ended or tenured position in a university who teaches and conducts research.
Description (Gr)	Ένα ακαδημαϊκό πρόσωπο το οποίο βρίσκεται στο αρχικό στάδιο της σταδιοδρομίας του με αορίστου ή μόνιμη θέση σε ένα πανεπιστήμιο που διδάσκει και διεξάγει έρευνα.
AltNames, Synonyms	Λέκτορας
is Sub-class of	`AcademicStaff`
has Sub-classes	–

Object property	**Range**	**Relation it describes**
(memberOf)	Department	The Object property is inherited from parent class (AcademicStaff).
(worksFor)	EducationalOrganization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).

Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits its Data properties from the parent class (AcademicStaff).

Class	**`Researcher`**
Description (En)	An academic person that conducts research, participates in research projects or leads research groups.
Description (Gr)	Μέλος του ακαδημαϊκού προσωπικού που διεξάγει έρευνα, συμμετέχει σε ερευνητικά έργα ή είναι υπεύθυνος ερευνητικών ομάδων.
AltNames, Synonyms	Ερευνητής
is Sub-class of	`AcademicStaff`
has Sub-classes	–

Object property	**Range**	**Relation it describes**

(memberOf)	Department	The Object property is inherited from parent class (AcademicStaff).
(worksFor)	Educational Organization	The Object property is inherited from the parent class (Employee).
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, staffSpecialization, gender, staffPosition, officeNo, officeHours, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits its Data properties from the parent class (AcademicStaff).

Class	`Student`	
Description (En)	A learner who attends an educational program.	
Description (Gr)	Ο εκπαιδευόμενος που παρακολουθεί κάποιο εκπαιδευτικό πρόγραμμα.	
AltNames, Synonyms	Φοιτητής, Σπουδαστής, Μαθητής, Εκπαιδευόμενος	
is Sub-class of	`Person`	
has Sub-classes	`UndergraduateStudent`, `PostgraduateStudent`	
Disjoint with	AdministrativeStaff, AcademicStaff	
Object property	**Range**	**Relation it describes**
enrolledIn	ProgramofStudy	A Student is enrolled (registered) in a Study Program. Inverse of: ***hasRegistered*** Disjoint with: ***hasDegree***
attendsCourse	Course	A Student attends some Courses of a study program. Inverse of: ***hasAttender*** Disjoint with: ***instructorOf***, ***hasCompleted***, ***assistsInCourse***
hasCompleted	Course	Student has successfully completed some Courses. Disjoint with: ***attendsCourse***
(hasDegree)	ProgramofStudy	Inherited property from root-class of the sub-tree (Person).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**

enrollmentDate	Date	The exact date that student has enrolled in a Study Program (e.g. 2016-09-01).
graduationDate	Date	The exact date that student has graduated a Study Program (e.g. 2017-03-15).
(personID, profilePic, title, firstName, lastName, familyName, gender, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url)	String, anyURI	The class inherits its Data properties from the parent class (Person).

Class	**`UndergraduateStudent`**	
Description (En)	A student who attend a university program up to the level to the Bachelor's degree.	
Description (Gr)	Ένας φοιτητής που παρακολουθεί κάποιο προπτυχιακό πανεπιστημιακό πρόγραμμα σπουδών.	
AltNames, Synonyms	Undergraduate Student, Προπτυχιακός Φοιτητής	
is Sub-class of	`Student`	
has Sub-classes	–	
Disjoint with	Assistant	
Object property	**Range**	**Relation it describes**
(enrolledIn)	ProgramofStudy	The Object property is inherited from the parent class (Student).
(attendsCourse)	Course	The Object property is inherited from the parent class (Student).
(hasCompleted)	Course	The Object property is inherited from the parent class (Student).
(hasDegree)	ProgramofStudy	The Object property is inherited from the parent class (Student).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, gender, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url, enrollmentDate, graduationDate)	String, anyURI, Date	The class inherits its Data properties from the parent class (Student).

Class	**PostgraduateStudent**	
Description (En)	A student who own a Bachelor's degree and studying for an academic or professional certificate.	
Description (Gr)	Ένας φοιτητής που κατέχει πτυχίο Bachelor και σπουδάζει για ακαδημαϊκό ή επαγγελματικό πιστοποιητικό σπουδών.	
AltNames, Synonyms	Postgraduate Student, Μεταπτυχιακός Φοιτητής	
is Sub-class of	Student	
has Sub-classes	–	
Object property	**Range**	**Relation it describes**
(enrolledIn)	ProgramofStudy, Course	The Object property is inherited from the parent class (Student).
(attendsCourse)	Course	The Object property is inherited from the parent class (Student).
(hasCompleted)	ProgramofStudy, Course	The Object property is inherited from the parent class (Student).
(hasDegree)	ProgramofStudy	The Object property is inherited from the parent class (Student).
(alumnusOf)	Department	Inherited property from root-class of the sub-tree (Person).
(hasPublication)	Resource	Inherited property from root-class of the sub-tree (Person).
Data property	**Declared as**	**Definition of Property**
(personID, profilePic, title, firstName, lastName, familyName, gender, homePhone, workPhone, mobilePhone, faxNumber, email, workEmail, url, enrollmentDate, graduationDate)	String, anyURI, Date	The class inherits its Data properties from the parent class (Student).

The Course class

Class	`Course`	
Description (En)	An institutional unit of teaching that lasts one academic term with a specific subject, usually is led by one or more instructors and has a fixed group of students.	
Description (Gr)	Μια μονάδα διδασκαλίας εκπαιδευτικού ιδρύματος που διαρκεί έναν ακαδημαϊκό εξάμηνο με ένα συγκεκριμένο θέμα, συνήθως διδάσκεται από έναν ή περισσότερους εκπαιδευτές και απευθύνεται σε μια σταθερή ομάδα μαθητών.	
AltNames, Synonyms	Μάθημα, Σεμινάριο	
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)	
has Sub-classes	–	
Object property	**Range**	**Relation it describes**
includedIn	ProgramofStudy	A Course can be included in various Study Programs. Inverse property: ***includesCourse***
coversKA	KnowledgeArea	Course covers part of a Knowledge Area. Inverse property: ***areaOfCourse***
offeredBy	Department	A Course is offered by Department. Inverse property: ***offersCourse***
hasAttender	Student	A Course has Student as Attender. Inverse property: ***attendsCourse*** Disjoint with: ***hasSupervisor***, ***hasIstructor***, ***hasCourseAssistant***
hasSyllabus	Syllabus	A Course has Syllabus.
hasPrerequisite	Course	A Course may require previous knowledge from other Courses.
Data property	**Declared as**	**Definition of Property**
code	String	The identifier that uniquely identifies a Course in the Institute.
csName	String	The official name of a Course.

The Syllabus ontology

Class	**`Syllabus`**
Description (En)	A document that addresses course information, expectations and responsibilities to students.
Description (Gr)	Ένα έγγραφο που επικοινωνεί προς τους φοιτητές τις πληροφορίες του μαθήματος, τις προσδοκίες καθώς και τις υποχρεώσεις τους.
AltNames, Synonyms	Course Syllabus, Περίγραμμα Μαθήματος
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)
has Sub-classes	–

Object property	**Range**	**Relation it describes**
coversTopic	Topic	A Course may Cover several Topics. The coverage of a Topic is depending from how many hours are being taught; A national or international curriculum guide of the Body of Knowledge can provide information about the coverage, the learning outcomes and the minimum hours. Inverse property: ***coveredInCourse***
aimsToLO	LearningOutcome	Course aims to several Learning Outcomes in connection with the covered Topics. Inverse property: ***acquiredInCourse***
syllabusOf	Course	It is Syllabus of a Course. Inverse property: ***hasSyllabus***
hasSupervisor	Professor	A Course can have a Professor as Supervisor who manages all the activities within a Course such as the design, the implementation, the lessons, the outcomes, etc. Inverse property: ***supervisorOf***
hasInstructor	Professor	A Course has one or more a Professors as Instructors. Inverse property: ***instructorOf*** Disjoints with: ***hasAttender***, ***hasCourseAssistant***
hasCourseAssistant	Assistant	A Course has one or more Assistants. Inverse property: ***assistsInCourse*** Disjoints with: ***hasSupervisor***, ***hasInstructor***, ***hasAttender***
hasEvent	TeachingEvent, EvaluationEvent	A Course is composed of several Events; most of them are Teaching Events and some are Evaluation Events.
hasLM	Resource	A Course may use several Resources as Learning Material.
followsTEMethod	Teaching EvaluationMethod	Course follows some Teaching and Evaluation Methods during lessons.

Data property	**Declared as**	**Definition of Property**
code	String	The identifier that uniquely identifies a Course in the Institute.
csName	String	The official name of a Syllabus.

courseType	String	The type of the Course. There are two types of course available: · *Core*: Courses that cover core topics and are deemed essential for an academic degree. · *Compulsory*: A term with the same meaning as "*Core*" which is used in syllabuses. It's also expresses the Courses that cover core topics and are deemed essential for an academic degree. · *Elective*: a Course that the student can select from other alternative courses.
semesterinProgram	Short	The number of semester within the ProgramofStudy where the Course is offered and starts (e.g. 3).
academicYear	String	The academic year where the Course is offered and starts (e.g. 2017-18 or 2017-2018).
language	language	The Language that used during the lessons, at the assignments and at the evaluation tests of a Course.(e.g. 'en' for English, 'gr' for Greek).
offeredtoErasmus	Boolean	A Course can be offered to Erasmus students or not.
startDate	Date	The date that the course lessons start (e.g. 2017-10-01).
endDate	Date	The date that the course lessons end (e.g. 2018-01-29).
duration	duration	The type represents the duration of time of a Course. It is expressed as a number of years, months, days, hours, minutes, and seconds (e.g. P4M=4 months).
weeklyHours	duration	The weekly teaching hours of the course (e.g. PT3H=3 hours, PT60M=60 minutes)
weeklyClass Schedule	String	The weekly class schedule presents the time, day and dates that the events of the Course are scheduled.
weeklyWorkload	duration	The workload in hours that required for the learning activities and assignments of a Course in weekly basis. The hours of the lessons are not included (e.g. PT3H50M=3 hours and 50 minutes, PT60M=60 minutes).
creditsECTS	decimal	The ECSTS credits that corresponds to a Course and added to the total credits of the student's study program after the successfully completion of a Course.
courseURL	anyURI	The webpage of a Course where someone can find useful information about the Course, the Syllabus, the Assignments, etc.
eClassURL	anyURI	The URL of the Course's eClass platform.
versionNo	Decimal	The version number of a Syllabus.
evaluationInfo	String	Information about Evaluation and Performance of the Course.
description	String	Description of the Course or Event.

Class	**Event**	
Description (En)	All events that happen in a class within a course.	
Description (Gr)	Όλα τα γεγονότα που συμβαίνουν σε μια τάξη κατά τη διάρκεια του προγράμματος μαθημάτων.	
AltNames, Synonyms	Συμβάν	
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)	
has Sub-classes	`TeachingEvent`, `EvaluationEvent`	
Object property	**Range**	**Relation it describes**
hasEventAssistant	Assistant	An Event may have one or more Assistants.
hasLM	LearningMaterial	An Event may use several Resources as Learning Material.
Data property	**Declared as**	**Definition of Property**
label	String	The Title of an Event.
eventDateTime	DateTime	The date and time the Event occurs. If the event lasts more than one time then the value is the date the event is starting (e.g. 2018-03-14T12:30:00 means 12:30 pm at 14 March, 2018).
eventWeekNo	short	The number of the week in the semester or the Course that the Event occurs.
duration	duration	The type represents the duration of time of an Event. It is expressed as a number of years, months, days, hours, minutes, and seconds (e.g. PT2H45M=2 hours and 45 minutes, PT50M=50 minutes).
language	language	The speaking and/or writing language that is used in an Event (e.g. 'EN' for English, 'GR' for Greek).
eventClassroom	String	The classroom, laboratory or any other location that the Event takes place.
eventWorkload	duration	The workload in hours that required for the learning activities of an Event; the duration of the Event is not included (e.g. PT4H30M=4 hours and 30 minutes, PT90M=90 minutes).

Class	**TeachingEvent**	
Description (En)	Events that happen in a class which are related to teaching.	
Description (Gr)	Γεγονότα που συμβαίνουν σε μια τάξη και σχετίζονται με τη διδασκαλία.	
AltNames, Synonyms	Συμβάν Διδασκαλίας, Διδασκαλία, Teaching Event, Teaching Lesson	
is Sub-class of	`Event`	
has Sub-classes	–	
Disjoint with	EvaluationEvent	
Object property	**Range**	**Relation it describes**

hasTopic	Topic	A Teaching Event has one or more Topics.
expectsLO	LearningOutcome	A Teaching Event expects (aims to) one or more Learning Outcomes for students.
hasTeacher	Professor	A Teaching Event has at least one Professor as Instructor (Teacher). Inverse property: ***teachesIn***
followsTEMethod	TeachingEvaluation Method	An Event in the classroom follows one or more Teaching Methods or either Evaluation Methods.
(hasEventAssistant)	Assistant	Inherited object property of the parent class (*Event*).
(hasLM)	LearningMaterial	Inherited object property of the parent class (*Event*).
Data property	**Declared as**	**Definition of Property**
(label, eventDateTime, eventWeekNo, duration, language, eventClassroom, eventWorkload)	String, language, DateTime, duration	The class inherits its Data properties from the parent class (*Event*).

Class	**`EvaluationEvent`**
Description (En)	Events included in evaluation procedure.
Description (Gr)	Γεγονότα που συμπεριλαμβάνονται στη διαδικασία αξιολόγησης.
AltNames, Synonyms	Συμβάν Εξετάσεων, Εξέταση, Evaluation, Examination
is Sub-class of	`Event`
has Sub-classes	–
Disjoint with	TeachingEvent

Object property	**Range**	**Relation it describes**
hasGrader	Professor	An Evaluation Event can have at least one Professor as Grader.
followsTEMethod	TeachingEvaluation Method	An Event in classroom follows one or more Teaching Methods or either Evaluation Methods.
(hasEventAssistant)	Assistant	Inherited object property of the parent class (*Event*).
(hasLM)	LearningMaterial	Inherited object property of the parent class (*Event*).
Data property	**Declared as**	**Definition of Property**
(label, eventDateTime, eventWeekNo, duration, language, eventClassroom, eventWorkload)	String, language, DateTime, duration	The class inherits its Data properties from the parent class (*Event*).

Class	`TeachingEvaluationMethod`
Description (En)	The manner and different kinds of teaching methods and learning activities within a course.
Description (Gr)	Ο τρόπος και τα είδη μεθόδων διδασκαλίας και μαθησιακών δραστηριοτήτων που χρησιμοποιούνται κατά τη διάρκεια των μαθημάτων.
AltNames, Synonyms	Μέθοδος Διδασκαλίας
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)
has Sub-classes	–

Object property	Range	Relation it describes
-		

Data property	Declared as	Definition of Property
methodName	String	The name of Method that is followed in a Teaching/Evaluation Event. There is a specific list of allowed values[41, 42] for this property: · *Lecture*: the instructor oral presents the subject or teaches people with little interaction. · *Demonstration*: "involves showing by reason or proof, explaining or making clear by use of examples or experiments". · *Discussion*: the subject is being examined and discussed in classroom where each student can express his opinion. · *Seminar*: students prepare by themselves and present their work for discussion and critique. · *Study*: the instructor assigns readings for each session which are then discussed in a meeting or a conference. · *Homework*: students are writing short or long papers or essays about a topic, solving problem sets, etc. · *Laboratory*: most of the work is taken place in a laboratory. · *Project*: students are organized in groups, work on a topic and meet with the instructor weekly for discussion and guidance. · *OralExamination*: the examiner poses questions to the student or group of students who have to answer in order to demonstrate sufficient knowledge about the subject. · *WrittenExamination*: students participate in a written examination which intended to measure the student's knowledge, skills and aptitude.
description	String	Description of a TeachingEvaluationMethod
mediaUsed	String	The devices, tools and objects that are used to store and deliver teaching and learning content and data.

[41] Wikipedia: en.wikipedia.org/wiki/Teaching_method
[42] Wikipedia: en.wikipedia.org/wiki/Course_(education)

Class	`Topic`	
Description (En)	A subject of study, a small piece from Body of Knowledge that could be organized into course and covered during lessons.	
Description (Gr)	Ένα θέμα μελέτης, ένα μικρό κομμάτι από το Σώμα της Γνώσης που θα μπορούσε να οργανωθεί σε πορεία και να καλυφθεί κατά τη διάρκεια των μαθημάτων.	
AltNames, Synonyms	Subject, Θέμα	
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)	
has Sub-classes	–	
Object property	**Range**	**Relation it describes**
listedInKU	KnowledgeUnit	Topic is listed by KnowledgeUnit. Inverse property: ***lists***
coveredInCourse	Syllabus	A Topic may be covered in several Courses included in different ProgramofStudies. Inverse property: ***coversTopic***
Data property	**Declared as**	**Definition of Property**
label	String	The Name of an educational Topic.
topicType	String	The type of the topic defines the importance of the topic for inclusion in a study program. The types of a topic could be the values: *Core*, *Core-Tier-1*, *Core-Tier-2*, and *Elective*.
duration	duration	The topic hours represents the minimum amount of time that is required in order to reach the sufficient depth of coverage. It is expressed as a number of years, months, days, hours, minutes, and seconds (e.g. PT6H30M=6 hours and 30 minutes, PT4H=4 hours, PT60M=60 minutes).

Class	`LearningOutcome`
Description (En)	The learning outcomes, specific knowledge, skills and competences, which the students will acquire with the successful completion of the course.
Description (Gr)	Τα μαθησιακά αποτελέσματα, οι ειδικές γνώσεις, δεξιότητες και ικανότητες, τις οποίες οι σπουδαστές θα αποκτήσουν με την επιτυχή ολοκλήρωση του μαθήματος.
AltNames, Synonyms	Learning Objective, Μαθησιακό Αποτέλεσμα
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)
has Sub-classes	–

Object property	Range	Relation it describes
listedInKA	KnowledgeUnit	LearningOutcome is listed by KnowledgeUnit. Inverse property: ***lists***
acquiredInCourse	Syllabus	A Learning Outcome may expected to be acquired in various Courses. Inverse property: ***aimsToLO***
Data property	**Declared as**	**Definition of Property**
label	String	The Name of the Learning Outcome.
levelofMastery	String	Each *Learning Outcome* has an associated *Level of Mastery* students are expected to achieve with respect to the *Topics* specified. There are three values for *Level of Mastery* ([98]): · *Familiarity*: student understands what a concept is or what it means. · *Usage*: student is able to use and apply a concept. · *Assessment*: student is able to consider a concept from multiple viewpoints and select the appropriate approach for a problem.

Class	**Resource** aligned with `DC:elements`
Description (En)	The teaching and learning resources that are being used during a course. Books, academic articles, notes as well as webpages, videos, images, audios and other stuff with educational interest for a course can be included to the learning material for a course. The resource could be an official Publication. A work of a person which its content is available to the general public. The content may vary; usually it refers to text, images, audio or visual content, software programs, etc. The text includes papers and articles in newspapers, magazines, books, journals, etc.
Description (Gr)	Οι πηγές διδασκαλίας και εκμάθησης που χρησιμοποιούνται κατά τη διάρκεια ενός μαθήματος. Βιβλία, ακαδημαϊκά άρθρα, σημειώσεις καθώς και ιστοσελίδες, βίντεο, εικόνες, ηχητικά και άλλα στοιχεία με εκπαιδευτικό ενδιαφέρον μπορούν να συμπεριληφθούν στο εκπαιδευτικό υλικό για ένα μάθημα. Η πηγή ενδέχεται να αποτελεί επίσημη Δημοσίευση. Ένα έργο ενός προσώπου, το περιεχόμενο του οποίου είναι διαθέσιμο στο ευρύ κοινό. Το περιεχόμενο ποικίλει. συνήθως αναφέρεται σε κείμενο, εικόνες, ηχητικό ή οπτικό περιεχόμενο, προγράμματα λογισμικού, κλπ. Το κείμενο περιλαμβάνει άρθρα και άρθρα σε εφημερίδες, περιοδικά, βιβλία, περιοδικά κλπ.
AltNames, Synonyms	Πηγή
is Sub-class of	`Thing` (root-class, by default the superclass of all classes)
has Sub-classes	–

Object property	Range	Relation it describes
dc:creator	Person	An entity primarily responsible for making the resource. Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity. Here is used to indicate the Author of a work.
dc:contributor	Person	An entity responsible for making contributions to the resource. Examples of a Contributor include a person, an organization, or a service. Typically, the name of a Contributor should be used to indicate the entity.

Data property	Declared as	Definition of Property
dc:title	String	A name given to the resource. Typically, a Title will be a name by which the resource is formally known.
dc:subject	String	The topic of the resource. Typically, the subject will be represented using keywords, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary.
dc:date	Date	A point or period of time associated with an event in the lifecycle of the resource. Date may be used to express temporal information at any level of granularity. Recommended best practice is to use an encoding scheme, such as the W3CDTF profile of ISO 8601 [W3CDTF] (e.g. 2017-03-15).
dc:language	Language	A language of the resource. Recommended best practice is to use a controlled vocabulary such as RFC 4646 [RFC4646].
dc:source	String	A related resource from which the described resource is derived. The described resource may be derived from the related resource in whole or in part. Recommended best practice is to identify the related resource by means of a string conforming to a formal identification system. Here: The various types of media where the work is published. The most common types are: • Book, • Journal, • Workshop, • Conference, • Proceedings, • Thesis, • Magazine, • Newspaper, • Webpage, • UnofficialPublication, etc.

dc:type	String	The nature or genre of the resource. The various types of the work. The most common types of works are: • Article, • Chapter, • Paper, • Booklet, • Poster, • Thesis, • Software, • Specification, • Report, • UnofficialPublication, etc. A Thesis could be BachelorThesis, MasterThesis or DoctoralThesis.
dc:format	String	The file format, physical medium, or dimensions of the resource. Examples of dimensions include size and duration. Recommended best practice is to use a controlled vocabulary such as the list of Internet Media Types.
dc:publisher	String	An entity responsible for making the resource available. Examples of a Publisher include a person, an organization, or a service. Typically, the name of a Publisher should be used to indicate the entity.
resourceCitation	String	The official cited reference for the published learning material or unofficial in case it isn't published.
URL	anyURI	The address of the webpage of an Organization, a Person or a Publication
dc:description	String	An account of the resource. Description may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource.

3.4.6. CREATE INSTANCES

The last step in ontology building process is to create *instances* of classes. Instances are also known as *individuals* and we can also think them as examples of the classes. We can describe in detail relevant instances by defining their names, classes they belong to, instances from other classes which they are related to, attribute names and values.

We enriched our ontology with a competent number of instances. The role of instances in evaluation process is vital; the answers that ontology returns in the queries of competency questions are based on individuals. Along with the creation of instances, we test the sufficiency and expressiveness of the properties; whether the defined properties can establish the relations the concept participates (object properties) and to include the required information (data properties).

Creating instances in Protégé 5 includes three steps: (i) select the class the instance belongs to, (ii) create the instance and giving a name, (iii) add values to the properties the instance inherits from its class. In Protégé both terms, "instances" and "individuals" are used.

Below, the individuals (instances) that have been added to main classes are presented and analyzed. We emphasized in the area of Computer Science domain as an example case study, thus most of the instances are related to Computer Science. Yet, some instances related to the subject of Psychology have been created. The focus has given to those classes that participate in the queries based on competency questions in order the ontology to be able to give answers. Some figures also included that illustrate the most typical examples of relations between individuals (instances). It must be noted that, when we activate *HermiT* reasoner [24] in Protégé, a number of properties are automatically inferred from the inference engine of the reasoner; these properties are presented in yellow background.

Instances of classes in FieldofStudy sub-tree

Most of the instances that created here are related to the knowledge coverage of Computer Science. We used the object property `partOfFS` to establish relation between a lower class to a higher class of the sub-tree, for example an instance of class `KnowledgeUnit` is `partOfFS` an instance of class `KnowledgeArea`. *Figure 55* presents all the created instances of the classes of *FieldofStudy* sub-tree.

In class **Discipline** the following instances have been created:

- Anthropology
- Biology
- Chemistry
- Computer_Science
- Earth_Sciences
- Economics
- Human_Geography
- Mathematics
- Physics
- Political_Science
- Psychology
- Sociology
- Space_Sciences
- Statistics

Two instances have been added in class **KnowledgeBody**, namely American_Psychological_Association_Standards_2010 which is partofFS *Psychology* (instance of class Discipline) and Computer_Science_Curricula_2013 which is partOfFS *Computer_Science* (instance of class Discipline).

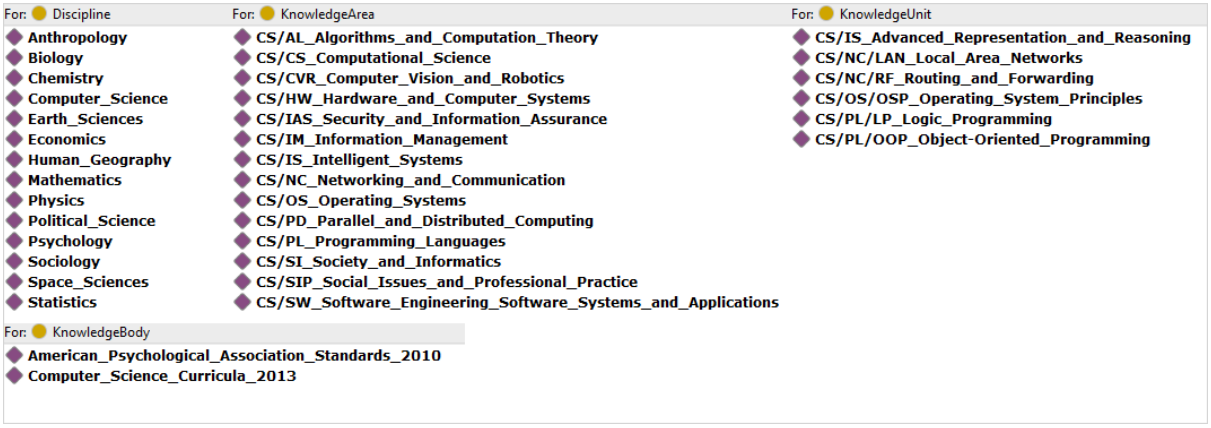


Figure 55: Instances of Classes in FieldofStudy sub-tree in Protégé

We referenced the *Computer Science Curricula 2013* [95] and added the following instances in class **KnowledgeArea**:

- CS/AL_Algorithms_and_Computation_Theory
- CS/CS_Computational_Science
- CS/CVR_Computer_Vision_and_Robotics
- CS/HW_Hardware_and_Computer_Systems
- CS/IAS_Security_and_Information_Assurance
- CS/IM_Information_Management
- CS/IS_Intelligent_Systems
- CS/NC_Networking_and_Communication
- CS/OS_Operating_Systems

- CS/PD_Parallel_and_Distributed_Computing
- CS/PL_Programming_Languages
- CS/SIP_Social_Issues_and_Professional_Practice
- CS/SI_Society_and_Informatics
- CS/SW_Software_Engineering_Software_Systems_and_Applications

In *Figure 56* for example, the properties of instance CS/PL_Programming_ Languages are presented. The properties in yellow background (*Figure 56*) are automatically inferred from the reasoner because areaOfCourse is inversed property of coversKA which has expressly stated.

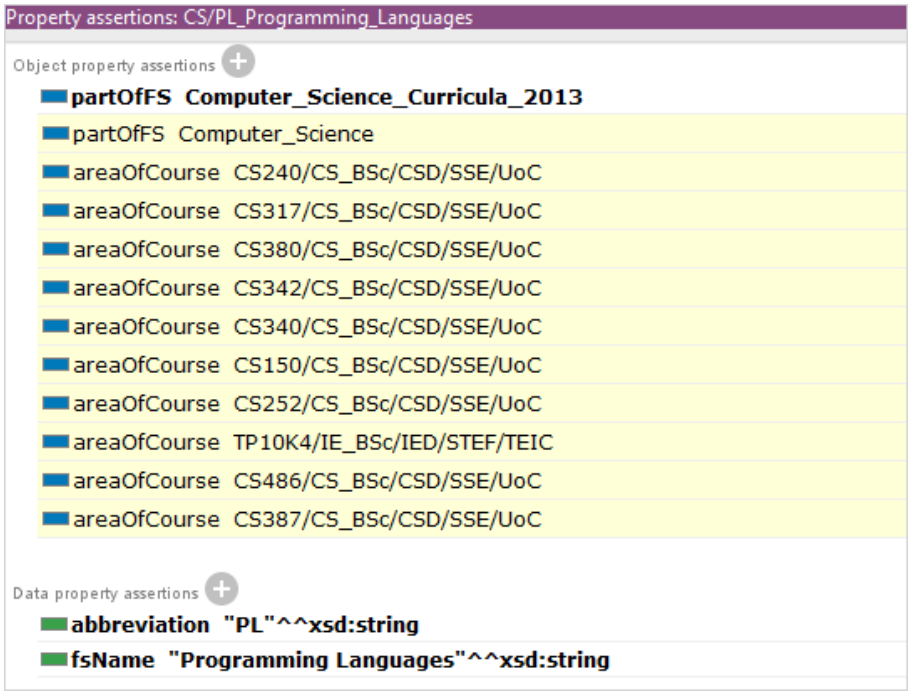


Figure 56: Properties of instance *CS/PL_Programming_Languages*

In class **KnowledgeUnit** the following set of instances have been added (*Figure 55*) and we used the object property lists to establish the relations "*KU lists a set of Topics*" and "*KU lists a set of Learning Outcomes*" with instances of Topic and LearningOutcome classes respectively.

- CS/IS_Advanced_Representation_and_Reasoning
- CS/NC/LAN_Local_Area_Networks
- CS/NC/RF_Routing_and_Forwarding
- CS/OS/OSP_Operating_System_Principles
- CS/PL/LP_Logic_Programming

- CS/PL/OOP_Object-Oriented_Programming

In *Figure 57* for example, properties of instance CS/NC/LAN_Local_Area_Networks are presented with *Topics* to be included in the red rectangles (point 1) and *LearningOutcomes* to be included in green rectangles (point 2).

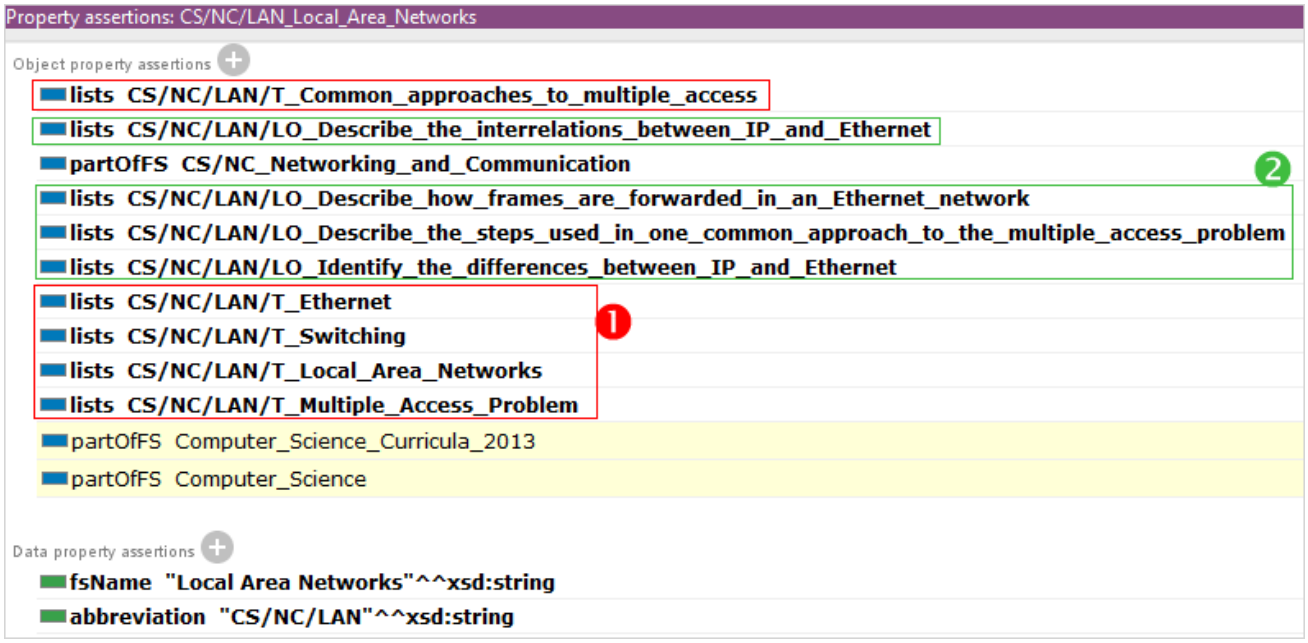


Figure 57: Properties of instance *CS/NC/LAN_Local_Area_Networks*

Figure 58 shows the graph with classes in *FieldofStudy* sub-tree and their instances in Protégé:

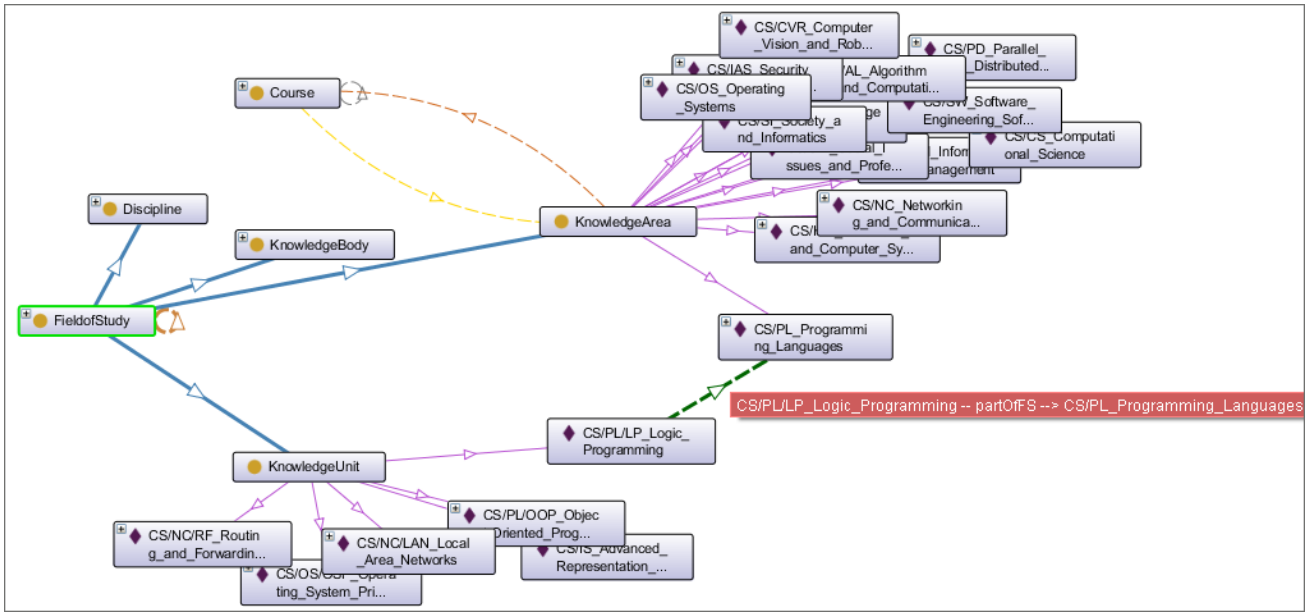


Figure 58: Graph with classes in *FieldofStudy* sub-tree and their instances in Protégé

Instances of classes in EducationalOrganization sub-tree

Considering the individuals of degrees that we will be created in *ProgramofStudy* sub-tree later, we created the appropriate individuals in these classes that will provide these degrees. *Figure 59* presents the instances of classes in *EducationalOrganization* sub-tree in Protégé. The small difference in names collated to the names in the lists below, is because of the annotation `label` that used for giving meaningful names.

Figure 59: Individuals of classes in EducationalOrganization sub-tree in Protégé

In **`University`** class two individuals have been created:

- `TEI_of_Crete`
- `University_of_Crete`

In **`School`** class three individuals have been created:

- `School_of_Engineering_of_TEI_of_Crete`
- `School_of_Sciences_and_Engineering_of_University_of_Crete`
- `School_of_Social_Sciences_of_University_of_Crete`

In **`Department`** class three individuals have been created, relevant with degrees from *ProgramofStudy* sub-tree:

- `Computers_Sciences_Department_of_University_of_Crete`
- `Department_of_Informatics_Engineering_of_TEI_of_Crete`
- `Psychology_Department_of_University_of_Crete`

In **Institute** class we added the individual `1o_IEK_Irakleiou`.

In *Figure 60* the annotations and property assertions of individual *Department_of_Informatics_ Engineering_of_TEI_of_Crete* are presented. Specifically, in the red points are presented: (1) the annotated labels, (2) the Programs that the department provides, (3) the School that the Department belongs to, (4) the Employees of the Department (to reduce the image dimensions only some of them are shown), (5) the University that the Department belongs to, (6) the Courses that offered by the Department, (7) the Academic staff of the Department, (8) the Persons that have an affiliation with the Department, and (9) the information that have been attached to the data properties of the Department. The properties in yellow background are semantically inferred by the reasoner.

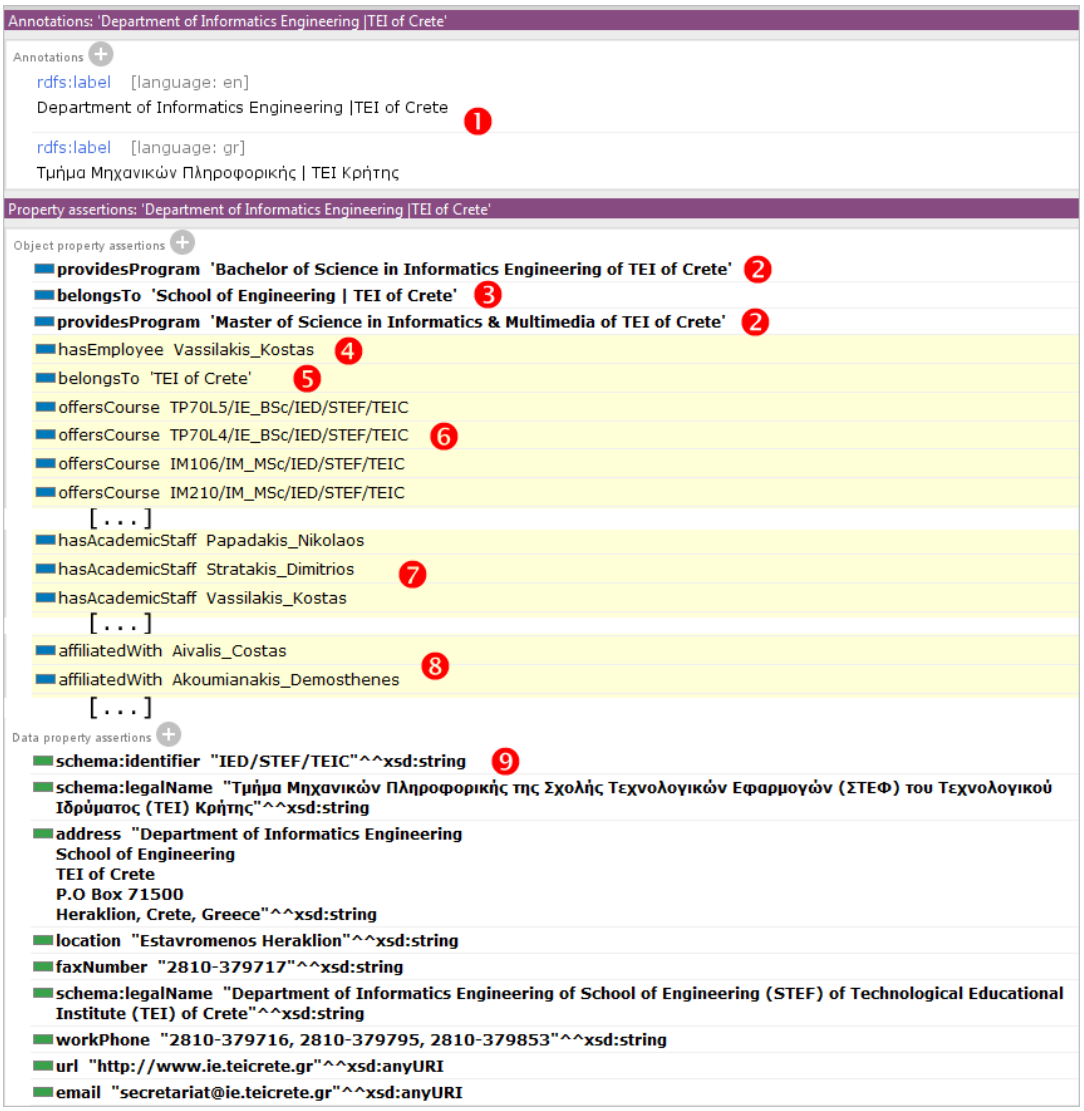


Figure 60: Annotations and Property assertions of individual *Department of Informatics Engineering of TEI of Crete*

Figure 61 shows the graph with classes in *EducationalOrganization* sub-tree and their instances in Protégé. The yellow lines with arrows depict the relation *belongsTo* between the individuals.

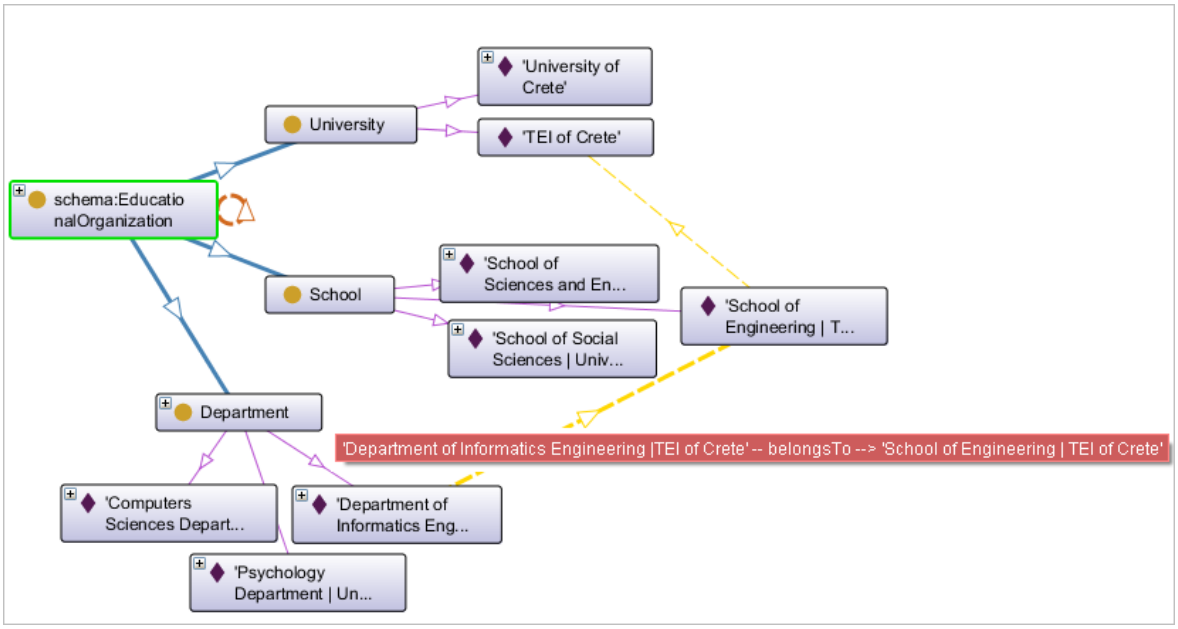


Figure 61: Graph with classes in *EducationalOrganization* sub-tree and their instances

Instances of classes in ProgramofStudy sub-tree

The instances that have been created in each class in this sub-tree are typical examples of each type/level of degree. Each instance can establish relations with instances of classes `EducationalOrganization` (in our examples with instances of `Department` class), `KnowledgeBody`, `Course` and `Student` with the object properties `providedBy`, `hasKB`, `includesCourse` and `hasRegistered` respectively. *Figure 62* presents the instances of classes in *ProgramofStudy* sub-tree in Protégé. It must be noted that, the small difference in names as shown in Protégé collated to the names in the lists below, is because of the annotation `label` that used for giving meaningful names.

Figure 62: Instances of classes in ProgramofStudy sub-tree in Protégé

In **`Bachelor`** class three individuals have been created, corresponding one Bachelor degree for each individual of `Department` class:

- `BSc_Computer_Science/CS_BSc/CSD/SSE/UoC`
  (label: *Bachelor of Science in Computer Science of University of Crete*)
- `BSc_Informatics_Engineering/IE_BSc/IED/STEF/TEIC`
  (label: *Bachelor of Science in Informatics Engineering of TEI of Crete*)
- `BSc_Psychology/PSY_BSc/SoC/UoC`
  (label: *Bachelor of Science in Psychology of University of Crete*)

In **`Master`** class three individuals have been created, corresponding one Master degree for each individual of `Department` class:

- `MSc_Computer_Science_and_Engineering/CSE_MSc/CSD/SSE/UoC`
  (label: *Master of Science in Computer Science and Engineering of University of Crete*)
- `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC`
  (label: *Master of Science in Informatics & Multimedia of TEI of Crete*)
- `MSc_School_Psychology/SP_MSc/PSY/SoC/UoC`
  (label: *Master of Science in School Psychology of University of Crete*)

In **`Doctorate`** class two individuals have been created, corresponding one Doctorate degree for each individual of `Department` class that belongs to *University_of_Crete*:

- `PhD_Clinical_Psychology/CP_PhD/Psy/SoC/UoC`
  (label: *PhD in Clinical Psychology of University of Crete*)
- `PhD_Computer_Science_and_Engineering/CSE_PhD/CSD/SSE/UoC`
  (label: *PhD in Computer Science and Engineering of University of Crete*)

In **`Certificate`** class the individual *DEK_Technician_for_Networks_and_Telecommunications_of_IEK* has been created and in class `SchoolDiploma` the individual *High_School_Diploma* (*Figure 62*).

Figure 63 below presents the Property assertions of individual '*Master of Science in Informatics & Multimedia of TEI of Crete*' (IRI: *MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC*). In red points are presented the following object property assertions: (1) Courses that this MSc program includes, (2) Department that provides the program, (3) Bachelor degrees that required in order a person to enroll in this program, (4) Students that are enrolled in the program. The last assertions with the relations `hasRegisterd` and `affiliatedWith` (yellow background) are semantically inferred by the reasoner. In point 5 of *Figure 63* sthe data property assertions of the program are presented.

Figure 64 presents the graph with classes in `ProgramofStudy` sub-tree and their instances. The dashed arrows show the relations that this IE_MSc program requires a *Bachelor degree* (point 1) and is provided by the *Department of Informatics Engineering* (point 2).

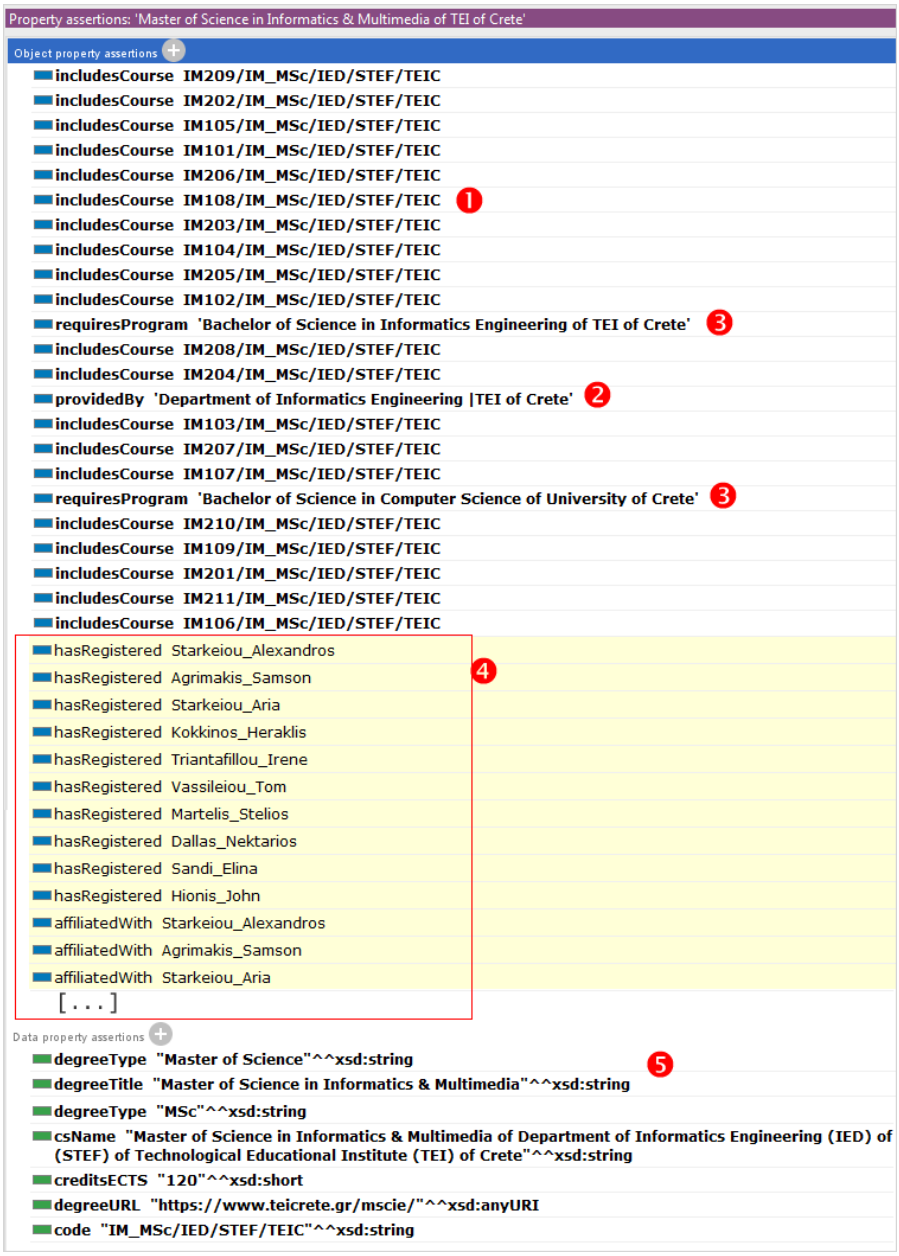


Figure 63: Property assertions of individual *MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC*

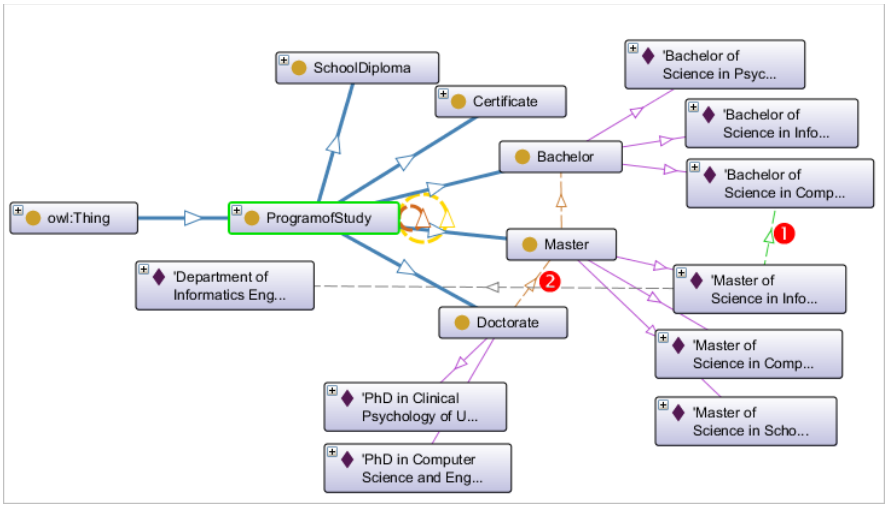


Figure 64: Graph with classes in ProgramofStudy sub-tree and their instances

Instances of classes in Person sub-tree

A competent number of instances have been created in the three classes of this sub-tree that have important role in our ontology, which are the classes `Professor`, `Assistant` and `Student`.

The instances of sub-classes are also instances of class `Person`, so they can establish relations with instances of classes `ProgramofStudy`, `Department`, and `Resource` with object properties `hasDegree`, `alumnusOf`, and `hasPublication` respectively. Each instance of class `Professor` can establish relations with instances of class `Syllabus` with object properties `suppervisorOf`, `instructorOf`, with class `Department` as `memberOf` and with instances of class `Event` with object properties `teachesIn`, `hasGrader`. Each instance of class `Assistant` can establish relations with instances of class `Syllabus` and `Event` with object properties `assistsInCourse`, and `assistsInEvent` respectively. As to the instances of class `Student`, they can establish relations with instances of class `Course` with object properties `attendsCourse`, `hasCompleted`, and with instances of class `ProgramofStudy` with property `enrolledIn`. *Figure 65* presents the instances of classes in `Person` sub-tree in Protégé. The instances of class `Professor` are real professors and some instances of `Assistant` class while the instances of class `Student` are not real persons.

Figure 65: Instances of the 3 most important classes in *Person* sub-tree in Protégé

The property assertions of professor `Papadakis Nikos` are presented in *Figure 66*: (1) Degrees that this person has, (2) Department that this person work for and he is member of its Academic Staff, (3) Courses that this Professor supervises, (4) affiliation relations between this person and other instances, (5) Courses that the proffesor teaches (as professor), (6) Publications that this person has authored, and (7) data properties of this person.

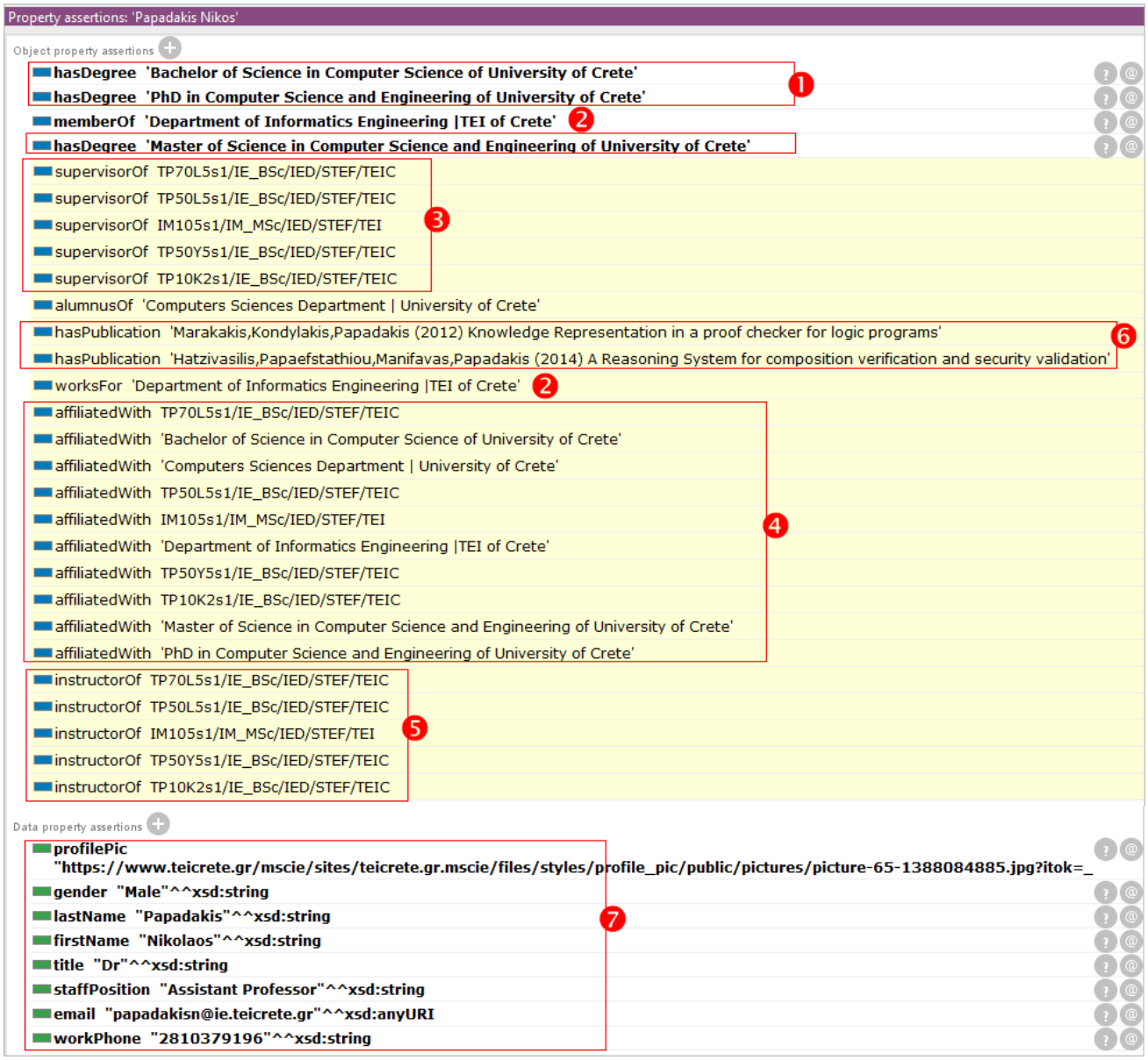


Figure 66 : Property assertions of professor *Papadakis Nikos*

Figure 67 depicts the graph with relations that Professor `Vassilakis Kostas` has with instances from other classes: (1) Department that the professor is member of its Academic staff, (2) Degrees that he has, (3) one of his Publications, (4) Courses that he supervises, and (5) Courses he teaches.

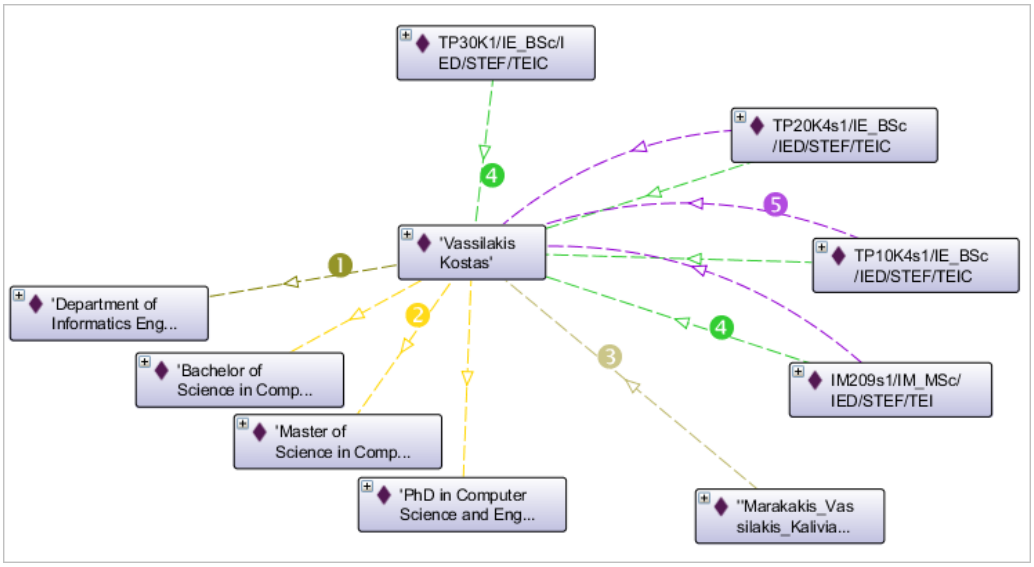


Figure 67: Graph with relations that Professor *Vassilakis Kostas* has with instances of other classes

All students belong to class `Student` (*Figure 65*) and are classified to its sub-classes with automatic inference from the reasoner based on some rules and restrictions (*points 1*) as shown in *Figure 68*, where 9 students are classified to `UndergraduateStudent` class (*point 2*) and 16 students are classified to `PostgraduateStudent` class (*point 3*).

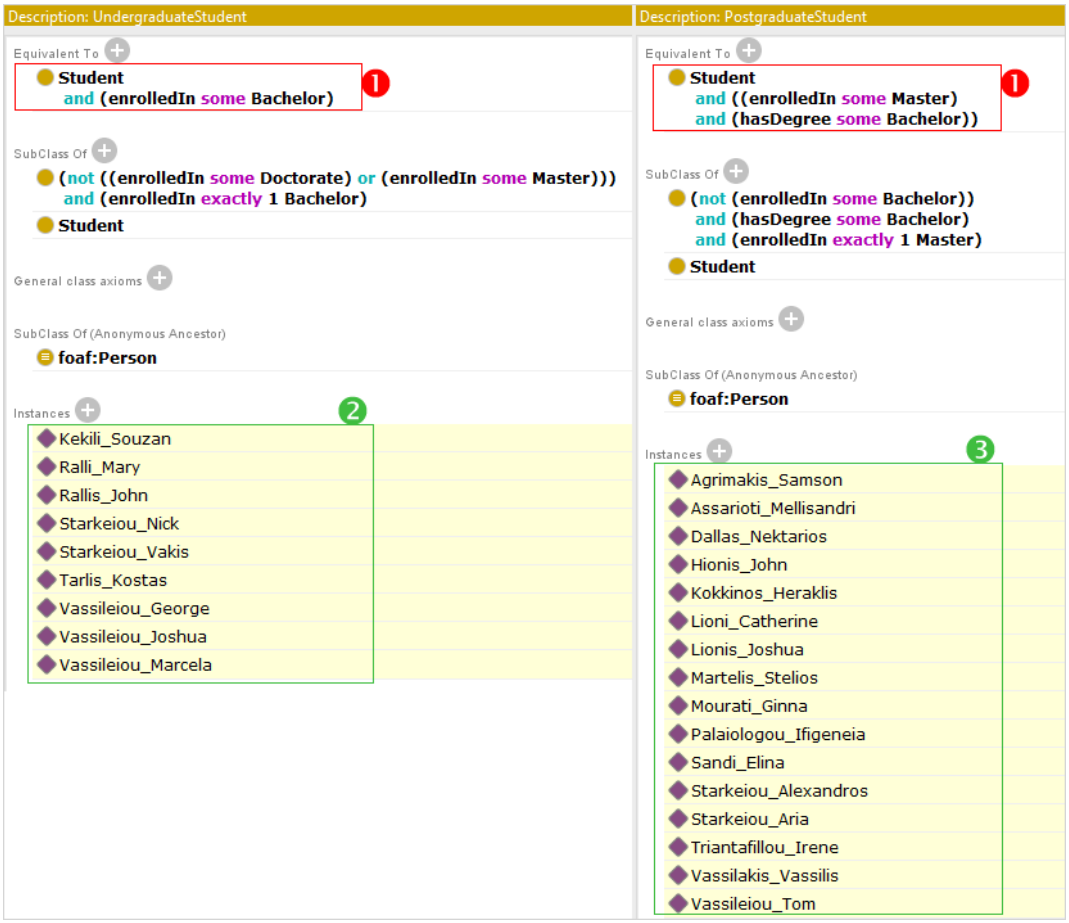


Figure 68: Classification of *Students* in sub-classes *UndergraduateStudent*, *PostgraduateStudent*

Figure 69 depicts the graph with relations that student `Hionis John` has with instances from other classes: (1) Degree that student has, (2) Program of study that student is enrolled in, (3) Courses that student currently attends, (4) Courses that student has successfully completed.

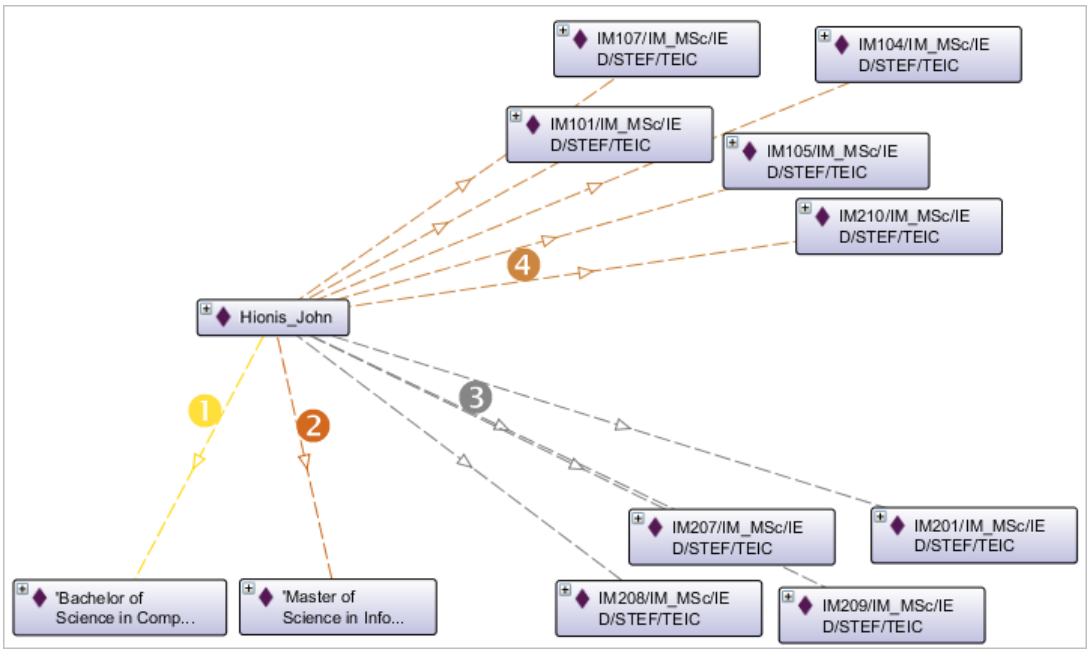


Figure 69: Graph with relations that student *Hionis_John* has with instances from other classes

Instances of classes Course and Syllabus

These are core classes in our ontology, thus a large number of instances have been created with courses and their syllabi which are included in study programs of *ProgramofStudy* sub-tree. In Course class we created 155 instances total (*Figure 70*), 64 of them are included in study program *Bachelor of Science in Informatics Engineering of TEI of Crete*, 60 courses are included in study program *Bachelor of Science in Computer Science of University of Crete*, 3 courses are included in study program *Bachelor of Science in Psychology of University of Crete*, 20 courses are included in study program *Master of Science in Informatics & Multimedia of TEI of Crete*, 2 courses are included in study program *Master of Science in Computer Science and Engineering of University of Crete*, 1 course for study program *Master of Science in Computer Science and Engineering of University of Crete*, and 1 course for study program *PhD in Clinical Psychology of University of Crete* (*Figure 70*).

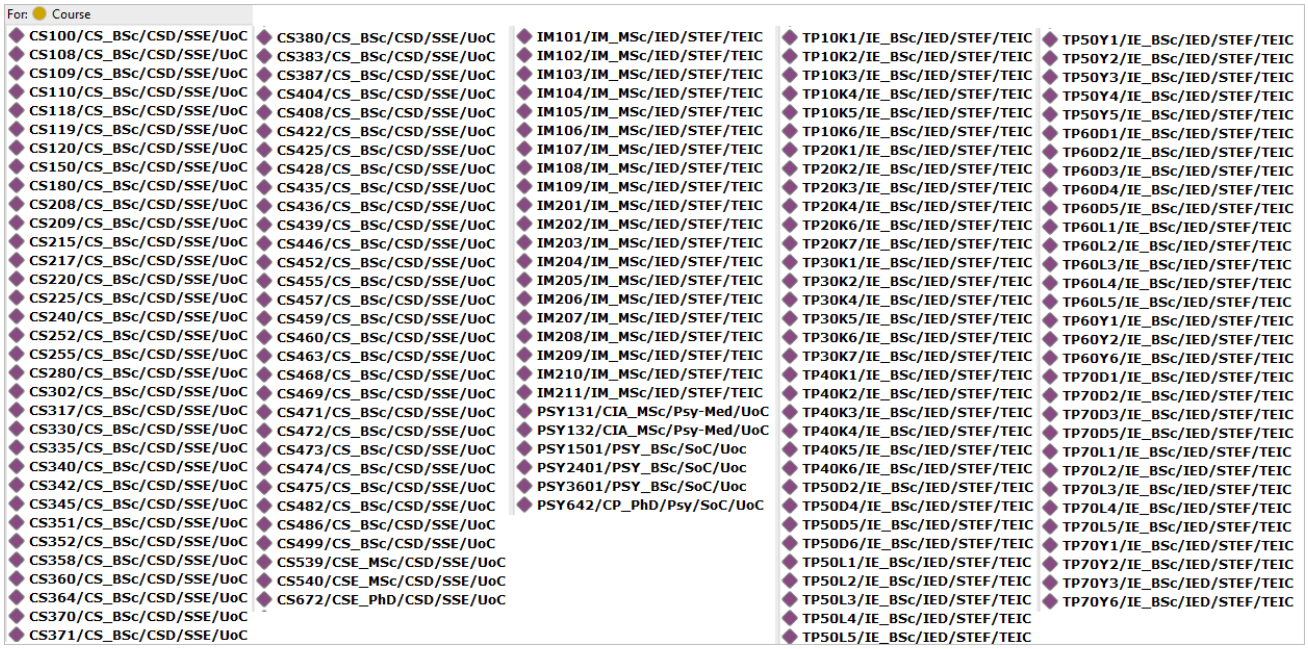


Figure 70: Instances of *Course* class

Each Course establishes connections with instances of other classes, as shown in red points in *Figure 71* below: (1) Study program that includes the course, (2) Department that offers the course, (3) Syllabus of the course, (4) Knowledge Areas (KA) that course covers, and (6) data properties of the course.

Each Course is connected with its Syllabus which is an instance of class `Syllabus` by using the relation `hasSyllabus`. When a course has multiple Syllabus versions, it is connected with more than one instances of Syllabus class as shown in points 3 in *Figure 72*. Also courses that have prerequisites are connected with other Courses using the relation `hasPrerequisite` as shown in *Figure 72*: line with point 2 shows that course *CS335* has the course *CS118* as prerequisite and lines with point 1 show that course CS335 is prerequisite in courses *CS455*, *CS439*, *CS435*, *CS539*, and *CS436*.

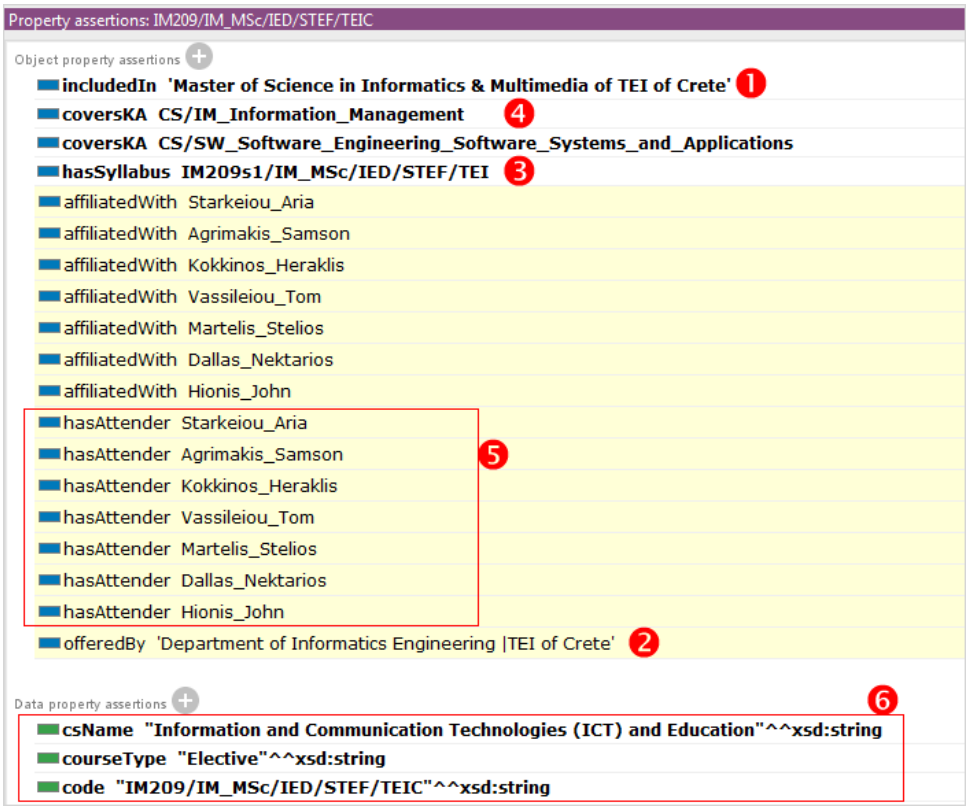


Figure 71: Properties of Course IM209/IM_MSc/IED/TEIC

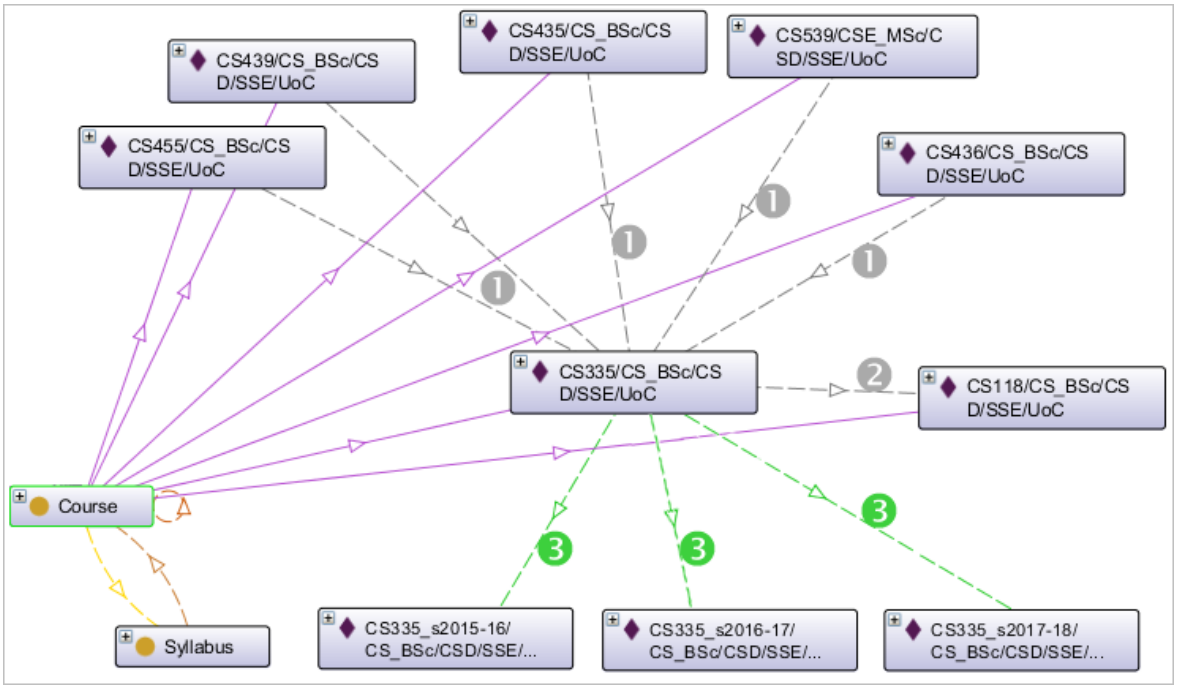


Figure 72: Prerequisites and Syllabi of Course CS335/CS_BSc/CSD/SSE/UoC

Instances of classes in Event sub-tree

In `Event` sub-tree, some instances in `TeachingEvent` and `EvaluationEvent` sub-classes have been created, as shown in *Figure 73*:

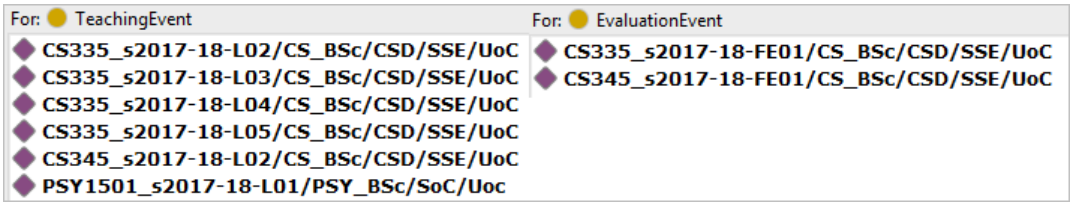


Figure 73: Instances in TeachingEvent and EvaluationEvent sub-classes

Each instance of `TeachingEvent` is connected with instances of classes `Professor`, `Assistant`, `Topic` and `LearningOutcome`, as shown in red points in *Figure 74*: (1) Teacher of Event, (2) Assistant of Event, (3) Topics of Event, (4) Learning Outcomes of Event, (5) data properties of Event.

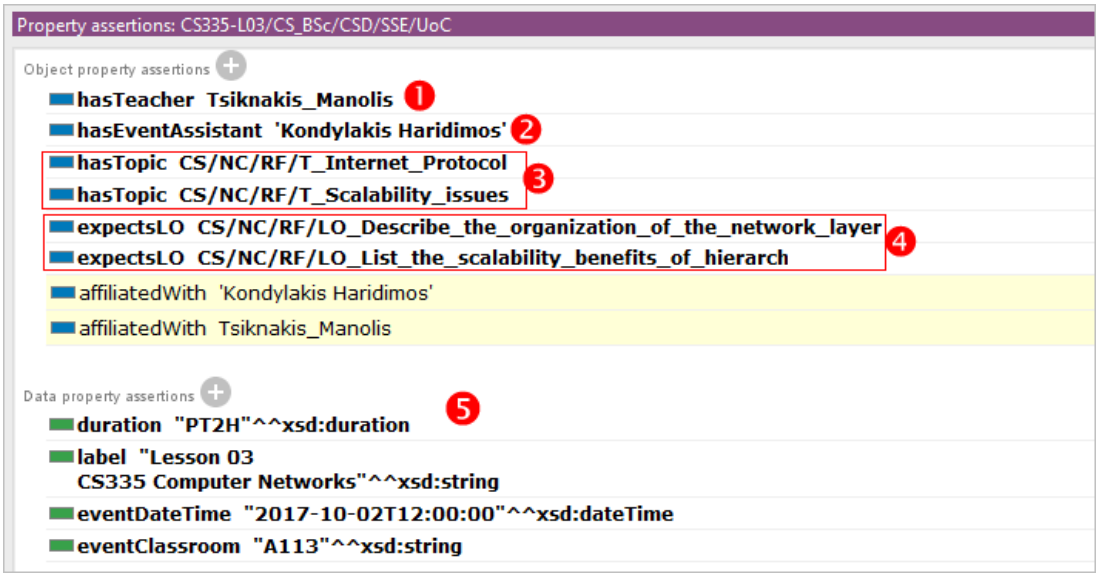


Figure 74: Property assertions of TeachingEvent CS335-L03

The class `TeachingEvaluationMethod` includes 8 instances that correspond to the the different kinds of teaching methods and learning activities within a course, as shown in *Figure 75*.

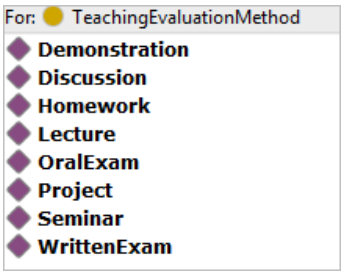


Figure 75: Instances of class *TeachingEvaluationMethod*

Instances of classes Topic and LearningOutcome

These classes include instances of Topics and Learning Outcomes related to *Computer Science* and *Psychology*.

`Topic` class includes 35 instances (*Figure 76*) which are establish connections with instances of classes `KnowledgeUnit`, `Syllabus` and `TeachingEvent` with relationships `listedInKU`, `coveredInCourse` and `hasTopic` respectively (*Figure 77*).

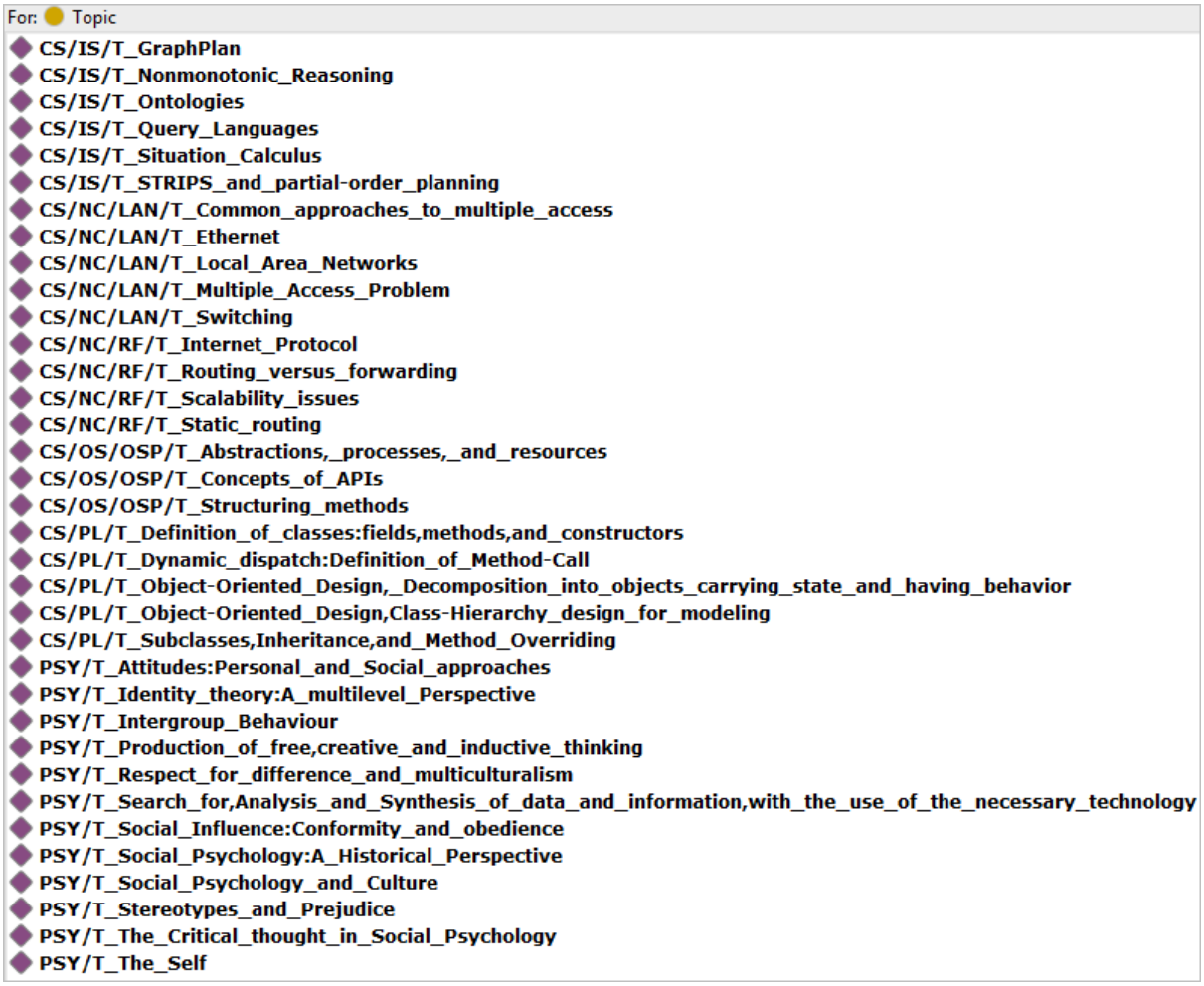


Figure 76: Instances of Topic class

Figure 77 depicts the property assertions of Topic "*Object-Oriented Design, Class hierarchy design for modeling*" which is listed in Knowledge Unit (KU) "*OOP Object Oriented Programming*" and being covered in Courses *TP10K4*, *IM201*, *CS252*, *TP50L5*, *CS150*, *CS343*. These courses are included in different study programs, as their first letters point out.

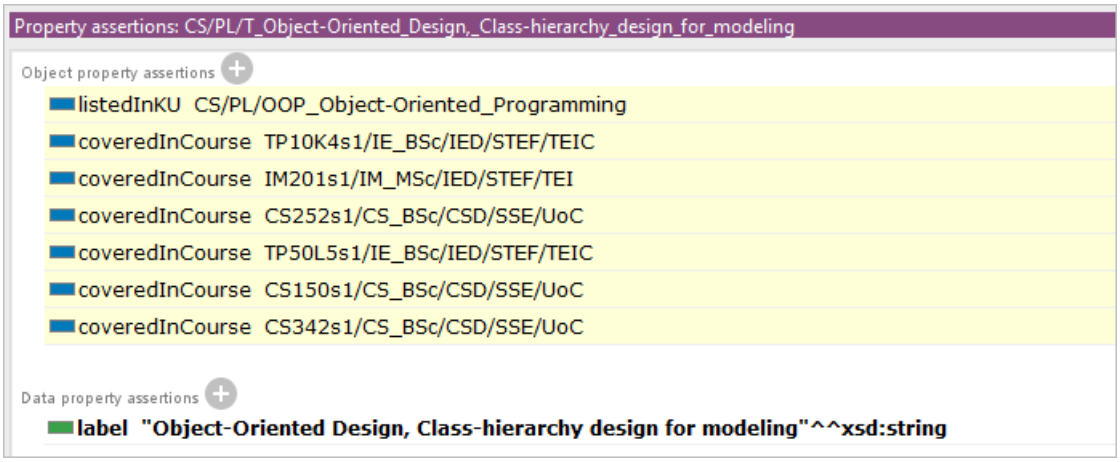


Figure 77: Property assertions of Topic "Object-Oriented Design, Class hierarchy design for modeling"

`LearningOutcome` class includes 23 instances (*Figure 78*) which are establish connections with instances of classes `KnowledgeUnit`, `Syllabus` and `TeachingEvent` with relationships `listedInKU`, `acquiredInCourse` and `expectsLO` respectively (*Figure 79*).

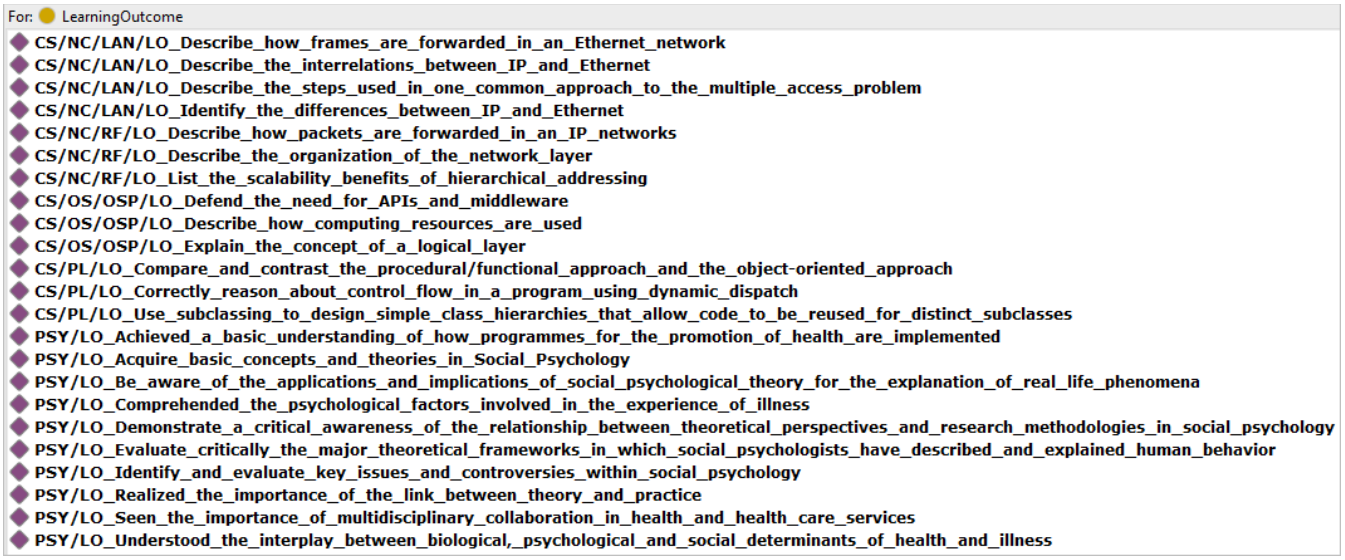


Figure 78: Instances of LearningOutcome class

Figure 79 depicts the property assertions of Learning Outcome "*Identify the differences between IP and Ethernet*" which is listed in Knowledge Unit (KU) "*LAN Local Area Networks*" and being covered in Courses *IM102*, *TP40K2*, *CS435*, *TP50D2*, *CS335*, which are included in different study programs.

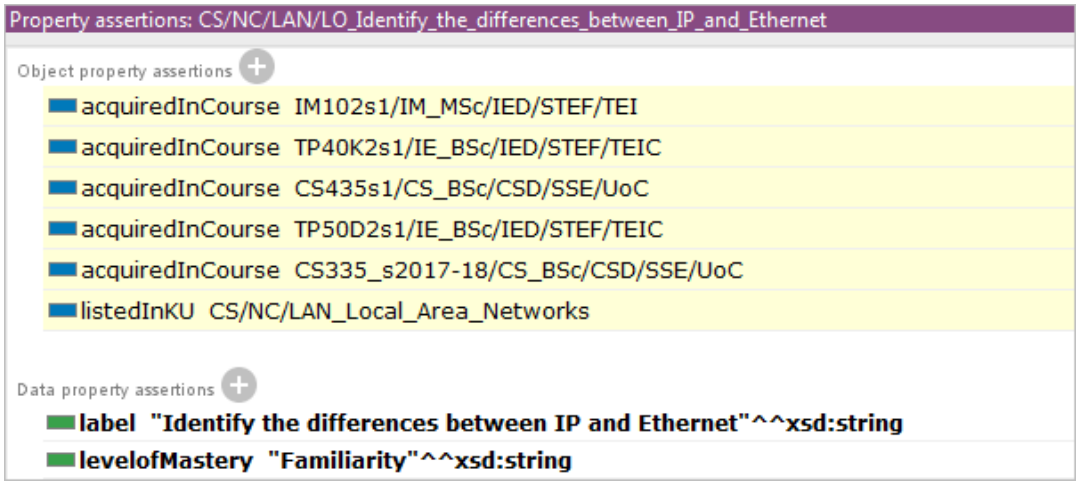


Figure 79: Property assertions of Learning Outcome "Identify the differences between IP and Ethernet"

Instances of class Resource

`Resource` class includes instances that are either used as Learing Material during events or they are publications of persons included in this ontology. A number of resources are also both, publications of persons that being used as learning material. The class includes 17 instances (*Figure 80*) that establish connection with instances of classes `Person`, `Syllabus` and `Event` with the relationships `dc:creator`, `dc:contributor`, and `hasLm`.

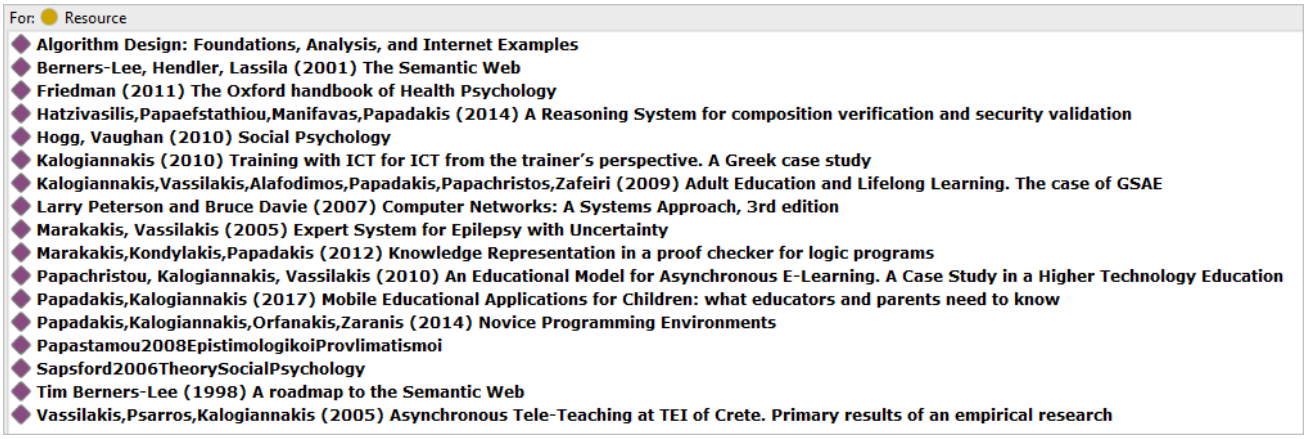


Figure 80: Instances of Resource class

Figure 81 and *Figure 82* depict the property assertions of two resources, with information such as: (1) label of the resource, (2) creators and contributors, (3) citation of the resource, (4) title, (5) date that is published, (6) media that is published and other data properties.

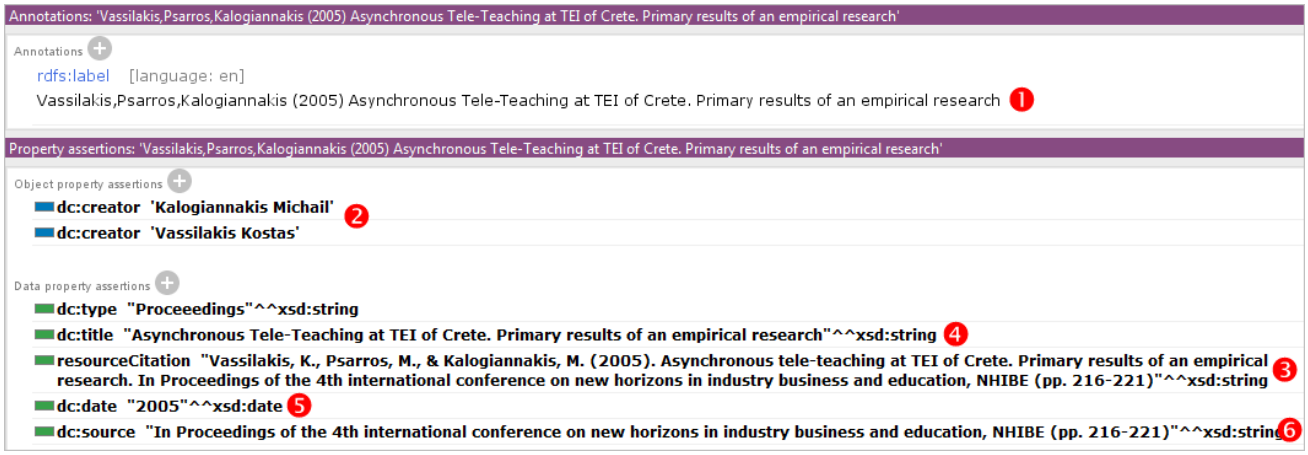


Figure 81: Property assertions of resource *Vassilakis,Psarros,Kalogiannakis (2005)*

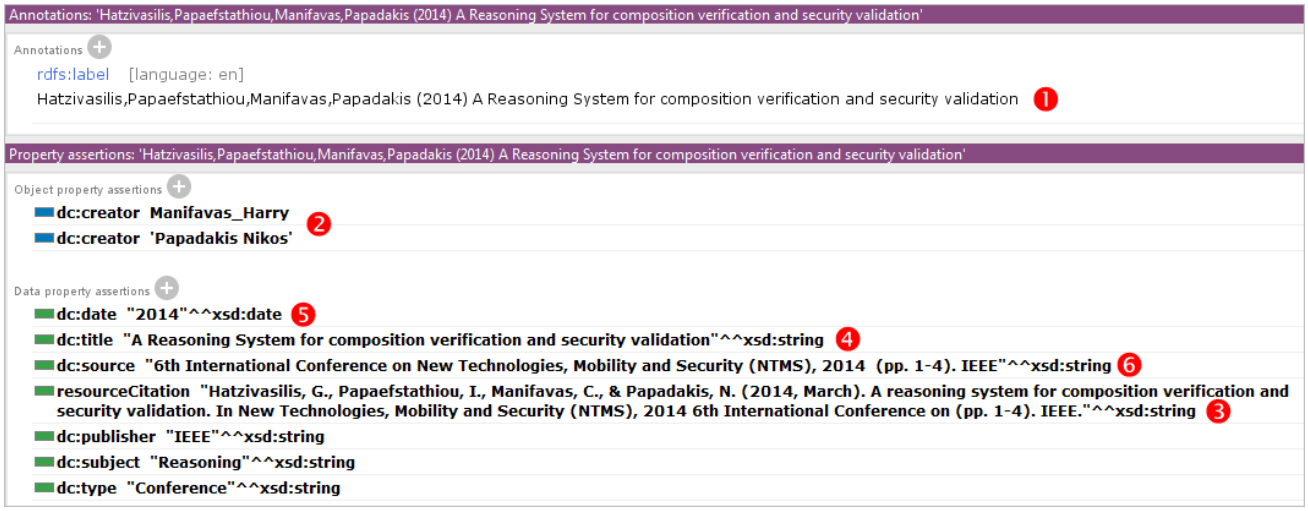


Figure 82: Property assertions of resource *Hatzivasilis,Papaefstathiou,Manifavas,Papadakis (2014)*

CHAPTER 4. RESULTS

After the building process has completed, the developed ontology comprises of 41 concepts in a taxonomy, 54 objects properties for establishing relationships and 76 data properties for describing concepts characteristics. In this chapter, the rest steps in ontology development process are analyzed, i.e. Evaluation and Documentation.

4.1. EVALUATION

During the construction process and between the phases, evaluation and refinement was performed. We followed internal technical evaluation approach where after the end of each phase, evaluation of definitions was performed by author and domain experts. The evaluation points included check of the ontology structure, the syntax of definitions and the content in the definitions [99]. Special attention was paid to the *evaluation of three Cs*: *consistency*, *completeness* and *conciseness*. Consequently, the refinement of the ontology based on the comments of domain experts. After the construction process has completed, the final evaluation of the developed ontology has taken place.

The final technical evaluation included, except manual examination, software tools that offer automate test against the most common errors and pitfalls in ontology development. It must be noted that, during ontology building we used the *HermiT* reasoner that comes with Protégé 5 [24]. When activating the reasoner, the ontology is tested for consistency, subsuption and satisfiability. An example is illustrated in *Figure 83* which is the explanation message for an inconsistency error that the reasoner returns when a student is enrolled in a Bachelor (point 1) and a Master program (point 2) which is not accepted, based on some restrictions we have made (point 3).

Figure 83: HermiT's explanation message for an inconsistency error

Secondly, we used *OOPS!* (*OntOlogy Pitfalls Scanner!*)[43], an automated tool that helps authors to detect common pitfalls in ontology development [100]. *OOPS!* is a web-based tool that scans ontology for potential pitfalls[44] that could lead to modeling errors. The choice (given as bullet) "*Category for Evaluation*" provides groups of pitfalls classified in dimensions and evaluating criteria (*Figure 84*).

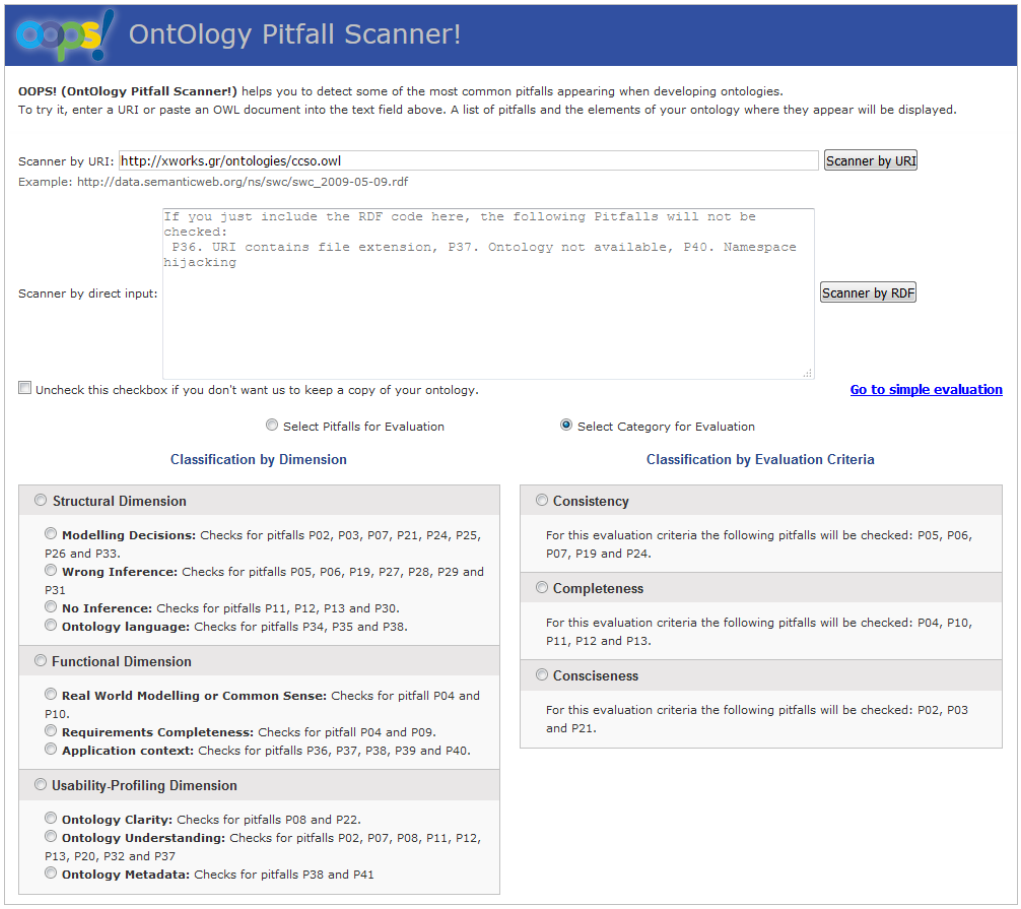


Figure 84: Pitfalls classification in OOPS

This feature is used to test the *structural*, *functional*, and *usability-profiling* dimensions of our ontology as well as to evaluate its *consistency*, *completeness* and *conciseness*. The results were very good while only some minor pitfalls have been noticed due to its unofficial release and support, as shown in *Figure 85*.

[43] OOPS! (OntOlogy Pitfalls Scanner!) : http://oops.linkeddata.es
[44] OOPS! catalogue of common pitfalls: http://oops.linkeddata.es/catalogue.jsp

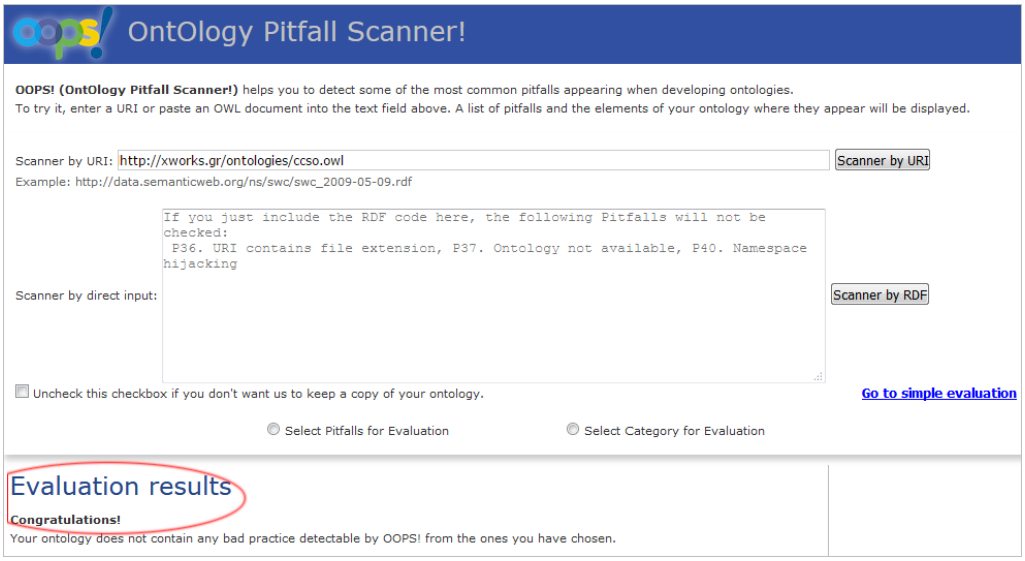


Figure 85: Evaluation results when testing the *consistency*, *completeness* and *conciseness* of CCSO with OOPS

After technical validation has completed, the ontology evaluated against the set of competency questions that have been made in the beginning of the development. The corresponding SPARQL queries, based on the competency questions, are executed and examined the relevance and correctness of answers given by ontology. Two different environments have been used for executing the queries: the *Snap SPARQL Query* tab in Protégé (*Figure 86*) and the *Virtuoso SPAQL endpoint* (*Figure 87*) where the ontology has been uploaded. In *Figure 86* the environment of *Snap SPARQL Query* tab (*point 1*) in Protégé is depicted, with a sample SPARQL query (*point 3*) and its results (*point 4*). It must be noted, that we used the *Snap SPARQL Query* tab that works with the reasoner and returns inferred results, rather than *SPAQL Query* tab (*point 2, Figure 86*) which does not.

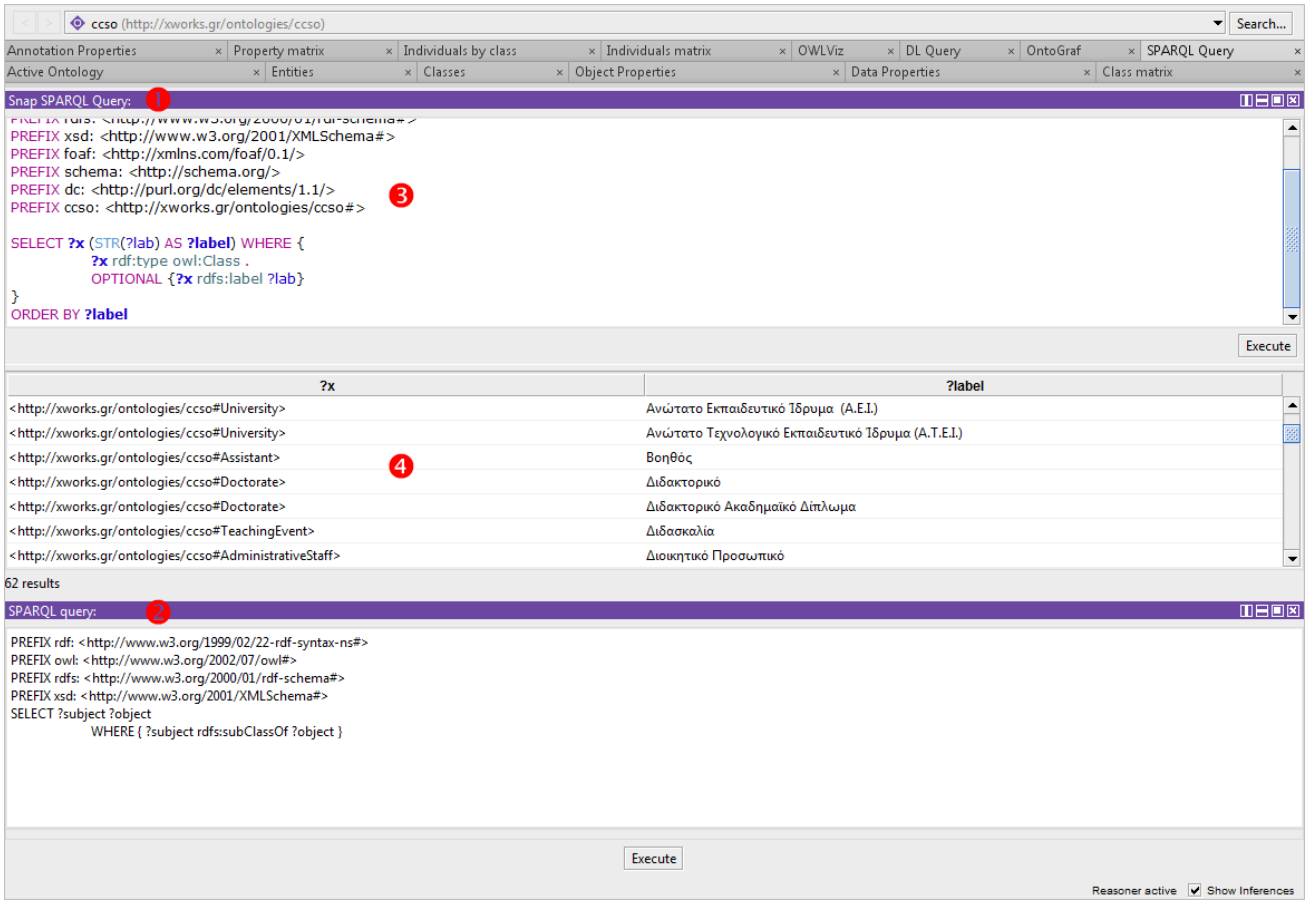


Figure 86: Environment of *Snap SPARQL Query* tab in Protégé

In *Figure 87* the environment of *Virtuoso SPAQL endpoint* is depicted: (1) URL address of CCSO Virtuoso SPARQL Endpoint, (2) Graph IRI of CCSO, (3) sample SPARQL query, (4) results of the query in HTML format which is depicted in another tab of browser.

The URL for *Virtuoso SPAQL endpoint* is http://83.212.124.52:8890/sparql and the Graph IRI for CCSO: http://localhost:8890/CCSO

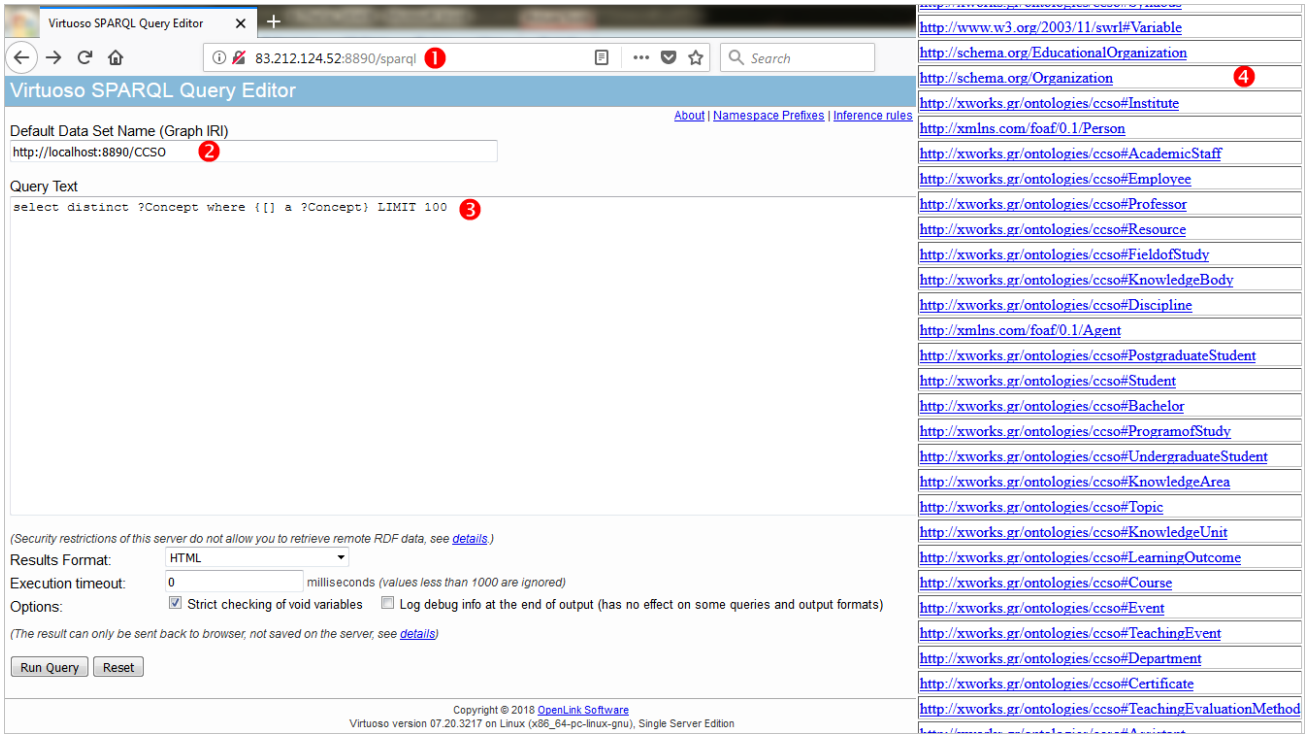


Figure 87: Environment of *Virtuoso SPARQL Endpoint*

Querying ontology with Competency Questions

Some extra competency questions have been added in the list of questions that had been defined in the beginning of the ontology building, in order the ontology to be tested against a wide variety of scenarios.

The complete list of *Competency Questions* is following, classified in four sub-sets according the related class or the purpose, and afterwards the testing process is illustrated. The SPARQL Queries that built from the questions, the execution of the Queries and examination of the answers are fully reported.

Questions related to Educational Organization

1. Which study programs does a Department provide?
2. Which Courses does a Department offer?
3. Who are the alumni of a Department?
4. Who are the employees of a Department?
5. Who are members of the Academic Staff of a Department?

Questions related to Person

6. Which degrees does a Person have?
7. Which Courses does a Professor teach?
8. Which Courses does a Professor supervise on?
9. Which Courses does an Assistant assist in?
10. Which Publications has a Person authored?
11. Which Courses does a Student attend?
12. Which Courses has a Student successfully completed?

Questions related to Course and Syllabus

13. Who are the Instructors of a Course?
14. Who are the Assistants of a Course?
15. Who are the Students that attend a Course?
16. Which are the Syllabi of a Course?
17. Which are the Topics and Learning Outcomes of a Course?

Questions related to study planning

18. Which Courses are included in a study program?
19. Which are the core Courses in a study program?
20. Which are the prerequisites of a Course?
21. Which Courses have prerequisites (and what are they) in a study program?
22. Which Courses cover a Knowledge Area?
23. Which Courses cover a Topic?
24. Which Courses aim to a Learning Outcome?
25. How many credits has a Student got so far?
26. Can a Student enroll in a Course?
27. Can a Person register in a study program for a degree?

The structure of a typical SPAQL Query is depicted in *Table 7* where:

- *Prefix declarations*, used for abbreviating URIs
- *Dataset definition*, states what RDF graph is being queried
- *SELECT* result clause, defines what information to return from the query

- *Query pattern*, specifies what to query for in the underlying dataset
- *Optional* part tries to match a graph pattern, but doesn't fail the whole query if the optional match fails
- *Query modifiers*, are slicing, ordering, or rearranging query results

```
PREFIX eg: <http://example.com/resources/>   #prefix declarations
...
FROM ...     # dataset definition
SELECT ...   # result clause
WHERE {
OPTIONAL { ... }
                ...     }
ORDER BY ... # query modifiers
```

Table 7: The structure of a typical SPAQL Query

In the beginning of each SPAQL Query, the following prefixes have been added for abbreviating URIs and namespaces in order to deal with them more elegantly:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX schema: <http://schema.org/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ccso: <http://xworks.gr/ontologies/ccso#>
```

Moreover, the following list of abbreviations has been used, corresponding to the most common entities that are involved in queries:

- **PS** is abbreviation for *Person*
- **Std** is abbreviation for *Student*
- **PF** is abbreviation for *Professor*
- **CA** is abbreviation for *Course Assistant*
- **PS** is abbreviation for *ProgramofStudy*
- **CS** is abbreviation for *Course*
- **SL** is abbreviation for *Syllabus*
- **Dpt** is abbreviation for *Department*

Questions related to Educational Organization

1. Which study programs does a Department provide?

(A)

```
SELECT  ?label  ?PS
WHERE  { ?DPT schema:identifier ?id .
         ?DPT ccso:providesProgram ?PS .
OPTIONAL { ?PS rdfs:label ?label  } .
         FILTER regex(str(?id), "IED/STEF/TEIC") }
```

```
SELECT ?label ?PS
  WHERE { ?DPT schema:identifier ?id .
          ?DPT ccso:providesProgram ?PS .
  OPTIONAL { ?PS rdfs:label ?label } .
    FILTER regex(str(?id), "IED/STEF/TEIC") }
```

?label	?PS
Bachelor of Science in Informatics Engineering of TEI of Crete@en	<http://xworks.gr/ontologies/ccso#BSc_Informatics_Engineering/IE_BSc/IED/STEF/TEIC>
Master of Science in Informatics & Multimedia of TEI of Crete@en	<http://xworks.gr/ontologies/ccso#MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC>

Comment: Ontology returns the labels and IRIs of the 2 study programs provided by Department with id `IED/STEF/TEIC`. The individuals are from different classes (i.e. *Bachelor*, *Master*).

(B)

```
SELECT  ?label  ?PS
 WHERE  { ?DPT schema:identifier ?id .
          ?DPT ccso:providesProgram ?PS .
  OPTIONAL { ?PS rdfs:label ?label  } .
  FILTER regex(str(?id), "CSD/SSE/UoC") }
```

label	PS
"Bachelor of Science in Computer Science of University of Crete"@en	http://xworks.gr/ontologies/ccso#BSc_Computer_Science/CS_BSc/CSD/SSE/UoC
"Master of Science in Computer Science and Engineering of University of Crete"@en	http://xworks.gr/ontologies/ccso#MSc_Computer_Science_and_Engineering/CSE_MSc/CSD/SSE/UoC
"PhD in Computer Science and Engineering of University of Crete"@en	http://xworks.gr/ontologies/ccso#PhD_Computer_Science_and_Engineering/CSE_PhD/CSD/SSE/UoC

Comment: In this query, Ontology returns the labels and IRIs of the 3 study programs provided by Department with id `CSD/SSE/UoC`. The individuals are from different classes (i.e. *Bachelor*, *Master*, *Doctorate*).

2. Which Courses does a Department offer?

(A)

```
SELECT  ?CSname ?CS
   WHERE  { ?DPT schema:identifier ?id .
            ?DPT ccso:offersCourse ?CS .
OPTIONAL { ?CS ccso:csName ?CSname } .
         FILTER regex(str(?id), "CSD/SSE/UoC") }
  ORDER BY ?CS
```

```
SELECT ?CSname ?CS
  WHERE { ?DPT schema:identifier ?id .
          ?DPT ccso:offersCourse ?CS .
OPTIONAL { ?CS ccso:csName ?CSname } .
    FILTER regex(str(?id), "CSD/SSE/UoC") }
ORDER BY ?CS
```

?CSname	?CS
Algorithms in Bioinformatics^^xsd:string	<http://xworks.gr/ontologies/ccso#CS482/CS_BSc/CSD/SSE/UoC>
Principles of Distributed Computing^^xsd:string	<http://xworks.gr/ontologies/ccso#CS486/CS_BSc/CSD/SSE/UoC>
Bachelor's Thesis^^xsd:string	<http://xworks.gr/ontologies/ccso#CS499/CS_BSc/CSD/SSE/UoC>
Advanced Topics on Wireless Networks and Mobile SystemsCourse in Computer Science an...	<http://xworks.gr/ontologies/ccso#CS539/CSE_MSc/CSD/SSE/UoC>
Advanced Topics in Programming Languages Development^^xsd:string	<http://xworks.gr/ontologies/ccso#CS540/CSE_MSc/CSD/SSE/UoC>
Advanced Topics in Computational Vision^^xsd:string	<http://xworks.gr/ontologies/ccso#CS672/CSE_PhD/CSD/SSE/UoC>

63 results

Comment: Ontology responds that 63 courses are provided by department with id `CSD/SSE/UoC`. The answer includes course's name (CSname) and course's IRI (CS).

(B)

```
SELECT  ?CSname ?CS ?PSlabel
   WHERE  { ?DPT schema:identifier ?id .
      ?DPT ccso:offersCourse ?CS .
OPTIONAL { ?CS ccso:csName ?CSname } .
      ?CS ccso:includedIn ?PS .
      ?DPT ccso:providesProgram ?PS .
 OPTIONAL { ?PS rdfs:label ?PSlabel } .
      FILTER regex(str(?id), "IED/STEF/TEIC") }
 ORDER BY ?CS
```

CSname	CS	PSlabel
"Project Management and Research Methodologies"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#IM101/IM_MSc/IED/STEF/TEIC	"Master of Science in Informatics & Multimedia of TEI of Crete"@en
"Computer Networks"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#IM102/IM_MSc/IED/STEF/TEIC	"Master of Science in Informatics & Multimedia of TEI of Crete"@en
"Advanced Multimedia Technologies"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#IM103/IM_MSc/IED/STEF/TEIC	"Master of Science in Informatics & Multimedia of TEI of Crete"@en
"Advanced Software Engineering"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#IM104/IM_MSc/IED/STEF/TEIC	"Master of Science in Informatics & Multimedia of TEI of Crete"@en
"Semantic Web"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#IM105/IM_MSc/IED/STEF/TEIC	"Master of Science in Informatics & Multimedia of TEI of Crete"@en

Comment: Ontology responds that 84 courses are provided by department with id `IED/STEF/TEIC`. This time we changed a little the query and the answer includes the course's name (CSname) , course's IRI (CS) and the study program (PSlabel) this course includes.

3. Who are the alumni of a Department?

(A)

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type foaf:Person .
 OPTIONAL { ?PR foaf:firstName ?name } .
 OPTIONAL { ?PR foaf:lastName ?surname }.
 OPTIONAL { ?PR schema:email ?email } .
      ?PR ccso:alumnusOf ?DPT .
      ?DPT schema:identifier ?id .
      FILTER regex(str(?id), "IED/STEF/TEIC") }
  ORDER BY ?surname
```

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type foaf:Person .
 OPTIONAL { ?PR foaf:firstName ?name } .
 OPTIONAL { ?PR foaf:lastName ?surname }.
 OPTIONAL { ?PR schema:email ?email } .
      ?PR ccso:alumnusOf ?DPT .
      ?DPT schema:identifier ?id .
      FILTER regex(str(?id), "IED/STEF/TEIC") }
 ORDER BY ?surname
```

?PR	?surname	?name	?email
<http://xworks.gr/ontologies/ccso#Agrimakis_Samson>	Agrimakis^^xsd:string	Samson^^xsd:string	
<http://xworks.gr/ontologies/ccso#Kokkinos_Heraklis>	Kokkinos^^xsd:string	Heraklis^^xsd:string	Kokkinos_Heraklis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Kornaros_George>	Kornaros^^xsd:string	George^^xsd:string	kornaros@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Martelis_Stelios>	Martelis^^xsd:string	Stelios^^xsd:string	Martelis_Stelios@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Raderis_Jim>	Raderis^^xsd:string	Jim^^xsd:string	Raderis_Jim@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Sandi_Elina>	Sandi^^xsd:string	Elina^^xsd:string	Sandi_Elina@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Starkeiou_Alexandros>	Starkeiou^^xsd:string	Alexandros^^xsd:string	Starkeiou_Alexandros@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Starkeiou_Aria>	Starkeiou^^xsd:string	Aria^^xsd:string	Starkeiou_Aria@edu.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Vassileiou_Tom>	Vassileiou^^xsd:string	Tom^^xsd:string	Vassileiou_Tom@ie.teicrete.gr^^xsd:anyURI

9 results

Comment: There are 9 persons who are alumni of department with id `IED/STEF/TEIC`. The answer includes the elements alumnus' IRI, surname, name and email.

(B)

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type foaf:Person .
     ?PR foaf:firstName ?name .
     ?PR foaf:lastName ?surname .
   OPTIONAL { ?PR schema:email ?email } .
     ?PR ccso:alumnusOf ?DPT .
```

```
        ?DPT schema:identifier ?id .
        FILTER regex(str(?id), "CSD/SSE/UoC") }
    ORDER BY ?surname
```

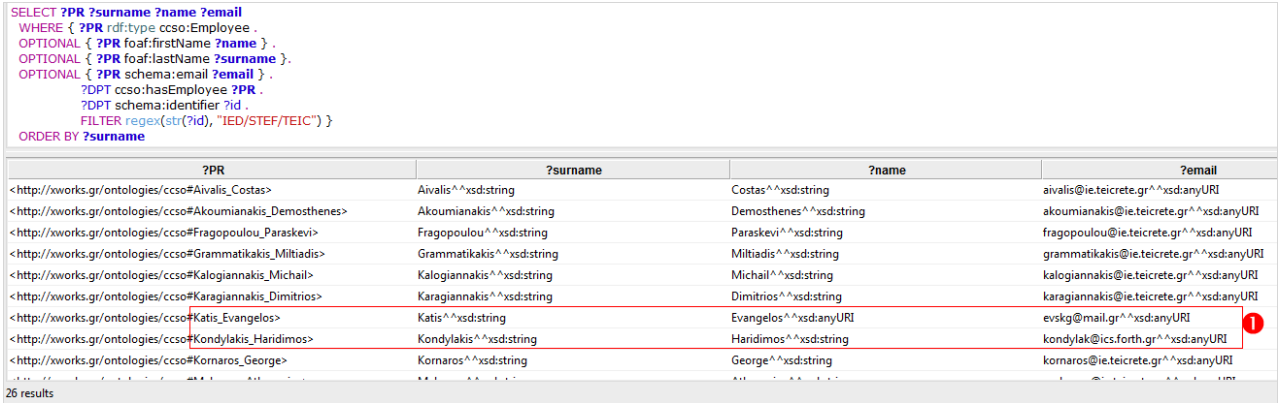
PR	surname	name	email
http://xworks.gr/ontologies/ccso#Akoumianakis_Demosthenes	"Akoumianakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Demosthenes"^^<http://www.w3.org/2001/XMLSchema#string>	"akoumianakis@ie.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Dallas_Nektarios	"Dallas"^^<http://www.w3.org/2001/XMLSchema#string>	"Nektarios"^^<http://www.w3.org/2001/XMLSchema#string>	"Dallas_Nektarios@csd.uoc.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Hionis_John	"Hionis"^^<http://www.w3.org/2001/XMLSchema#string>	"John"^^<http://www.w3.org/2001/XMLSchema#string>	"Hionis_John@ie.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Kalogiannakis_Michail	"Kalogiannakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Michail"^^<http://www.w3.org/2001/XMLSchema#string>	"kalogiannakis@ie.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Kondylakis_Haridimos	"Kondylakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Haridimos"^^<http://www.w3.org/2001/XMLSchema#string>	"kondylak@ics.forth.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Kornaros_George	"Kornaros"^^<http://www.w3.org/2001/XMLSchema#string>	"George"^^<http://www.w3.org/2001/XMLSchema#string>	"kornaros@ie.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Lionis_Joshua	"Lionis"^^<http://www.w3.org/2001/XMLSchema#string>	"Joshua"^^<http://www.w3.org/2001/XMLSchema#string>	"Lionis_Joshua@csd.uoc.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Papadakis_Nikos	"Papadakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Nikolaos"^^<http://www.w3.org/2001/XMLSchema#string>	"papadakisn@ie.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Starkeiou_Michael	"Starkeiou"^^<http://www.w3.org/2001/XMLSchema#string>	"Michael"^^<http://www.w3.org/2001/XMLSchema#string>	"Starkeiou_Michael@csd.uoc.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Triantafillou_Irene	"Triantafillou"^^<http://www.w3.org/2001/XMLSchema#string>	"Irene"^^<http://www.w3.org/2001/XMLSchema#string>	"Triantafillou_Irene@csd.uoc.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Tsiknakis_Manolis	"Tsiknakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Manolis"^^<http://www.w3.org/2001/XMLSchema#string>	"tsiknakis@ie.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Vassilakis_Kostas	"Vassilakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Konstantinos"^^<http://www.w3.org/2001/XMLSchema#string>	"kostas@cs.teicrete.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>
http://xworks.gr/ontologies/ccso#Vassilakis_Vassilis	"Vassilakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Vassilis"^^<http://www.w3.org/2001/XMLSchema#string>	"Vassilakis_Vassilis@csd.uoc.gr"^^<http://www.w3.org/2001/XMLSchema#anyURI>

Comment: Results show that the department with id `CSD/SSE/UoC` has 13 alumni. The answer includes the elements alumnus' IRI, surname, name and email.

4. Who are the employees of a Department?

(A)

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type ccso:Employee .
  OPTIONAL { ?PR foaf:firstName ?name } .
  OPTIONAL { ?PR foaf:lastName ?surname }.
  OPTIONAL { ?PR schema:email ?email } .
      ?DPT ccso:hasEmployee ?PR .
      ?DPT schema:identifier ?id .
      FILTER regex(str(?id), "IED/STEF/TEIC") }
  ORDER BY ?surname
```



?PR	?surname	?name	?email
<http://xworks.gr/ontologies/ccso#Aivalis_Costas>	Aivalis^^xsd:string	Costas^^xsd:string	aivalis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Akoumianakis_Demosthenes>	Akoumianakis^^xsd:string	Demosthenes^^xsd:string	akoumianakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Fragopoulou_Paraskevi>	Fragopoulou^^xsd:string	Paraskevi^^xsd:string	fragopoulou@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Grammatikakis_Miltiadis>	Grammatikakis^^xsd:string	Miltiadis^^xsd:string	grammatikakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Kalogiannakis_Michail>	Kalogiannakis^^xsd:string	Michail^^xsd:string	kalogiannakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Karagiannakis_Dimitrios>	Karagiannakis^^xsd:string	Dimitrios^^xsd:string	karagiannakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Katis_Evangelos>	Katis^^xsd:string	Evangelos^^xsd:anyURI	evskg@mail.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Kondylakis_Haridimos>	Kondylakis^^xsd:string	Haridimos^^xsd:string	kondylak@ics.forth.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Kornaros_George>	Kornaros^^xsd:string	George^^xsd:string	kornaros@ie.teicrete.gr^^xsd:anyURI

26 results

Comment: Results show that the department `IED/STEF/TEIC` has 26 employees. We notice that employees `Katis_Evangelos` and `Kondylakis_Haridimos` (*Point 1*) are included since they are *Course Assistants*.

(B)

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type ccso:Employee .
  OPTIONAL { ?PR foaf:firstName ?name } .
  OPTIONAL { ?PR foaf:lastName ?surname }.
  OPTIONAL { ?PR schema:email ?email } .
      ?DPT ccso:hasEmployee ?PR .
      ?DPT schema:identifier ?id .
      FILTER regex(str(?id), "CSD/SSE/UoC") }
  ORDER BY ?surname
```

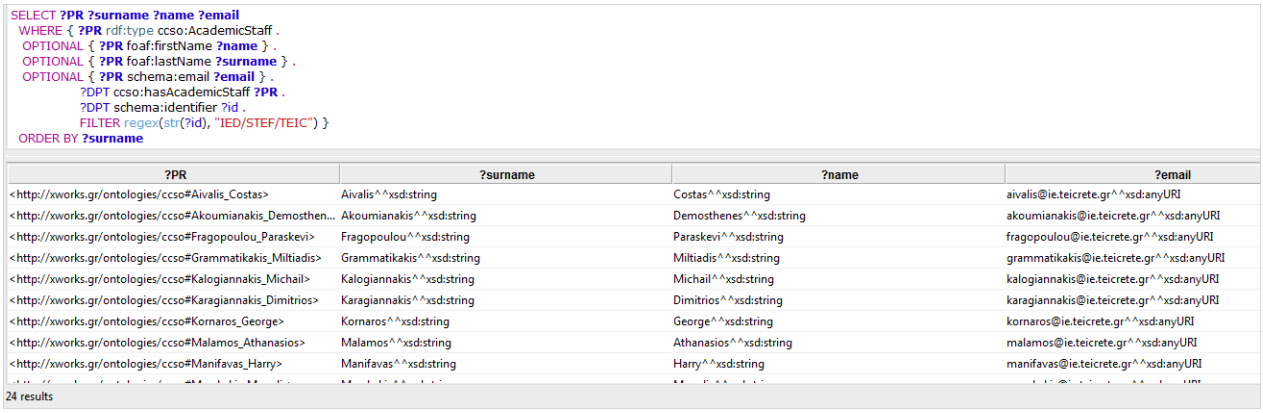
/ccso#Kalogiannakis_Michail	"Kalogiannakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Michail"^^<http://www.w3.org/2001/XMLSchema#string>	"kalogiannakis@ie.teicrete.gr"^^<
/ccso#Katevenis_Manolis	"Katevenis"^^<http://www.w3.org/2001/XMLSchema#string>	"Manolis"^^<http://www.w3.org/2001/XMLSchema#string>	"kateveni@csd.uoc.gr"^^<http://ww
/ccso#Kondylakis_Haridimos	"Kondylakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Haridimos"^^<http://www.w3.org/2001/XMLSchema#string>	"kondylak@ics.forth.gr"^^<http://
/ccso#Markatos_Evangelos	"Markatos"^^<http://www.w3.org/2001/XMLSchema#string>	"Evangelos"^^<http://www.w3.org/2001/XMLSchema#string>	
/ccso#Mouxtaris_Athanasios	"Mouxtaris"^^<http://www.w3.org/2001/XMLSchema#string>	"Athanasios"^^<http://www.w3.org/2001/XMLSchema#string>	"mouchtar@csd.uoc.gr"^^<http://ww
/ccso#Papadopouli_Maria	"Papadopouli"^^<http://www.w3.org/2001/XMLSchema#string>	"Maria"^^<http://www.w3.org/2001/XMLSchema#string>	"mpap@csd.uoc.gr"^^<http://www.w3

Comment: Results show that the department `CSD/SSE/UoC` has 18 employees. We notice that employees `Kalogiannakis_Michail` and `Kondylakis_Haridimos` (*red rectangles*) are listed again which means that they are employed in this university too (as they are in IED/STEF/TEIC in *Query 4A*).

5. Who are the members of the Academic Staff of a Department?

(A)

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type ccso:AcademicStaff .
   OPTIONAL { ?PR foaf:firstName ?name } .
   OPTIONAL { ?PR foaf:lastName ?surname } .
   OPTIONAL { ?PR schema:email ?email } .
      ?DPT ccso:hasAcademicStaff ?PR .
      ?DPT schema:identifier ?id .
      FILTER regex(str(?id), "IED/STEF/TEIC") }
  ORDER BY ?surname
```



?PR	?surname	?name	?email
<http://xworks.gr/ontologies/ccso#Aivalis_Costas>	Aivalis^^xsd:string	Costas^^xsd:string	aivalis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Akoumianakis_Demosthen...	Akoumianakis^^xsd:string	Demosthenes^^xsd:string	akoumianakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Fragopoulou_Paraskevi>	Fragopoulou^^xsd:string	Paraskevi^^xsd:string	fragopoulou@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Grammatikakis_Miltiadis>	Grammatikakis^^xsd:string	Miltiadis^^xsd:string	grammatikakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Kalogiannakis_Michail>	Kalogiannakis^^xsd:string	Michail^^xsd:string	kalogiannakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Karagiannakis_Dimitrios>	Karagiannakis^^xsd:string	Dimitrios^^xsd:string	karagiannakis@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Kornaros_George>	Kornaros^^xsd:string	George^^xsd:string	kornaros@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Malamos_Athanasios>	Malamos^^xsd:string	Athanasios^^xsd:string	malamos@ie.teicrete.gr^^xsd:anyURI
<http://xworks.gr/ontologies/ccso#Manifavas_Harry>	Manifavas^^xsd:string	Harry^^xsd:string	manifavas@ie.teicrete.gr^^xsd:anyURI

24 results

Comment: Results show that the Academic Staff of department `IED/STEF/TEIC` has 24 members. We notice that assistants `Katis_Evangelos` and `Kondylakis_Haridimos` are not listed (as in *Query 4A*) since they are not member of Academic Staff.

(B)

```
SELECT ?PR ?surname ?name ?email
  WHERE { ?PR rdf:type ccso:AcademicStaff .
   OPTIONAL { ?PR foaf:firstName ?name } .
   OPTIONAL { ?PR foaf:lastName ?surname } .
   OPTIONAL { ?PR schema:email ?email } .
      ?DPT ccso:hasAcademicStaff ?PR .
      ?DPT schema:identifier ?id .
      FILTER regex(str(?id), "CSD/SSE/UoC") }
  ORDER BY ?surname
```

PR	surname	name	
s/ccso#Argyros_Antonis	"Argyros"^^<http://www.w3.org/2001/XMLSchema#string>	"Antonis"^^<http://www.w3.org/2001/XMLSchema#string>	"argyros@csd.uoc.gr"^^<http://
s/ccso#Christofides_Vassilis	"Christofides"^^<http://www.w3.org/2001/XMLSchema#string>	"Vassilis"^^<http://www.w3.org/2001/XMLSchema#string>	"christofides@csd.uoc.gr"^^<ht
s/ccso#Faturou_Panagiota	"Faturou"^^<http://www.w3.org/2001/XMLSchema#string>	"Panagiota"^^<http://www.w3.org/2001/XMLSchema#string>	"faturu@csd.uoc.gr"^^<http://
s/ccso#Georgakopoulos_George	"Georgakopoulos"^^<http://www.w3.org/2001/XMLSchema#string>	"George"^^<http://www.w3.org/2001/XMLSchema#string>	"ggeo@csd.uoc.gr"^^<http://www
s/ccso#Kalogiannakis_Michail	"Kalogiannakis"^^<http://www.w3.org/2001/XMLSchema#string>	"Michail"^^<http://www.w3.org/2001/XMLSchema#string>	"kalogiannakis@ie.teicrete.gr"
s/ccso#Katevenis_Manolis	"Katevenis"^^<http://www.w3.org/2001/XMLSchema#string>	"Manolis"^^<http://www.w3.org/2001/XMLSchema#string>	"kateveni@csd.uoc.gr"^^<http:/
s/ccso#Markatos_Evangelos	"Markatos"^^<http://www.w3.org/2001/XMLSchema#string>	"Evangelos"^^<http://www.w3.org/2001/XMLSchema#string>	
s/ccso#Mouxtaris_Athanasios	"Mouxtaris"^^<http://www.w3.org/2001/XMLSchema#string>	"Athanasios"^^<http://www.w3.org/2001/XMLSchema#string>	"mouchtar@csd.uoc.gr"^^<http:/

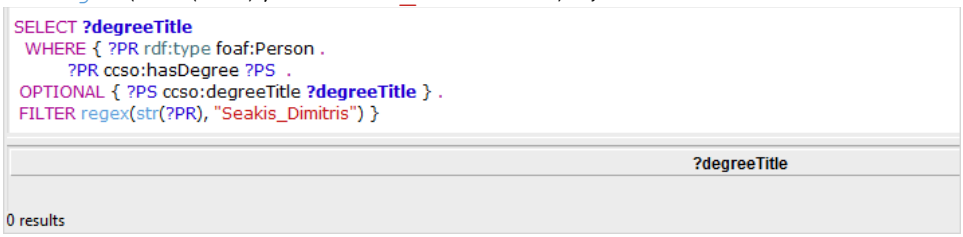
Comment: Results show that the Academic Staff of department `CSD/SSE/UoC` has 17 members. We notice that professor `Kalogiannakis_Michail` (*Query 4A*) is listed again since he is member of the academic staff of this department as in *Query 5A* too.

Questions related to Person

6. Which degrees does a Person have?

(A)

```
SELECT ?degreeTitle
  WHERE { ?PR rdf:type foaf:Person .
          ?PR ccso:hasDegree ?PS  .
  OPTIONAL { ?PS ccso:degreeTitle ?degreeTitle } .
  FILTER regex(str(?PR), "Seakis_Dimitris") }
```



Comment: Ontology returns no answers which mean that the person `Seakis_Dimitris` does not have any degree.

(B)

```
SELECT ?degreeTitle ?PR
  WHERE { ?PR rdf:type foaf:Person .
          ?PR foaf:lastName ?surname .
          ?PR ccso:hasDegree ?PS  .
  OPTIONAL { ?PS ccso:degreeTitle ?degreeTitle } .
          FILTER regex(str(?surname), "Starkeiou") }
```

degreeTitle	PR
"Bachelor of Science in Computer Science of University of Crete"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Michael
"Bachelor of Science in Informatics Engineering of TEI of Crete"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Aria
"Bachelor of Science in Informatics Engineering of TEI of Crete"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Alexandros
"Bachelor of Science in Psychology of University of Crete"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Catherine
"Master of Science in Computer Science and Engineering of University of Crete"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Michael
"Master of Science in School Psychology"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Catherine
"Doctor of Philosophy in Clinical Psychology"^^<http://www.w3.org/2001/XMLSchema#string>	http://xworks.gr/ontologies/ccso#Starkeiou_Catherine

Comment: This query responds all degrees, with a total of 7 degrees for persons with surname "Starkeiou". For example, the person `Starkeiou_Catherine` has 3 degrees, i.e. *BSc in Psychology*, *MSc in Psychology*, and *PhD in Clinical Psychology*.

7. Which Courses does a Professor teach?

(A)

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?PF rdf:type ccso:Professor .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
          ?PF ccso:instructorOf ?SL .
          ?SL ccso:syllabusOf ?CS .
          ?CS ccso:code ?CScode .
```

```
        ?CS ccso:includedIn ?PS .
    OPTIONAL { ?PS ccso:psName ?PSname } .
    OPTIONAL { ?CS ccso:csName ?CSname } .
    FILTER regex(str(?PFsurname), "Vassilakis" ) .          }
    ORDER BY ?CScode
```

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?PF rdf:type ccso:Professor .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
            ?PF ccso:instructorOf ?SL .
            ?SL ccso:syllabusOf ?CS .
            ?CS ccso:code ?CScode .
            ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
  OPTIONAL { ?CS ccso:csName ?CSname } .
  FILTER regex(str(?PFsurname), "Vassilakis" ) .          }
  ORDER BY ?CScode
```

?PSname	?CScode	?CSname
Master of Science in Informatics & Multimedia of Department of Informatics E...	IM209/IM_MSc/IED/STEF/TEIC^^xsd:string	Information and Communication Technologies (ICT) and Education
Bachelor of Science (BSc) in Informatics Engineering of Department of Informat...	TP10K4/IE_BSc/IED/STEF/TEIC^^xsd:string	Programming^^xsd:string
Bachelor of Science (BSc) in Informatics Engineering of Department of Informat...	TP20K4/IE_BSc/IED/STEF/TEIC^^xsd:string	Data Structures^^xsd:string

3 results

Comment: Results show that professor `Vassilakis` teaches in 3 courses in 2 study programs. The answer includes the properties course's code, course's name and the program's name.

(B)

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?PF rdf:type ccso:Professor .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
        ?PF ccso:instructorOf ?SL .
        ?SL ccso:syllabusOf ?CS .
        ?CS ccso:code ?CScode .
        ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
  OPTIONAL { ?CS ccso:csName ?CSname } .
  FILTER regex(str(?PFsurname), "Kalogiannakis" ) .       }
  ORDER BY ?CScode
```

PSname	CScode	CSname
"Bachelor of Science (BSc) in Computer Science of …"	"CS118/CS_BSc/CSD/SSE/UoC"	"Discrete Mathematics"
"Bachelor of Science (BSc) in Computer Science of …"	"CS150/CS_BSc/CSD/SSE/UoC"	"Programming"
"Master of Science in Informatics & Multimedia of …"	"IM106/IM_MSc/IED/STEF/TEIC"	"Usability Engineering"
"Master of Science in Informatics & Multimedia of …"	"IM209/IM_MSc/IED/STEF/TEIC"	"Information and Communication Technologies (ICT) and Education"
"Bachelor of Science (BSc) in Informatics Engineering of …"	"TP10K5/IE_BSc/IED/STEF/TEIC"	"Calculus"
"Bachelor of Science (BSc) in Informatics Engineering of …"	"TP20K7/IE_BSc/IED/STEF/TEIC"	"Discrete Mathematics"

Comment: Results show that professor `Kalogiannakis` teaches in 6 courses in 3 different study programs (*Points 1, 2, 3*). The answer includes the properties course's code, course's name and the program's name.

8. Which Courses does a Professor supervise on?

(A)

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?PF rdf:type ccso:Professor .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
        ?PF ccso:supervisorOf ?SL .
        ?SL ccso:syllabusOf ?CS .
  OPTIONAL { ?CS ccso:code ?CScode } .
        ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
```

```
        OPTIONAL { ?CS ccso:csName ?CSname } .
        FILTER regex(str(?PFsurname), "Papadakis" ) .  }
        ORDER BY ?CScode
```

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?PF rdf:type ccso:Professor .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
           ?PF ccso:supervisorOf ?SL .
           ?SL ccso:syllabusOf ?CS .
  OPTIONAL { ?CS ccso:code ?CScode } .
           ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
  OPTIONAL { ?CS ccso:csName ?CSname } .
  FILTER regex(str(?PFsurname), "Papadakis" ) . }
  ORDER BY ?CScode
```

?PSname	?CScode	?CSname
Master of Science in Informatics & Multimedia of Department of Inf...	IM104/IM_MSc/IED/STEF/TEIC^^xsd:string	Advanced Software Engineering^^xsd:string
Master of Science in Informatics & Multimedia of Department of Inf...	IM105/IM_MSc/IED/STEF/TEIC^^xsd:string	Semantic Web^^xsd:string
Bachelor of Science (BSc) in Informatics Engineering of Department ...	TP10K2/IE_BSc/IED/STEF/TEIC^^xsd:string	Modern Topics in Informatic^^xsd:string
Bachelor of Science (BSc) in Informatics Engineering of Department ...		Business Process ManagementInformation Systems and Workflows^
Bachelor of Science (BSc) in Informatics Engineering of Department ...		Machine Learning and Data Mining^^xsd:string
Bachelor of Science (BSc) in Informatics Engineering of Department ...		Programming Languages^^xsd:string

6 results

Comment: Results show that professor `Papadakis` is supervisor in 6 courses in 2 study programs. The answer includes the properties course's code, course's name and program's name, with 3 courses listed without `code`.

(B)

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?PF rdf:type ccso:Professor .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
       ?PF ccso:supervisorOf ?SL .
       ?SL ccso:syllabusOf ?CS .
  OPTIONAL { ?CS ccso:code ?CScode } .
       ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
  OPTIONAL { ?CS ccso:csName ?CSname } .
  FILTER regex(str(?PFsurname), "Akoumianakis" ) .  }
  ORDER BY ?CScode
```

PSname	CScode	CSname
"Bachelor of Science (BSc) in Informatic…"		"Databases"
"Bachelor of Science (BSc) in Informatic…"		"Advanced Topics in Databases"
"Bachelor of Science (BSc) in Informatic…"		"Designing the User Interface"
"Bachelor of Science (BSc) in Informatic…"		"Digital Technology and Applications in Engineering"
"Master of Science in Informatics & Mult…"	"IM208/IM_MSc/IED/STEF/TEIC"	"Computer-Supported Collaboration"

Comment: Results show that professor `Akoumianakis` is supervisor in 5 courses from two study programs. The answer includes the properties course's code, course's name and program's name with only one course to have a `code`.

9. Which Courses does an Assistant assist in?

(A)

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?CA rdf:type ccso:Assistant .
       ?CA ccso:assistsInCourse ?SL  .
       ?CA foaf:lastName ?CAsurname .
       ?SL ccso:syllabusOf ?CS .
       ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
       ?CS ccso:csName ?CSname .
   OPTIONAL { ?CS ccso:code ?CScode } .
   FILTER regex(str(?CAsurname), "Kondylakis" ) .        }
  ORDER BY ?CS
```

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?CA rdf:type ccso:Assistant .
          ?CA ccso:assistsInCourse ?SL  .
          ?CA foaf:lastName ?CAsurname .
 OPTIONAL { ?SL ccso:syllabusOf ?CS } .
          ?CS ccso:includedIn ?PS .
 OPTIONAL { ?PS ccso:psName ?PSname } .
          ?CS ccso:csName ?CSname .
  OPTIONAL { ?CS ccso:code ?CScode } .
  FILTER regex(str(?CAsurname), "Kondylakis" ) .        }
 ORDER BY ?CS
```

?PSname	?CScode	?CSname
Bachelor of Science (BSc) in Computer Science of Computers Sciences Depa...	CS150/CS_BSc/CSD/SSE/UoC^^xsd:string	Programming^^xsd:string
Bachelor of Science (BSc) in Computer Science of Computers Sciences Depa...	CS335/CS_BSc/CSD/SSE/UoC^^xsd:string	Computer Networks^^xsd:string
Bachelor of Science (BSc) in Computer Science of Computers Sciences Depa...		Languages and Compilers^^xsd:string
Bachelor of Science (BSc) in Computer Science of Computers Sciences Depa...	CS345/CS_BSc/CSD/SSE/UoC^^xsd:string	Operating Systems^^xsd:string
Bachelor of Science (BSc) in Computer Science of Computers Sciences Depa...		Files and Databases^^xsd:string
Master of Science in Informatics & Multimedia of Department of Informatics...	IM104/IM_MSc/IED/STEF/TEIC^^xsd:string	Advanced Software Engineering^^xsd:string
Master of Science in Informatics & Multimedia of Department of Informatics...	IM105/IM_MSc/IED/STEF/TEIC^^xsd:string	Semantic Web^^xsd:string
Master of Science in Informatics & Multimedia of Department of Informatics...	IM202/IM_MSc/IED/STEF/TEIC^^xsd:string	Wireless Networks^^xsd:string

8 results

Comment: Results show that assistant `Kondylakis` assists in 8 courses in 2 study programs. The answer includes the properties course's code, course's name and program's name and 3 programs do not have code.

(B)

```
SELECT ?PSname ?CScode ?CSname
  WHERE { ?CA rdf:type ccso:Assistant .
          ?CA ccso:assistsInCourse ?SL  .
          ?CA foaf:lastName ?CAsurname .
          ?SL ccso:syllabusOf ?CS .
          ?CS ccso:includedIn ?PS .
  OPTIONAL { ?PS ccso:psName ?PSname } .
          ?CS ccso:csName ?CSname .
   OPTIONAL { ?CS ccso:code ?CScode } .
   FILTER regex(str(?CAsurname), "Katis" ) .        }
  ORDER BY ?CS
```

PSname	CScode	CSname
"Master of Science in Informatics & Mult…"	"IM102/IM_MSc/IED/STEF/TEIC"	"Computer Networks"
"Master of Science in Informatics & Mult…"	"IM104/IM_MSc/IED/STEF/TEIC"	"Advanced Software Engineering"
"Master of Science in Informatics & Mult…"	"IM105/IM_MSc/IED/STEF/TEIC	"Semantic Web"
"Master of Science in Informatics & Mult…"	"IM201/IM_MSc/IED/STEF/TEIC"	"Data Structures and Algorithms"
"Master of Science in Informatics & Mult…"	"IM209/IM_MSc/IED/STEF/TEIC"	"Information and Communication Technologies (ICT) and Education"

Comment: Results show that assistant `Katis` assists in 5 courses, all from the same study program. The answer includes the properties course's code, course's name and program's name.

10. Which Publications has a Person authored?

(A)

```
SELECT ?Publabel ?Pubyear ?Pubtitle
  WHERE { ?Rsc rdf:type ccso:Resource .
          ?Rsc dc:creator ?PR  .
  OPTIONAL { ?Rsc rdfs:label ?Publabel } .
  OPTIONAL { ?Rsc dc:title ?Pubtitle } .
  OPTIONAL { ?Rsc dc:date ?Pubyear }
  OPTIONAL { ?PR foaf:lastName ?PRsurname } .
  OPTIONAL { ?PR foaf:firstName ?PRname } .
  FILTER regex(str(?PRsurname), "Vassilakis" )     }
  ORDER BY ?Pubyear
```

```
SELECT ?Publabel ?Pubyear ?Pubtitle
  WHERE { ?Rsc rdf:type ccso:Resource .
            ?Rsc dc:creator ?PR  .
  OPTIONAL { ?Rsc rdfs:label ?Publabel } .
  OPTIONAL { ?Rsc dc:title ?Pubtitle } .
  OPTIONAL { ?Rsc dc:date ?Pubyear }
  OPTIONAL { ?PR foaf:lastName ?PRsurname } .
  OPTIONAL { ?PR foaf:firstName ?PRname } .
  FILTER regex(str(?PRsurname), "Vassilakis" ) .  }
  ORDER BY ?Pubyear
```

?Publabel	?Pubyear	?Pubtitle
Marakakis, Vassilakis (2005) Expert System for Epilepsy with Uncertainty@en	2005^^...	Expert System for Epilepsy with Uncertainty^^xsd:string
Vassilakis,Psarros,Kalogiannakis (2005) Asynchronous Tele-Teaching at TEI of Crete. Primary r...	2005^^...	Asynchronous Tele-Teaching at TEI of Crete. Primary results of an empirical research^
Kalogiannakis,Vassilakis,Alafodimos,Papadakis,Papachristos,Zafeiri (2009) Adult Education an...	2009^^...	Adult Education and Lifelong Learning. The case of GSAE (General Secretary for Adult
Papachristou, Kalogiannakis, Vassilakis (2010) An Educational Model for Asynchronous E-Lear...	2010^^...	An Educational Model for Asynchronous E-Learning. A Case Study in a Higher Techno

4 results

Comment: Results show that person "Vassilakis" has authored 4 publications recorded with the label, *year*, *publication* and *title* of publication listed and ordered by *Publication year*.

(B)

```
SELECT ?Publabel ?Pubyear ?Pubtitle
  WHERE { ?Rsc rdf:type ccso:Resource .
          ?Rsc dc:creator ?PR  .
  OPTIONAL { ?Rsc rdfs:label ?Publabel } .
  OPTIONAL { ?Rsc dc:title ?Pubtitle } .
  OPTIONAL { ?Rsc dc:date ?Pubyear }
  OPTIONAL { ?PR foaf:lastName ?PRsurname } .
  OPTIONAL { ?PR foaf:firstName ?PRname } .
  FILTER regex(str(?PRsurname), "Papadakis" )     }
  ORDER BY ?Pubyear
```

Publabel	Pubyear	Pubtitle
"Marakakis,Kondylakis,Papadakis (2012) Knowledge Representation in a proof checker for logic …"	"2012"	"Knowledge Representation in a proof checker for logic programs"
"Hatzivasilis,Papaefstathiou,Manifavas,Papadakis (2014) A Reasoning System for composition …"	"2014"	"A Reasoning System for composition verification and security validation"

Comment: Results show that person "Papadakis" has authored 2 publications recorded with the label, *year*, *publication* and *title* of publication listed and ordered by *Publication year*.

11. Which Courses does a Student attend?

(A)

```
SELECT ?CS ?CSname
   WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
        ?Std ccso:attendsCourse ?CS .
        ?Std rdf:type ccso:Student .
   FILTER regex(str(?Std), "Starkeiou_Vakis") .     }
   ORDER BY ?CS
```

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
 OPTIONAL { ?CS ccso:csName ?CSname } .
          ?Std ccso:attendsCourse ?CS .
          ?Std rdf:type ccso:Student .
  FILTER regex(str(?Std), "Starkeiou_Vakis") .  }
  ORDER BY ?CS
```

?CS	?CSname
<http://xworks.gr/ontologies/ccso#CS109/CS_BSc/CSD/SSE/UoC>	English II^^xsd:string
<http://xworks.gr/ontologies/ccso#CS118/CS_BSc/CSD/SSE/UoC>	Discrete Mathematics^^xsd:string
<http://xworks.gr/ontologies/ccso#CS120/CS_BSc/CSD/SSE/UoC>	Digital Design^^xsd:string
<http://xworks.gr/ontologies/ccso#CS180/CS_BSc/CSD/SSE/UoC>	Logic^^xsd:string

4 results

Comment: Results show that student `Starkeiou_Vakis` attends in 4 courses with their IRIs and names listed.

(B)

```
SELECT ?Stdsurname ?Stdname ?CS ?CSname
   WHERE { ?CS rdf:type ccso:Course .
      ?CS ccso:csName ?CSname .
      ?Std ccso:attendsCourse ?CS .
      ?Std rdf:type ccso:Student .
      ?Std foaf:lastName ?Stdsurname .
      ?Std foaf:firstName ?Stdname .
   FILTER regex(str(?Stdsurname), "Starkeiou") .     }
   ORDER BY ?Stdname
```

Stdsurname	Stdname	CS	CSname
"Starkeiou"	"Aria"	http://xworks.gr/ontologies/ccso#IM209/IM_MSc/IED/STEF/TEIC	"Information and Communication Technologies (ICT)"
"Starkeiou"	"Aria"	http://xworks.gr/ontologies/ccso#IM201/IM_MSc/IED/STEF/TEIC	"Data Structures and Algorithms"
"Starkeiou"	"Aria"	http://xworks.gr/ontologies/ccso#IM205/IM_MSc/IED/STEF/TEIC	"Network Security"
"Starkeiou"	"Nick"	http://xworks.gr/ontologies/ccso#CS109/CS_BSc/CSD/SSE/UoC	"English II"
"Starkeiou"	"Nick"	http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"
"Starkeiou"	"Nick"	http://xworks.gr/ontologies/ccso#CS110/CS_BSc/CSD/SSE/UoC	"Calculus I"

Comment: In this query we see the courses that "Starkeiou" attend in with course's IRI and course's name listed. We notice that they are 6 courses total, `Starkeiou_Aria` attends in 3 courses (point 1) and `Starkeiou_Nick` attends in 3 courses (point 2).

12. Which Courses has a Student successfully completed?

(A)

```
SELECT ?CS ?CSname
   WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
      ?Std ccso:hasCompleted ?CS .
      ?Std rdf:type ccso:Student .
   FILTER regex(str(?Std), "Starkeiou_Vakis") .     }
   ORDER BY ?CS
```

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
 OPTIONAL { ?CS ccso:csName ?CSname } .
          ?Std ccso:hasCompleted ?CS .
          ?Std rdf:type ccso:Student .
  FILTER regex(str(?Std), "Starkeiou_Vakis") .   }
  ORDER BY ?CS
```

?CS	?CSname
<http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/UoC>	Introduction to Computer Science^^xsd:string
<http://xworks.gr/ontologies/ccso#CS108/CS_BSc/CSD/SSE/UoC>	English I^^xsd:string
<http://xworks.gr/ontologies/ccso#CS110/CS_BSc/CSD/SSE/UoC>	Calculus I^^xsd:string
<http://xworks.gr/ontologies/ccso#CS119/CS_BSc/CSD/SSE/UoC>	Linear Algebra^^xsd:string
<http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC>	Programming^^xsd:string

5 results

Comment: Results show that student `Starkeiou_Vakis` has successfully completed 5 courses with their IRIs and names listed.

(B)

```
SELECT ?CS ?CSname
   WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
      ?Std ccso:hasCompleted ?CS .
      ?Std rdf:type ccso:Student .
```

```
        FILTER regex(str(?Std), "Vassileiou_Marcela") .     }
        ORDER BY ?CS
```

CS	CSname
http://xworks.gr/ontologies/ccso#TP10K1/IE_BSc/IED/STEF/TEIC	"Physics"
http://xworks.gr/ontologies/ccso#TP10K2/IE_BSc/IED/STEF/TEIC	"Modern Topics in Informatic"
http://xworks.gr/ontologies/ccso#TP10K3/IE_BSc/IED/STEF/TEIC	"Digital Design"
http://xworks.gr/ontologies/ccso#TP10K4/IE_BSc/IED/STEF/TEIC	"Programming"
http://xworks.gr/ontologies/ccso#TP10K5/IE_BSc/IED/STEF/TEIC	"Calculus"
http://xworks.gr/ontologies/ccso#TP10K6/IE_BSc/IED/STEF/TEIC	"Education"
http://xworks.gr/ontologies/ccso#TP20K1/IE_BSc/IED/STEF/TEIC	"Microelectronics"
http://xworks.gr/ontologies/ccso#TP20K2/IE_BSc/IED/STEF/TEIC	"Computer Architecture"
http://xworks.gr/ontologies/ccso#TP20K3/IE_BSc/IED/STEF/TEIC	"Introduction to Telecommunications"
http://xworks.gr/ontologies/ccso#TP20K4/IE_BSc/IED/STEF/TEIC	"Data Structures"
http://xworks.gr/ontologies/ccso#TP20K6/IE_BSc/IED/STEF/TEIC	"Innovation, the Internet and Entrepreneurship"
http://xworks.gr/ontologies/ccso#TP20K7/IE_BSc/IED/STEF/TEIC	"Discrete Mathematics"

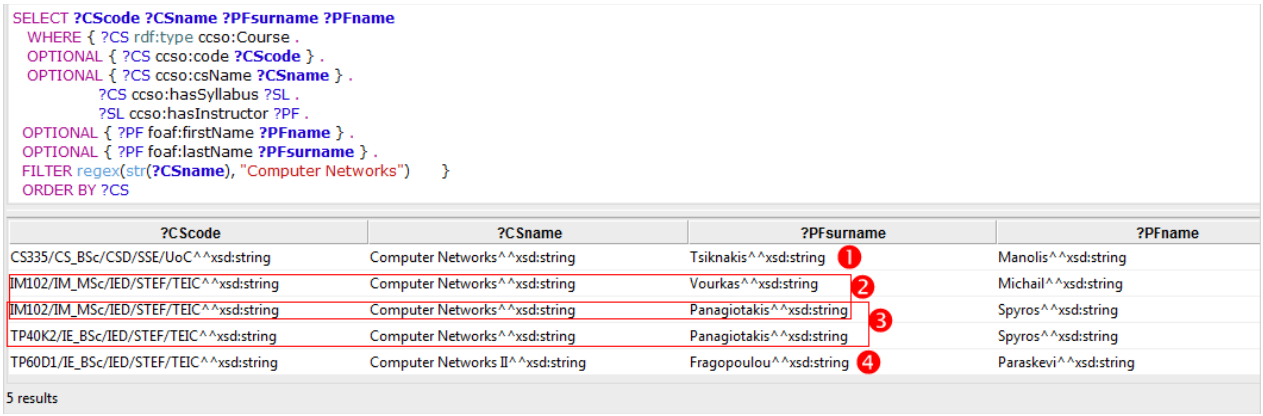
Comment: Results show that student `Vassileiou_Marcela` has successfully completed 12 courses with their IRIs and names listed.

Questions related to Course and Syllabus

13. Who are the Instructors of a Course?

(A)

```
SELECT ?CScode ?CSname ?PFsurname ?PFname
   WHERE { ?CS rdf:type ccso:Course .
   OPTIONAL { ?CS ccso:code ?CScode } .
   OPTIONAL { ?CS ccso:csName ?CSname } .
      ?CS ccso:hasSyllabus ?SL .
      ?SL ccso:hasInstructor ?PF .
  OPTIONAL { ?PF foaf:firstName ?PFname } .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
  FILTER regex(str(?CSname), "Computer Networks")        }
  ORDER BY ?CS
```



?CScode	?CSname	?PFsurname	?PFname
CS335/CS_BSc/CSD/SSE/UoC^^xsd:string	Computer Networks^^xsd:string	Tsiknakis^^xsd:string ❶	Manolis^^xsd:string
IM102/IM_MSc/IED/STEF/TEIC^^xsd:string	Computer Networks^^xsd:string	Vourkas^^xsd:string ❷	Michail^^xsd:string
IM102/IM_MSc/IED/STEF/TEIC^^xsd:string	Computer Networks^^xsd:string	Panagiotakis^^xsd:string ❸	Spyros^^xsd:string
TP40K2/IE_BSc/IED/STEF/TEIC^^xsd:string	Computer Networks^^xsd:string	Panagiotakis^^xsd:string	Spyros^^xsd:string
TP60D1/IE_BSc/IED/STEF/TEIC^^xsd:string	Computer Networks II^^xsd:string	Fragopoulou^^xsd:string ❹	Paraskevi^^xsd:string

5 results

Comment: In this query all courses with with term '`Computer Networks`' in their name are listed. The query returns 5 results with 4 distinct Courses, i.e. `CS335` with professor `Tsiknakis` as instructor (point 1), `IM102` with professors `Vourkas` and `Panagiotakis` as instructors (point 2), `TP40K2` with professor `Panagiotakis` as instructor (point 3) and `TP60D1` with

professor Fraggopoulou as instructor (point 4).

(B)

```
SELECT ?CScode ?CSname ?PFsurname ?PFname
   WHERE { ?CS rdf:type ccso:Course .
   OPTIONAL { ?CS ccso:code ?CScode } .
   OPTIONAL { ?CS ccso:csName ?CSname } .
      ?CS ccso:hasSyllabus ?SL .
      ?SL ccso:hasInstructor ?PF .
  OPTIONAL { ?PF foaf:firstName ?PFname } .
  OPTIONAL { ?PF foaf:lastName ?PFsurname } .
  FILTER regex(str(?CScode), "IM209")       }
  ORDER BY ?CS
```

CScode	CSname	PFsurname	PFname
"IM209/IM_MSc/IED/STEF/TEIC"	"Information and Communication Technologies (ICT) and Education"	"Kalogiannakis"	"Michail"
"IM209/IM_MSc/IED/STEF/TEIC"	"Information and Communication Technologies (ICT) and Education"	"Vassilakis"	"Konstantinos"

Comment: This query shows that the course with code 'IM209' has name 'ICT in Education' (CSname) and has 2 Instructors: professor Kalogiannakis and professor Vassilakis.

14. Who are the Assistants of a Course?

(A)

```
SELECT ?CSname ?CAsurname ?CAname
  WHERE { ?CS rdf:type ccso:Course .
 OPTIONAL {  ?CS ccso:csName ?CSname } .
      ?CS ccso:hasSyllabus ?SL .
      ?SL ccso:hasCourseAssistant ?CA .
 OPTIONAL { ?CA foaf:firstName ?CAname } .
 OPTIONAL {  ?CA foaf:lastName ?CAsurname } .
 FILTER regex(str(?CSname), "Semantic Web") }
 ORDER BY ?CAsurname
```

```
SELECT ?CSname ?CAsurname ?CAname
 WHERE { ?CS rdf:type ccso:Course .
OPTIONAL {          ?CS ccso:csName ?CSname } .
         ?CS ccso:hasSyllabus ?SL .
         ?SL ccso:hasCourseAssistant ?CA .
OPTIONAL { ?CA foaf:firstName ?CAname } .
OPTIONAL {          ?CA foaf:lastName ?CAsurname } .
FILTER regex(str(?CSname), "Semantic Web") }
ORDER BY ?CAsurname
```

?CSname	?CAsurname	?CAname
Semantic Web^^xsd:string	Katis^^xsd:string	Evangelos^^xsd:anyURI
Semantic Web^^xsd:string	Kondylakis^^xsd:string	Haridimos^^xsd:string

2 results

Comment: Results show that the course with name 'Semantic Web' has 2 Course Assistants: Katis and Kondylakis.

(B)

```
SELECT ?CScode ?CSname CAsurname ?CAname
   WHERE { ?CS rdf:type ccso:Course .
   OPTIONAL { ?CS ccso:code ?CScode } .
   OPTIONAL { ?CS ccso:csName ?CSname } .
      ?CS ccso:hasSyllabus ?SL .
      ?SL ccso:hasCourseAssistant ?CA .
  OPTIONAL { ?CA foaf:firstName ?CAname } .
  OPTIONAL { ?CA foaf:lastName ?CAsurname } .
```

```
        FILTER regex(str(?CScode), "CS119")        }
        ORDER BY ?CAsurname
```

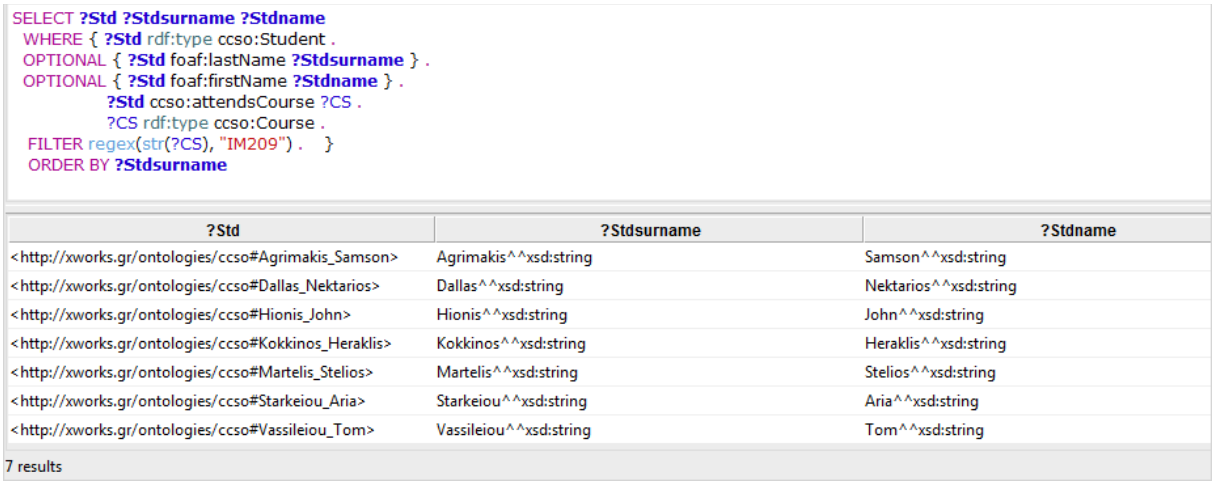
CScode	CSname	CAsurname	CAname
"CS119/CS_BSc/CSD/SSE/UoC"	"Linear Algebra"	"Gramma"	"Joanna"
"CS119/CS_BSc/CSD/SSE/UoC"	"Linear Algebra"	"Pappa"	"Pepi"

Comment: Results show that course 'CS119' has name 'Linear Algebra' and has 2 Course Assistants: Gramma and Pappa.

15. Who are the Students attend a Course?

(A)

```
SELECT ?Std ?Stdsurname ?Stdname
  WHERE { ?Std rdf:type ccso:Student .
  OPTIONAL { ?Std foaf:lastName ?Stdsurname } .
  OPTIONAL { ?Std foaf:firstName ?Stdname } .
       ?Std ccso:attendsCourse ?CS .
       ?CS rdf:type ccso:Course .
   FILTER regex(str(?CS), "IM209") .     }
   ORDER BY ?Stdsurname
```



?Std	?Stdsurname	?Stdname
<http://xworks.gr/ontologies/ccso#Agrimakis_Samson>	Agrimakis^^xsd:string	Samson^^xsd:string
<http://xworks.gr/ontologies/ccso#Dallas_Nektarios>	Dallas^^xsd:string	Nektarios^^xsd:string
<http://xworks.gr/ontologies/ccso#Hionis_John>	Hionis^^xsd:string	John^^xsd:string
<http://xworks.gr/ontologies/ccso#Kokkinos_Heraklis>	Kokkinos^^xsd:string	Heraklis^^xsd:string
<http://xworks.gr/ontologies/ccso#Martelis_Stelios>	Martelis^^xsd:string	Stelios^^xsd:string
<http://xworks.gr/ontologies/ccso#Starkeiou_Aria>	Starkeiou^^xsd:string	Aria^^xsd:string
<http://xworks.gr/ontologies/ccso#Vassileiou_Tom>	Vassileiou^^xsd:string	Tom^^xsd:string

7 results

Comment: Results show that course 'IM209' has 7 Attendees with Attender's IRI, name and surname listed.

(B)

```
SELECT ?Std ?Stdsurname ?Stdname ?CSname
  WHERE { ?Std rdf:type ccso:Student .
  OPTIONAL { ?Std foaf:lastName ?Stdsurname } .
  OPTIONAL { ?Std foaf:firstName ?Stdname } .
       ?Std ccso:attendsCourse ?CS .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS rdf:type ccso:Course .
   FILTER regex(str(?CS), "PSY1501") .     }
   ORDER BY ?Stdsurname
```

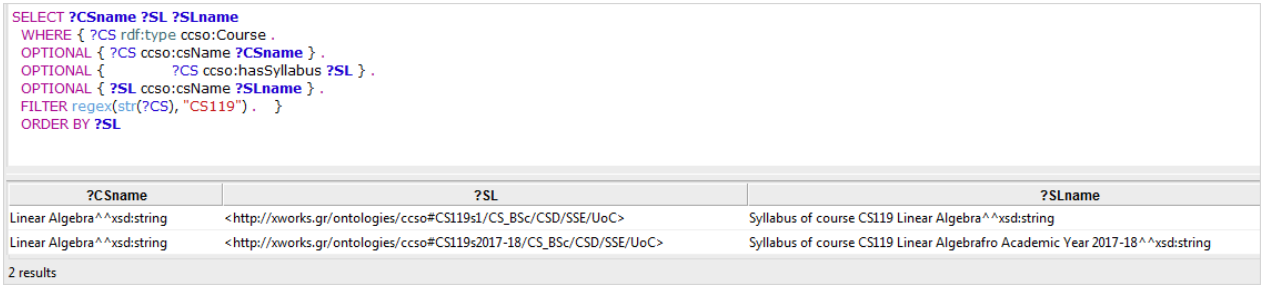
Std	Stdsurname	Stdname	CSname
http://xworks.gr/ontologies/ccso#Kekili_Souzan	"Kekili"	"Souzan"	"Social Psychology I: Introduction to Social Psychology"
http://xworks.gr/ontologies/ccso#Ralli_Mary	"Ralli"	"Mary"	"Social Psychology I: Introduction to Social Psychology"
http://xworks.gr/ontologies/ccso#Rallis_John	"Rallis"	"John"	"Social Psychology I: Introduction to Social Psychology"

Comment: Results show that course 'PSY1501' has name 'Social Psychology I' and has 3 Attendees with Attender's name & surname listed.

16. Which are the Syllabi of a Course?

(A)

```
SELECT ?CSname ?SL ?SLname
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
  OPTIONAL { ?CS ccso:hasSyllabus ?SL } .
  OPTIONAL { ?SL ccso:csName ?SLname } .
  FILTER regex(str(?CS), "CS119") .    }
  ORDER BY ?SL
```



?CSname	?SL	?SLname
Linear Algebra^^xsd:string	<http://xworks.gr/ontologies/ccso#CS119s1/CS_BSc/CSD/SSE/UoC>	Syllabus of course CS119 Linear Algebra^^xsd:string
Linear Algebra^^xsd:string	<http://xworks.gr/ontologies/ccso#CS119s2017-18/CS_BSc/CSD/SSE/UoC>	Syllabus of course CS119 Linear Algebrafro Academic Year 2017-18^^xsd:string

2 results

Comment: Results show that course `CS119` has name '`Linear Algebra`' and has 2 Syllabi: `CS119s1` and `CS119s2017-18`.

(B)

```
SELECT ?CSname ?SL ?SLname ?SLAcademicYear
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
      ?CS ccso:hasSyllabus ?SL .
  OPTIONAL { ?SL ccso:csName ?SLname } .
  OPTIONAL { ?SL ccso:academicYear ?SLAcademicYear } .
  FILTER regex(str(?CS), "TP10K1") .    }
  ORDER BY ?SL
```

CSname	SL	SLname	SLAcademicYear
"Physics"	http://xworks.gr/ontologies/ccso#TP10K1s2015-16/IE_BSc/IED/STEF/TEIC	"Syllabus of course TP10K1 Physics"	"2015-16"
"Physics"	http://xworks.gr/ontologies/ccso#TP10K1s2016-17/IE_BSc/IED/STEF/TEIC	"Syllabus of course TP10K1 Physics"	"2016-17"
"Physics"	http://xworks.gr/ontologies/ccso#TP10K1s2017-18/IE_BSc/IED/STEF/TEIC	"Syllabus of course TP10K1 Physics"	"2017-18"

Comment: Results show that course `TP10K1` has name '`Physics`' has 3 Syllabi: `TP10K1s2015-16`, `TP10K1s2016-17` & `TP10K1s2017-18` and the academic year for each one.

17. Which are the Topics and Learning Outcomes of a Course?

It's more convenient to split this query in two, one query for Topics and one query for Learning Outcomes.

(A)

```
SELECT ?Tlabel ?CS
   WHERE { ?CS rdf:type ccso:Course .
      ?CS ccso:hasSyllabus ?Syl .
      ?Syl ccso:coversTopic ?Topic .
   OPTIONAL { ?Topic ccso:label ?Tlabel } .
   FILTER regex(str(?CS), "CS335/CS_BSc/CSD/SSE/UoC")   }
   ORDER BY ?Topic
```

```
SELECT ?Tlabel ?CS
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:hasSyllabus ?Syl .
          ?Syl ccso:coversTopic ?Topic .
  OPTIONAL { ?Topic ccso:label ?Tlabel } .
  FILTER regex(str(?CS), "CS335") }
  ORDER BY ?Topic
```

?Tlabel	?CS
Common approaches to multiple access (exponential-backoff, time division multiplexing, etc)^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Ethernet^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Local Area Networks^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Multiple Access Problem^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Switching^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Internet Protocol (IP)^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Routing versus forwarding^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Scalability issues (hierarchical addressing)^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Static routing^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>

9 results

Comment: Results show that the course CS335 has 9 Topics with their labels listed.

(B)

```
SELECT ?LOlabel ?CS
    WHERE { ?CS rdf:type ccso:Course .
      ?CS ccso:hasSyllabus ?SL .
      ?SL ccso:aimsToLO ?LO .
    OPTIONAL { ?LO ccso:label ?LOlabel } .
    FILTER regex(str(?CS), "CS335")  }
    ORDER BY ?LO
```

```
SELECT ?LOlabel ?CS
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:hasSyllabus ?SL .
          ?SL ccso:aimsToLO ?LO .
  OPTIONAL { ?LO ccso:label ?LOlabel } .
  FILTER regex(str(?CS), "CS335") }
  ORDER BY ?LO
```

?LOlabel	?CS
Describe how frames are forwarded in an Ethernet network^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Describe the interrelations between IP and Ethernet^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Describe the steps used in one common approach to the multiple access problem^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Identify the differences between IP and Ethernet^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Describe how packets are forwarded in an IP networks^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
Describe the organization of the network layer^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>
List the scalability benefits of hierarchical addressing^^xsd:string	<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>

7 results

Comment: As to the Learning Outcomes, results show that the same course CS335 has 7 Learning Outcomes with their labels listed.

Questions related to study planning

18. Which Courses are included in a study program?

(A)

```
SELECT ?CS ?CSname
   WHERE { ?CS rdf:type ccso:Course .
      ?CS ccso:csName ?CSname .
      ?PS ccso:includesCourse ?CS .
  FILTER regex(str(?PS), "IE_BSc/IED/STEF/TEIC") .    }
  ORDER BY ?CS
```

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:csName ?CSname .
          ?PS ccso:includesCourse ?CS .
 FILTER regex(str(?PS), "IE_BSc/IED/STEF/TEIC") .   }
 ORDER BY ?CS
```

?CS	?CSname
<http://xworks.gr/ontologies/ccso#TP10K1/IE_BSc/IED/STEF/TEIC>	Physics^^xsd:string
<http://xworks.gr/ontologies/ccso#TP10K2/IE_BSc/IED/STEF/TEIC>	Modern Topics in Informatic^^xsd:string
<http://xworks.gr/ontologies/ccso#TP10K3/IE_BSc/IED/STEF/TEIC>	Digital Design^^xsd:string
<http://xworks.gr/ontologies/ccso#TP10K4/IE_BSc/IED/STEF/TEIC>	Programming^^xsd:string
<http://xworks.gr/ontologies/ccso#TP10K5/IE_BSc/IED/STEF/TEIC>	Calculus^^xsd:string
<http://xworks.gr/ontologies/ccso#TP10K6/IE_BSc/IED/STEF/TEIC>	Education^^xsd:string
<http://xworks.gr/ontologies/ccso#TP20K1/IE_BSc/IED/STEF/TEIC>	Microelectronics^^xsd:string
<http://xworks.gr/ontologies/ccso#TP20K2/IE_BSc/IED/STEF/TEIC>	Computer Architecture^^xsd:string
<http://xworks.gr/ontologies/ccso#TP20K3/IE_BSc/IED/STEF/TEIC>	Introduction to Telecommunications^^xsd:string

64 results

Comment: Results show that program `IE_BSc/IED/STEF/TEIC` includes 64 courses with course's IRI and course's name listed.

(B)

```
SELECT ?CS ?CSname
   WHERE { ?CS rdf:type ccso:Course .
           ?CS ccso:csName ?CSname .
           ?PS ccso:includesCourse ?CS .
  FILTER regex(str(?PS), "CS_BSc/CSD/SSE/UoC") .     }
  ORDER BY ?CS
```

CS	CSname
http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/UoC	"Introduction to Computer Science"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS108/CS_BSc/CSD/SSE/UoC	"English I"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS109/CS_BSc/CSD/SSE/UoC	"English II"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS110/CS_BSc/CSD/SSE/UoC	"Calculus I"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS118/CS_BSc/CSD/SSE/UoC	"Discrete Mathematics"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS119/CS_BSc/CSD/SSE/UoC	"Linear Algebra"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS120/CS_BSc/CSD/SSE/UoC	"Digital Design"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS180/CS_BSc/CSD/SSE/UoC	"Logic"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS208/CS_BSc/CSD/SSE/UoC	"English III"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS209/CS_BSc/CSD/SSE/UoC	"English IV"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS215/CS_BSc/CSD/SSE/UoC	"Applied Mathematics for Engineers"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS217/CS_BSc/CSD/SSE/UoC	"Probability"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS220/CS_BSc/CSD/SSE/UoC	"Digital Circuits Lab"^^<http://www.w3.org/2001/XMLSchema#string>

Comment: Results show that study program `CS_BSc/CSD/SSE/UoC` includes 60 courses with course's IRI and course's name listed.

19. Which are the core Courses in a study program?

(A)

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?PS ccso:includesCourse ?CS .
       ?CS ccso:courseType ?CStype .
 FILTER (regex(str(?PS), "IE_BSc/IED/STEF/TEIC") && regex(str(?CStype),
"Core")) .      }
  ORDER BY ?CS
```

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
            ?PS ccso:includesCourse ?CS .
            ?CS ccso:courseType ?CStype .
  FILTER (regex(str(?PS), "IE_BSc/IED/STEF/TEIC") && regex(str(?CStype), "Core")) .    }
  ORDER BY ?CS
```

?CS	?CSname
<http://xworks.gr/ontologies/ccso#TP30K5/IE_BSc/IED/STEF/TEIC>	Applied Mathematics^^xsd:string
<http://xworks.gr/ontologies/ccso#TP30K6/IE_BSc/IED/STEF/TEIC>	Terminology and Technical Writing^^xsd:string
<http://xworks.gr/ontologies/ccso#TP30K7/IE_BSc/IED/STEF/TEIC>	Multimedia Technology^^xsd:string
<http://xworks.gr/ontologies/ccso#TP40K1/IE_BSc/IED/STEF/TEIC>	Digital Signal Processing^^xsd:string
<http://xworks.gr/ontologies/ccso#TP40K2/IE_BSc/IED/STEF/TEIC>	Computer Networks^^xsd:string
<http://xworks.gr/ontologies/ccso#TP40K3/IE_BSc/IED/STEF/TEIC>	Internet Programming^^xsd:string
<http://xworks.gr/ontologies/ccso#TP40K4/IE_BSc/IED/STEF/TEIC>	Principles of Software Engineering^^xsd:string
<http://xworks.gr/ontologies/ccso#TP40K5/IE_BSc/IED/STEF/TEIC>	Linear Algebra^^xsd:string
<http://xworks.gr/ontologies/ccso#TP40K6/IE_BSc/IED/STEF/TEIC>	Teaching Informatics^^xsd:string

48 results

Comment: in this query the results are being filtered and list the courses included in program `IE_BSc/IED/STEF/TEIC` which have as `courseType` the value '`Core`.'.
Results show that program `IE_BSc/IED/STEF/TEIC` includes 48 Core Courses, (of the total of 64 Courses as shown in *Query 19A*) with course's IRI and course's name listed.

(B)

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?PS ccso:includesCourse ?CS .
       ?CS ccso:courseType ?CStype .
  FILTER (regex(str(?PS), "CS_BSc/CSD/SSE/UoC") && regex(str(?CStype),
"Core")) .      }
  ORDER BY ?CS
```

CS	CSname
http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/UoC	"Introduction to Computer Science"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS108/CS_BSc/CSD/SSE/UoC	"English I"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS109/CS_BSc/CSD/SSE/UoC	"English II"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS110/CS_BSc/CSD/SSE/UoC	"Calculus I"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS118/CS_BSc/CSD/SSE/UoC	"Discrete Mathematics"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS119/CS_BSc/CSD/SSE/UoC	"Linear Algebra"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS120/CS_BSc/CSD/SSE/UoC	"Digital Design"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS180/CS_BSc/CSD/SSE/UoC	"Logic"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS208/CS_BSc/CSD/SSE/UoC	"English III"^^<http://www.w3.org/2001/XMLSchema#string>
http://xworks.gr/ontologies/ccso#CS209/CS_BSc/CSD/SSE/UoC	"English IV"^^<http://www.w3.org/2001/XMLSchema#string>

Comment: Results show that program `CS_BSc/CSD/SSE/UoC` includes 24 Core Courses, with course's IRI and course's name listed, in a total of 60 Courses in the program (*Query 19B*).

20. Which are the prerequisites of a Course?

(A)

```
SELECT ?CS ?CSprerequisite ?CSprerequisitename
   WHERE { ?CS rdf:type ccso:Course .
       ?CS ccso:hasPrerequisite ?CSprerequisite .
  OPTIONAL { ?CSprerequisite ccso:csName ?CSprerequisitename } .
  FILTER regex(str(?CS), "TP10K1/IE_BSc/IED/STEF/TEIC") .      }
```

```
SELECT ?CS ?CSprerequisite ?CSprerequisitename
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:hasPrerequisite ?CSprerequisite .
  OPTIONAL { ?CSprerequisite ccso:csName ?CSprerequisitename } .
  FILTER regex(str(?CS), "TP10K1/IE_BSc/IED/STEF/TEIC") .    }
```

?CS	?CSprerequisite	?CSprerequisitename

0 results

Comment: In this query the course `TP10K1/IE_BSc/IED/STEF/TEIC` is tested. The query returns 0 results which mean that course `TP10K1` does not have any prerequisites.

(B)

```
SELECT ?CS ?CSprerequisite ?CSprerequisitename
   WHERE { ?CS rdf:type ccso:Course .
       ?CS ccso:hasPrerequisite ?CSprerequisite .
  OPTIONAL { ?CSprerequisite ccso:csName ?CSprerequisitename } .
  FILTER regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") .      }
```

CS	CSprerequisite	CSprerequisitename
http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC	http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/UoC	"Introduction to Computer Science"^
http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC	http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"^

Comment: In this query the course `CS240/CS_BSc/CSD/SSE/UoC` is tested. Results show that course `CS240` has 2 prerequisites: `CS100-Introduction to Computer Science` & `CS150-Programming`.

21. Which Courses have prerequisites (and what are they) in a study program?

(A)

```
SELECT ?CS ?CSname ?CSprerequisite ?CSprerequisitename
   WHERE { ?CS rdf:type ccso:Course .
       ?CS ccso:csName ?CSname .
       ?CS ccso:hasPrerequisite ?CSprerequisite .
       ?CSprerequisite ccso:csName ?CSprerequisitename .
       ?CS ccso:includedIn ?PS .
  FILTER regex(str(?PS), "CS_BSc/CSD/SSE/UoC") .     }
  ORDER BY ?CS
```

```
SELECT ?CS ?CSname ?CSprerequisite ?CSprerequisitename
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:csName ?CSname .
          ?CS ccso:hasPrerequisite ?CSprerequisite .
          ?CSprerequisite ccso:csName ?CSprerequisitename .
          ?CS ccso:includedIn ?PS .
  FILTER regex(str(?PS), "CS_BSc/CSD/SSE/UoC") .    }
  ORDER BY ?CS
```

?CS	?CSname	?CSprerequisite	?CSprerequisitename
<http://xworks.gr/ontologies/ccso#CS208/CS_BSc/CSD/SSE/U...	English III^^xsd:string	<http://xworks.gr/ontologies/ccso#CS109/CS_BSc/CSD/SSE/U...	English II^^xsd:string
<http://xworks.gr/ontologies/ccso#CS209/CS_BSc/CSD/SSE/U...	English IV^^xsd:string	<http://xworks.gr/ontologies/ccso#CS208/CS_BSc/CSD/SSE/U...	English III^^xsd:string
<http://xworks.gr/ontologies/ccso#CS215/CS_BSc/CSD/SSE/U...	Applied Mathematics for Engineers^^xsd:string	<http://xworks.gr/ontologies/ccso#CS110/CS_BSc/CSD/SSE/U...	Calculus I^^xsd:string
<http://xworks.gr/ontologies/ccso#CS217/CS_BSc/CSD/SSE/U...	Probability^^xsd:string	<http://xworks.gr/ontologies/ccso#CS110/CS_BSc/CSD/SSE/U...	Calculus I^^xsd:string
<http://xworks.gr/ontologies/ccso#CS220/CS_BSc/CSD/SSE/U...	Digital Circuits Lab^^xsd:string	<http://xworks.gr/ontologies/ccso#CS120/CS_BSc/CSD/SSE/U...	Digital Design^^xsd:string
<http://xworks.gr/ontologies/ccso#CS225/CS_BSc/CSD/SSE/U...	Computer Organization^^xsd:string	<http://xworks.gr/ontologies/ccso#CS120/CS_BSc/CSD/SSE/U...	Digital Design^^xsd:string
<http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/U...	Data Structures^^xsd:string	<http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/U...	Introduction to Computer Science^^xsd:string
<http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/U...	Data Structures^^xsd:string	<http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/U...	Programming^^xsd:string
<http://xworks.gr/ontologies/ccso#CS252/CS_BSc/CSD/SSE/U...	Object-Oriented Programming^^xsd:string	<http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/U...	Programming^^xsd:string

60 results

Comment: In this query the study program `BSc_Computer_Science/CS_BSc/CSD/SSE/UoC` is tested. We notice that study program `CS_BSc/CSD/SSE/UoC` has 60 courses that require knowledge from other courses as prerequisites.

For example, some inferences that can be drawn from the results are that course `CS240-Data Structures` has courses `CS100-Introduction to Computer Science` & `CS150-Programming` as prerequisites (Point 1) and course `CS120-Digital Design` is prerequisite in 2 courses: `CS220-Digital Circuits` & `CS225-Computer Organization` (Point 2).

(B)

```
SELECT ?CS ?CSname ?CSprerequisite ?CSprerequisitename
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:csName ?CSname .
          ?CS ccso:hasPrerequisite ?CSprerequisite .
          ?CSprerequisite ccso:csName ?CSprerequisitename .
          ?CS ccso:includedIn ?PS .
  FILTER regex(str(?PS), "IE_BSc/IED/STEF/TEIC") .      }
  ORDER BY ?CS
```

CS	CSname	CSprerequisite	CSprerequisitename
http://.../ccso#TP20K4/IE_BSc/IED/STEF/TEIC	"Data Structures"	http://.../ccso#TP10K4/IE_BSc/IED/STEF/TEIC	"Programming" ❶
http://.../ccso#TP20K7/IE_BSc/IED/STEF/TEIC	"Discrete Mathematics"	http://.../ccso#TP10K5/IE_BSc/IED/STEF/TEIC	"Calculus" ❷
http://.../ccso#TP30K5/IE_BSc/IED/STEF/TEIC	"Applied Mathematics"	http://.../ccso#TP10K5/IE_BSc/IED/STEF/TEIC	"Calculus"
http://.../ccso#TP40K4/IE_BSc/IED/STEF/TEIC	"Principles of Software Engineering"	http://.../ccso#TP10K4/IE_BSc/IED/STEF/TEIC	"Programming"
http://.../ccso#TP40K5/IE_BSc/IED/STEF/TEIC	"Linear Algebra"	http://.../ccso#TP30K5/IE_BSc/IED/STEF/TEIC	"Applied Mathematics"
http://.../ccso#TP50L2/IE_BSc/IED/STEF/TEIC	"Plan-Driven and Agile Programming"	http://.../ccso#TP10K4/IE_BSc/IED/STEF/TEIC	"Programming"
http://.../ccso#TP60D1/IE_BSc/IED/STEF/TEIC	"Computer Networks II"	http://.../ccso#TP40K2/IE_BSc/IED/STEF/TEIC	"Computer Networks"
http://.../ccso#TP60D3/IE_BSc/IED/STEF/TEIC	"Multimedia Application Development"	http://.../ccso#TP40K3/IE_BSc/IED/STEF/TEIC	"Internet Programming"
http://.../ccso#TP60L1/IE_BSc/IED/STEF/TEIC	"Advanced Topics in Databases"	http://.../ccso#TP30K2/IE_BSc/IED/STEF/TEIC	"Databases"
http://.../ccso#TP60L2/IE_BSc/IED/STEF/TEIC	"Artificial Intelligence"	http://.../ccso#TP50L3/IE_BSc/IED/STEF/TEIC	"Logic Programming"
http://.../ccso#TP70L1/IE_BSc/IED/STEF/TEIC	"Knowledge Systems"	http://.../ccso#TP60L2/IE_BSc/IED/STEF/TEIC	"Artificial Intelligence"

Comment: In this query the study program `BSc_Informatics_Engineering/IE_BSc/IED/STEF/TEIC` is tested. The query results 11 courses, in the study program `IE_BSc/IED/STEF/TEIC`, which have prerequisites and the answer includes course's IRI, course's name, prerequisite-course's IRI, and prerequisite-course's name.

For example, some inferences that can be drawn from the results are that course `TP20K4-Data Structures` has course `TP10K4 - Programming` as prerequisite (Point 1) and course `TP10K5-Calculus` is prerequisite in 2 courses, the `TP20K7 - Discrete Mathemetics` & `TP30K5 - Applied Mathematics` (Point 2).

22. Which Courses cover a Knowledge Area?

(A)

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
          ?CS ccso:coversKA ?KA .
          ?KA ccso:fsName ?KAname .
  FILTER regex(str(?KAname), "Networking and Communication") .     }
  ORDER BY ?CS
```

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:csName ?CSname .
          ?CS ccso:coversKA ?KA .
          ?KA ccso:fsName ?KAname .
  FILTER regex(str(?KAname), "Networking and Communication") .   }
   ORDER BY ?CS
```

?CS	?CSname
<http://xworks.gr/ontologies/ccso#CS330/CS_BSc/CSD/SSE/UoC>	Introduction to Telecommunication Theory^^xsd:string
<http://xworks.gr/ontologies/ccso#CS370/CS_BSc/CSD/SSE/UoC>	Digital Signal Processing^^xsd:string
<http://xworks.gr/ontologies/ccso#CS435/CS_BSc/CSD/SSE/UoC>	Network Technology & Programming Lab^^xsd:string
<http://xworks.gr/ontologies/ccso#CS436/CS_BSc/CSD/SSE/UoC>	Software Defined Networks (SDN)^^xsd:string
<http://xworks.gr/ontologies/ccso#CS439/CS_BSc/CSD/SSE/UoC>	Wireless Networks and Moblile Computing^^xsd:string
<http://xworks.gr/ontologies/ccso#CS474/CS_BSc/CSD/SSE/UoC>	Multimedia Technology^^xsd:string
<http://xworks.gr/ontologies/ccso#IM102/IM_MSc/IED/STEF/TEIC>	Computer Networks^^xsd:string
<http://xworks.gr/ontologies/ccso#IM103/IM_MSc/IED/STEF/TEIC>	Advanced Multimedia Technologies^^xsd:string
<http://xworks.gr/ontologies/ccso#IM202/IM_MSc/IED/STEF/TEIC>	Wireless Networks^^xsd:string

27 results

Comment: In this query the knowledge area 'Networking and Communication' is tested. Results show that KA 'Networking and Communication' is covered in 27 Courses that included in different study programs (it is infered from the Course's IRI), with course's IRI and course's name listed.

(B)

```
SELECT ?CS ?CSname
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS ccso:coversKA ?KA .
       ?KA ccso:fsName ?KAname .
  FILTER regex(str(?KAname), "Programming Languages") .     }
  ORDER BY ?CS
```

CS	CSname
http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"
http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC	"Data Structures"
http://xworks.gr/ontologies/ccso#CS252/CS_BSc/CSD/SSE/UoC	"Object-Oriented Programming"
http://xworks.gr/ontologies/ccso#CS317/CS_BSc/CSD/SSE/UoC	"Applied Stochastic Processes"
http://xworks.gr/ontologies/ccso#CS340/CS_BSc/CSD/SSE/UoC	"Languages and Compilers"
http://xworks.gr/ontologies/ccso#CS342/CS_BSc/CSD/SSE/UoC	"Parallel Programming"
http://xworks.gr/ontologies/ccso#CS380/CS_BSc/CSD/SSE/UoC	"Algorithms and Complexity"
http://xworks.gr/ontologies/ccso#CS387/CS_BSc/CSD/SSE/UoC	"Introduction to Artificial Intelligence"
http://xworks.gr/ontologies/ccso#CS486/CS_BSc/CSD/SSE/UoC	"Principles of Distributed Computing"
http://xworks.gr/ontologies/ccso#TP10K4/IE_BSc/IED/STEF/TEIC	"Programming"
http://xworks.gr/ontologies/ccso#TP30K4/IE_BSc/IED/STEF/TEIC	"Algorithms"
http://xworks.gr/ontologies/ccso#TP40K3/IE_BSc/IED/STEF/TEIC	"Internet Programming"

Comment: In this query the knowledge area 'Programming Languages' is tested. Results show that KA 'Programming Languages' is covered in 12 Courses that included in different study programs, with course's IRI and course's name listed..

23. Which Courses cover a Topic?

(A)

```
SELECT ?CS ?CSname ?Topiclabel
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS ccso:hasSyllabus ?SL .
       ?SL ccso:coversTopic ?Topic .
       ?Topic rdf:type ccso:Topic .
   OPTIONAL { ?Topic ccso:label ?Topiclabel } .
   FILTER  (regex(str(?Topiclabel), "Multiple Access") || regex
  (str(?Topiclabel), "multiple access"))   . }
   ORDER BY ?CS
```


```
SELECT ?CS ?CSname ?Topiclabel
 WHERE { ?CS rdf:type ccso:Course .
 OPTIONAL { ?CS ccso:csName ?CSname } .
          ?CS ccso:hasSyllabus ?SL .
          ?SL ccso:coversTopic ?Topic .
          ?Topic rdf:type ccso:Topic .
 OPTIONAL { ?Topic ccso:label ?Topiclabel } .
 FILTER  (regex(str(?Topiclabel), "Multiple Access") || regex(str(?Topiclabel), "multiple access")) . }
 ORDER BY ?CS
```

?CS	?CSname	?Topiclabel
<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>	Computer Networks^^xsd:string	Multiple Access Problem^^xsd:string
<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>	Computer Networks^^xsd:string	Common approaches to multiple access
<http://xworks.gr/ontologies/ccso#CS435/CS_BSc/CSD/SSE/UoC>	Network Technology & Programming Lab^^xsd:string	Multiple Access Problem^^xsd:string
<http://xworks.gr/ontologies/ccso#CS435/CS_BSc/CSD/SSE/UoC>	Network Technology & Programming Lab^^xsd:string	Common approaches to multiple access
<http://xworks.gr/ontologies/ccso#IM102/IM_MSc/IED/STEF/TEIC>	Computer Networks^^xsd:string	Multiple Access Problem^^xsd:string
<http://xworks.gr/ontologies/ccso#IM102/IM_MSc/IED/STEF/TEIC>	Computer Networks^^xsd:string	Common approaches to multiple access
<http://xworks.gr/ontologies/ccso#TP40K2/IE_BSc/IED/STEF/TEIC>	Computer Networks^^xsd:string	Multiple Access Problem^^xsd:string
<http://xworks.gr/ontologies/ccso#TP50D2/IE_BSc/IED/STEF/TEIC>	Wireless Communication TransmissionStructures^^xsd:...	Multiple Access Problem^^xsd:string

9 results

Comment: In this query the topic '`Multiple Access`' is tested, with both versions of topic's name (first letter capital and not), because *Snap SPAQL plugin* is case sensitive. Results show that Topics relevant to '`Multiple Access`' subject are covered in 9 Courses with course's IRI, course's name and Topic's name listed.

(B)

```
SELECT ?CS ?CSname ?Topiclabel
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS ccso:hasSyllabus ?SL .
       ?SL ccso:coversTopic ?Topic .
       ?Topic rdf:type ccso:Topic .
 OPTIONAL { ?Topic ccso:label ?Topiclabel } .
FILTER  (regex(str(?Topiclabel), "inheritance") || regex(str(?Topiclabel),
"Inheritance"))  . }
ORDER BY ?CS
```

CS	CSname	Topiclabel
http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#CS252/CS_BSc/CSD/SSE/UoC	"Object-Oriented Programming"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#CS342/CS_BSc/CSD/SSE/UoC	"Parallel Programming"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#IM201/IM_MSc/IED/STEF/TEIC	"Data Structures and Algorithms"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#TP20K4/IE_BSc/IED/STEF/TEIC	"Data Structures"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#TP50L3/IE_BSc/IED/STEF/TEIC	"Logic Programming"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#TP50L5/IE_BSc/IED/STEF/TEIC	"Programming Languages"	"Subclasses, inheritance, and method overriding"
http://xworks.gr/ontologies/ccso#TP60D5/IE_BSc/IED/STEF/TEIC	"Parallel Processing and Systems"	"Subclasses, inheritance, and method overriding"

Comment: In this query the topic '`inheritance`' is tested, with both versions of topic's name (first letter capital and not), because *Snap SPAQL plugin* is case sensitive. Results show that Topics relevant to '`inheritance`' subject are being covered in 8 Courses in 3 different study programs, with course's IRI, course's name and Topic's name listed.

24. Which Courses aim to a Learning Outcome?

(A)

```
SELECT ?CS ?CSname ?LOlabel
   WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS ccso:hasSyllabus ?SL .
       ?SL ccso:aimsToLO ?LO .
       ?LO rdf:type ccso:LearningOutcome .
  OPTIONAL { ?LO ccso:label ?LOlabel } .
  FILTER  (regex(str(?LOlabel), "multiple access") || regex(str(?LOlabel),
"Multiple Access"))  . }
 ORDER BY ?CS
```

```
SELECT ?CS ?CSname ?LOlabel
  WHERE { ?CS rdf:type ccso:Course .
 OPTIONAL { ?CS ccso:csName ?CSname } .
          ?CS ccso:hasSyllabus ?SL .
          ?SL ccso:aimsToLO ?LO .
          ?LO rdf:type ccso:LearningOutcome .
 OPTIONAL { ?LO ccso:label ?LOlabel } .
 FILTER  (regex(str(?LOlabel), "multiple access") || regex(str(?LOlabel), "Multiple Access"))  . }
ORDER BY ?CS
```

?CS	?CSname	?LOlabel
<http://xworks.gr/ontologies/ccso#CS335/CS_BSc/CSD/SSE/UoC>	Computer Networks^^xsd:string	Describe the steps used in one common approach to the multiple access problem^^
<http://xworks.gr/ontologies/ccso#CS435/CS_BSc/CSD/SSE/UoC>	Network Technology & Programming Lab^^x...	Describe the steps used in one common approach to the multiple access problem^^
<http://xworks.gr/ontologies/ccso#IM102/IM_MSc/IED/STEF/TEIC>	Computer Networks^^xsd:string	Describe the steps used in one common approach to the multiple access problem^^
<http://xworks.gr/ontologies/ccso#TP40K2/IE_BSc/IED/STEF/TEIC>	Computer Networks^^xsd:string	Describe the steps used in one common approach to the multiple access problem^^

4 results

Comment: In this query the learning outcome '`Multiple Access`' is tested, with both versions of topic's name (first letter capital and not), because *Snap SPAQL plugin* is case sensitive. Results show that 4 Courses in 3 different study programs has subject '`Multiple Access`' as Learning Outcome, with course's IRI, course's name and LO's name listed.

(B)

```
SELECT ?CS ?CSname ?LOlabel
   WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS ccso:hasSyllabus ?SL .
       ?SL ccso:aimsToLO ?LO .
       ?LO rdf:type ccso:LearningOutcome .
  OPTIONAL { ?LO ccso:label ?LOlabel } .
  FILTER  (regex(str(?LOlabel), "object-oriented") || regex(str(?LOlabel),
"Object-Oriented"))  . }
 ORDER BY ?CS
```

CS	CSname	LOlabel
http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC	"Programming"	"Compare and contrast the procedural/functional approach and the object-oriented approach"
http://xworks.gr/ontologies/ccso#CS252/CS_BSc/CSD/SSE/UoC	"Object-Oriented Programming"	"Compare and contrast the procedural/functional approach and the object-oriented approach"
http://xworks.gr/ontologies/ccso#CS342/CS_BSc/CSD/SSE/UoC	"Parallel Programming"	"Compare and contrast the procedural/functional approach and the object-oriented approach"
http://xworks.gr/ontologies/ccso#TP10K4/IE_BSc/IED/STEF/TEIC	"Programming"	"Compare and contrast the procedural/functional approach and the object-oriented approach"
http://xworks.gr/ontologies/ccso#TP20K4/IE_BSc/IED/STEF/TEIC	"Data Structures"	"Compare and contrast the procedural/functional approach and the object-oriented approach"
http://xworks.gr/ontologies/ccso#TP40K4/IE_BSc/IED/STEF/TEIC	"Principles of Software Engineering"	"Compare and contrast the procedural/functional approach and the object-oriented approach"

Comment: In this query the learning outcome '`Object-Oriented`' is tested, with both versions of topic's name (first letter capital and not). Results show that 6 Courses has subject '`Object-Oriented`' as Learning Outcome in 2 different study programs, with course's IRI, course's name and LO's name listed.

25. How many credits has a Student got so far?

(A)

```
SELECT ?Std (SUM(?CScredit) AS ?completedCredits)
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
       ?CS ccso:hasSyllabus ?SL .
       ?SL ccso:creditsECTS ?CScredit .
       ?Std ccso:hasCompleted ?CS .
       ?Std rdf:type ccso:Student .
   FILTER regex(str(?Std), "Starkeiou_Vakis") .    }
GROUP BY ?Std
```

```
SELECT ?Std (SUM(?CScredit) AS ?completedCredits)
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
            ?CS ccso:hasSyllabus ?SL .
            ?SL ccso:creditsECTS ?CScredit .
            ?Std ccso:hasCompleted ?CS .
            ?Std rdf:type ccso:Student .
   FILTER regex(str(?Std), "Starkeiou_Vakis") .   }
GROUP BY ?Std
```

?Std	?completedCredits
<http://xworks.gr/ontologies/ccso#Starkeiou_Vakis>	40

1 results

Comment: In this query the student `Starkeiou_Vakis` is tested. Results show that student `Starkeiou_Vakis` has currently 40 ECTS credits total from the completed courses.

(B)

```
SELECT ?Std (SUM(?CScredit) AS ?completedCredits)
  WHERE { ?CS rdf:type ccso:Course .
  OPTIONAL { ?CS ccso:csName ?CSname } .
        ?CS ccso:hasSyllabus ?SL .
        ?SL ccso:creditsECTS ?CScredit .
        ?Std ccso:hasCompleted ?CS .
        ?Std rdf:type ccso:Student .
   FILTER regex(str(?Std), "Vassileiou_Marcela") .     }
GROUP BY ?Std
```

Std	completedCredits
http://xworks.gr/ontologies/ccso#Vassileiou_Marcela	60

Comment: In this query the student `Vassileiou_Marcela` is tested. Results show that student `Vassileiou_Marcela` has currently 60 ECTS credits total from the completed courses.

26. Can a Student enroll in a Course?

In first example, the scenario examines if student `Starkeiou_Vakis` can enroll in course `CS240/CS_BSc/CSD/SSE/UoC`. In order to enroll in a course, the student should have completed all prerequisites of this course, if any.

(A1)

```
SELECT ?CS ?CSprerequisite ?CSprerequisitename
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:hasPrerequisite ?CSprerequisite .
  OPTIONAL { ?CSprerequisite ccso:csName ?CSprerequisitename } .
  FILTER regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") .   }
```

?CS	?CSprerequisite	?CSprerequisitename
<http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC>	<http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/UoC>	Introduction to Computer Science^^xsd:string
<http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC>	<http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC>	Programming^^xsd:string

2 results

Comment A1: First, the course `CS240/CS_BSc/CSD/SSE/UoC` is tested for prerequisites. Results show that course `CS240/CS_BSc` has 2 prerequisites, the courses `CS100` & `CS150`.

(A2)

```
SELECT ?CSprerequisite
WHERE { ?CSprerequisite rdf:type ccso:Course .
    ?CS rdf:type ccso:Course .
            ?CS ccso:hasPrerequisite ?CSprerequisite .
            ?STD rdf:type ccso:Student .
            FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
  FILTER (regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") && regex(str(?STD), "Starkeiou_Vakis"))    }
```

?CSprerequisite

Comment A2: Next, the ontology is queried for prerequisites of course `CS240/CS_BSc/CSD/SSE/UoC` that student `Starkeiou_Vakis` has not completed (filter `NOT EXISTS`). The ontology returns no results, which means that student `Starkeiou_Vakis` has completed all the prerequisite courses of course `CS240`. This query is used to verify the result of the next query below.

(A3)

```
ASK { ?CSprerequisite rdf:type ccso:Course .
      ?CS rdf:type ccso:Course .
      ?CS ccso:hasPrerequisite ?CSprerequisite .
      ?STD rdf:type ccso:Student .
      FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
   FILTER    (regex(str(?CS),    "CS240/CS_BSc/CSD/SSE/UoC")        &&
regex(str(?STD), "Starkeiou_Vakis"))     }
```

```
ASK { ?CSprerequisite rdf:type ccso:Course .
    ?CS rdf:type ccso:Course .
            ?CS ccso:hasPrerequisite ?CSprerequisite .
            ?STD rdf:type ccso:Student .
            FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
  FILTER (regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") && regex(str(?STD), "Starkeiou_Vakis"))    }
```

Result
False

Comment A3: Finally, the ontology is asked (`ASK` query) if there are any prerequisites of course '`CS240/CS_BSc/CSD/SSE/UoC`' that student `Starkeiou_Vakis` has not completed (filter `NOT EXISTS`). The ontology returns FALSE which means that there no prerequisites of course `CS240` that student `Starkeiou_Vakis` has not completed. Therefore, student `Starkeiou_Vakis` can enroll in course `CS240`.

It must be noted that *Snap SPARQL Query* plugin does not support `ASK` query neither the negation `NOT EXISTS`, thus the *SPARQL Query* plugin is used instead.

In second example, the scenario examines if student `Starkeiou_Nick` can enroll in course '`CS240/CS_BSc/CSD/SSE/UoC`'. In order to enroll in a course, the student should have completed all prerequisites of this course, if any.

(B1)

```
SELECT ?CS ?CSprerequisite ?CSprerequisitename
  WHERE { ?CS rdf:type ccso:Course .
          ?CS ccso:hasPrerequisite ?CSprerequisite .
  OPTIONAL { ?CSprerequisite ccso:csName ?CSprerequisitename } .
  FILTER regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") .    }
```

?CS	?CSprerequisite	?CSprerequisitename
<http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC>	<http://xworks.gr/ontologies/ccso#CS100/CS_BSc/CSD/SSE/UoC>	Introduction to Computer Science^^xsd:string
<http://xworks.gr/ontologies/ccso#CS240/CS_BSc/CSD/SSE/UoC>	<http://xworks.gr/ontologies/ccso#CS150/CS_BSc/CSD/SSE/UoC>	Programming^^xsd:string

2 results

Comment B1: First, the course '`CS240/CS_BSc/CSD/SSE/UoC`' is tested for prerequisites. Results show that course `CS240/CS_BSc` has 2 prerequisites, course `CS100` & `CS150`.

(B2)

```
SELECT ?CSprerequisite
WHERE { ?CSprerequisite rdf:type ccso:Course .
    ?CS rdf:type ccso:Course .
            ?CS ccso:hasPrerequisite ?CSprerequisite .
            ?STD rdf:type ccso:Student .
            FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
  FILTER (regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") && regex(str(?STD), "Starkeiou_Nick"))    }
```

?CSprerequisite
CS150/CS_BSc/CSD/SSE/UoC

Comment B2: Next, the ontology is queried for the prerequisites of course 'CS240/CS_BSc/CSD/SSE/UoC' that student Starkeiou_Nick has not completed (filter NOT EXISTS). The ontology returns 1 result, course CS150, which means that student Starkeiou_Vakis has not completed all the prerequisite courses of course CS240. This query is used to verify the result of the next query below.

It must be noted that *Snap SPARQL Query* plugin does not support ASK query neither the negation NOT EXISTS, thus the *SPARQL Query* plugin is used instead.

(B3)

```
ASK { ?CSprerequisite rdf:type ccso:Course .
      ?CS rdf:type ccso:Course .
      ?CS ccso:hasPrerequisite ?CSprerequisite .
      ?STD rdf:type ccso:Student .
      FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
   FILTER (regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC")  &&
regex(str(?STD), "Starkeiou_Nick"))     }
```

```
ASK { ?CSprerequisite rdf:type ccso:Course .
    ?CS rdf:type ccso:Course .
            ?CS ccso:hasPrerequisite ?CSprerequisite .
            ?STD rdf:type ccso:Student .
            FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
  FILTER (regex(str(?CS), "CS240/CS_BSc/CSD/SSE/UoC") && regex(str(?STD), "Starkeiou_Nick"))    }
```

Result
True

Comment B3: Finally, the ontology is asked (ASK query) if there are any prerequisites of course 'CS240/CS_BSc/CSD/SSE/UoC' that student Starkeiou_Nick has not completed (filter NOT EXISTS). The ontology returns TRUE, which means that there is at least one prerequisite of course CS240 that student Starkeiou_Nick has not completed. Therefore, student Starkeiou_Nick can not enroll in course CS240.

In third example, the Virtuoso SPARQL endpoint is used and the scenario examines if student Vassileiou_Marcela can enroll in course 'TP40K4/IE_BSc/IED/STEF/ TEIC'. In order to enroll in a course, the student should have completed all prerequisites of this course, if any.

(C1)

```
ASK { ?CSprerequisite rdf:type ccso:Course .
      ?CS rdf:type ccso:Course .
      ?CS ccso:hasPrerequisite ?CSprerequisite .
      ?STD rdf:type ccso:Student .
      FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
   FILTER (regex(str(?CS), "TP40K4/IE_BSc/IED/STEF/TEIC")  &&
regex(str(?STD), "Vassileiou_Marcela"))     }
```

```
ASK { ?CSprerequisite rdf:type ccso:Course .
      ?CS rdf:type ccso:Course .
        ?CS ccso:hasPrerequisite ?CSprerequisite .
        ?STD rdf:type ccso:Student .
        FILTER (NOT EXISTS{ ?STD ccso:hasCompleted ?CSprerequisite } ).
   FILTER (regex(str(?CS), "TP40K4/IE BSc/IED/STEF/TEIC")  && regex(str(?STD), "Vassileiou Marcela"))     }
```

Run Query | Reset

false

Comment: The ontology is asked (`ASK` query) if there are any prerequisites of course `TP40K4/IE_BSc/IED/STEF/TEIC` that student `Vassileiou_Marcela` has not completed (filter `NOT EXISTS`). The ontology returns FALSE which means that there no prerequisites of course `TP40K4` that student `Vassileiou_Marcela` has not completed. Therefore, student `Vassileiou_Marcela` can enroll in course `TP40K4`.

27. Can a Person register in a study program for a degree?

In first example, the scenario examines if `Assarioti_Mellisandri` can register in study program `MSc_School_Psychology/SP_MSc/PSY/SoC/UoC`. In order to register in a study program, the person should have at least one of the required degrees.

(A1)

```
SELECT ?PSprerequisite ?PSprerequisitename
  WHERE { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
  OPTIONAL { ?PSprerequisite ccso:psName ?PSprerequisitename } .
  FILTER regex(str(?PS), "MSc_School_Psychology/SP_MSc/PSY/SoC/UoC") .    }
```

?PSprerequisite	?PSprerequisitename
<http://xworks.gr/ontologies/ccso#BSc_Psychology/PSY_BSc/SoC/UoC>	Bachelor of Science (BSc) in Psychology of Psychology Department ...

Comment A1: First, the study program `MSc_School_Psychology/SP_MSc/PSY/SoC/ UoC` is tested for required degrees. Results show that study program `MSc_School_Psychology/SP_MSc` requires the degree `BSc in Psychology`.

(A2)

```
SELECT ?PSprerequisite ?PSprerequisitename
  WHERE { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
  OPTIONAL { ?PSprerequisite ccso:psName ?PSprerequisitename } .
          ?STD rdf:type ccso:Student .
          ?STD ccso:hasDegree ?CSprerequisite .
  FILTER (regex(str(?PS), "MSc_School_Psychology/SP_MSc/PSY/SoC/UoC")  && regex(str(?STD), "Assarioti_Mellisandri"))    }
```

?PSprerequisite	?PSprerequisitename
<http://xworks.gr/ontologies/ccso#BSc_Psychology/PSY_BSc/SoC/UoC>	Bachelor of Science (BSc) in Psychology of Psychology Department ...

1 results

Comment A2: Next, the ontology is queried for required degrees for study program `MSc_School_Psychology/SP_MSc/PSY/SoC/UoC` that `Assarioti_Mellisandri` has. The ontology returns one result that shows `Assarioti_Mellisandri` has the required degree `BSc_Psychology` for study program `MSc_School_Psychology`.

(A3)

```
ASK { ?PSprerequisite rdf:type ccso:ProgramofStudy .
      ?PS rdf:type ccso:ProgramofStudy .
      ?PS ccso:requiresProgram ?PSprerequisite .
      ?STD rdf:type ccso:Student .
      ?STD ccso:hasDegree ?CSprerequisite .
```

```
        FILTER (regex(str(?PS), "MSc_School_Psychology/SP_MSc/PSY/SoC/UoC")  &&
      regex(str(?STD), "Assarioti_Mellisandri"))      }
```

```
ASK { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
          ?STD rdf:type ccso:Student .
          ?STD ccso:hasDegree ?CSprerequisite .
 FILTER (regex(str(?PS), "MSc_School_Psychology/SP_MSc/PSY/SoC/UoC") && regex(str(?STD), "Assarioti_Mellisandri"))   }
```

Result
True

Comment A3: Finally, the ontology is asked (`ASK` query) if `Assarioti_ Mellisandri` has a required degree for study program `MSc_School_Psychology`. The ontology returns TRUE which means that `Assarioti_ Mellisandri` has at least one of the required degrees. Therefore, `Assarioti_ Mellisandri` can register in study program `MSc School Psychology`.

It must be noted that *Snap SPARQL Query* plugin does not support `ASK` query, thus the *SPARQL Query* plugin is used instead.

In second example, the scenario examines if `Raderis_Jim` can register in study program `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC`. In order to register in a study program, the person should have one of the degrees that required.

(B1)

```
SELECT ?PSprerequisite ?PSprerequisitename
  WHERE { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
  OPTIONAL { ?PSprerequisite ccso:psName ?PSprerequisitename } .
FILTER regex(str(?PS), "MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC") .   }
```

?PSprerequisite	?PSprerequisitename
<http://xworks.gr/ontologies/ccso#BSc_Computer_Science/CS_BSc/CSD/SSE/UoC>	Bachelor of Science (BSc) in Computer Science of Computers Sciences Department (CSD) ...
<http://xworks.gr/ontologies/ccso#BSc_Informatics_Engineering/IE_BSc/IED/STEF/TEIC>	Bachelor of Science (BSc) in Informatics Engineering of Department of Informatics Engine...

2 results

Comment B1: First, the study program `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC` is tested for required degrees. Results show that study program `MSc_Informatics_&_Multimedia/SP_MSc` requires the degrees `BSc_Computer_Science` & `Bsc_Informatics_Engineering` (at least one of them).

(B2)

```
SELECT ?PSprerequisite ?PSprerequisitename
  WHERE { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
  OPTIONAL { ?PSprerequisite ccso:psName ?PSprerequisitename } .
          ?STD rdf:type ccso:Student .
          ?STD ccso:hasDegree ?PSprerequisite .
  FILTER (regex(str(?PS), "MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC")  && regex(str(?STD), "Raderis_Jim"))   }
```

?PSprerequisite	?PSprerequisitename
<http://xworks.gr/ontologies/ccso#BSc_Informatics_Engineering/IE_BSc/IED/STEF/TEIC>	Bachelor of Science (BSc) in Informatics Engineering of Department of Informatics Engineering...

1 results

Comment B2: Next, the ontology is queried for required degrees for study program `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC` that `Raderis_Jim` has. The ontology returns one result that shows `Raderis_Jim` has the required degree `Bsc_Informatics_Engineering`.

(B3)

```
ASK { ?PSprerequisite rdf:type ccso:ProgramofStudy .
      ?PS rdf:type ccso:ProgramofStudy .
      ?PS ccso:requiresProgram ?PSprerequisite .
      ?STD rdf:type ccso:Student .
      ?STD ccso:hasDegree ?PSprerequisite .
   FILTER (regex(str(?PS), "MSc_Informatics_&_Multimedia/IM_MSc/IED/
```

```
STEF/TEIC")   && regex(str(?STD), "Raderis_Jim"))      }
```

```
ASK { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
          ?STD rdf:type ccso:Student .
          ?STD ccso:hasDegree ?PSprerequisite .
 FILTER (regex(str(?PS), "MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC") && regex(str(?STD), "Raderis_Jim"))    }
```

Result
True

Comment B3: Finally, the ontology is asked (`ASK` query) if `Raderis_Jim` has a required degree for the study program `MSc_Informatics_&_Multimedia`. The ontology returns TRUE which means that student `Raderis_Jim` has at least one of the required degrees. Therefore, `Raderis_Jim` can register in study program `MSc_Informatics_&_Multimedia`.
It must be noted that *Snap SPARQL Query* plugin does not support `ASK` query, thus the *SPARQL Query* plugin is used instead.

In second example, the scenario examines if `Lioni_Catherine` can register in study program `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC`. In order to register in a study program, the student should have one of the degrees that required.

(C1)

```
SELECT ?degreeTitle
  WHERE { ?PR rdf:type foaf:Person .
          ?PR ccso:hasDegree ?PS  .
 OPTIONAL { ?PS ccso:degreeTitle ?degreeTitle } .
 FILTER regex(str(?PR), "Lioni_Catherine") }
```

?degreeTitle
Bachelor of Science in Psychology of University of Crete^^xsd:string

1 results

Comment C1: As already has examined in *Query 27B1*, study program `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC` requires one of the degrees `BSc_Computer_Science` & `Bsc_Informatics_Engineering`.
Thus, this query examines which degrees `Lioni_Catherine` has. Results show that `Lioni_Catherine` has the degree `Bachelor of Science in Psychology of University of Crete`.

(C2)

```
SELECT ?PSprerequisite
  WHERE { ?PSprerequisite rdf:type ccso:ProgramofStudy .
          ?PS rdf:type ccso:ProgramofStudy .
          ?PS ccso:requiresProgram ?PSprerequisite .
          ?STD rdf:type ccso:Student .
          ?STD ccso:hasDegree ?PSprerequisite .
 FILTER (regex(str(?PS), "MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC") && regex(str(?STD), "Lioni_Catherine "))    }
```

?PSprerequisite

0 results

Comment C2: Next, the ontology is queried for required degrees for study program `MSc_Informatics_&_Multimedia/IM_MSc/IED/STEF/TEIC` that `Lioni_Catherine` has. The ontology returns no results, which means `Lioni_Catherine` does not have any of the required degrees for

MSc_Informatics_&_Multimedia.

(C3)

```
ASK { ?PSprerequisite rdf:type ccso:ProgramofStudy .
      ?PS rdf:type ccso:ProgramofStudy .
      ?PS ccso:requiresProgram ?PSprerequisite .
      ?STD rdf:type ccso:Student .
      ?STD ccso:hasDegree ?PSprerequisite .
   FILTER (regex(str(?PS), "MSc_Informatics_&_Multimedia/IM_MSc/IED/
STEF/TEIC")  && regex(str(?STD), "Lioni_Catherine"))     }
```

Comment C3: Finally, the ontology is asked (`ASK` query) if `Lioni_Catherine` has a required degree for the study program `MSc_Informatics_&_Multimedia`. The ontology returns FALSE which means that `Lioni_Catherine` has not at least one of the required degrees. Therefore, `Lioni_Catherine` cannot register in study program `MSc Informatics & Multimedia`.

4.2. PRESENTATION

The final developed ontology comprises of 41 concepts in a taxonomy, 9 of which are the top-level concepts of the ontology, namely the *FieldofStudy*, *EducationalOrganization*, *Person*, *ProgramofStudy*, *Course*, *Syllabus*, *Event*, *Topic* and *Resource* (*Figure 88*).

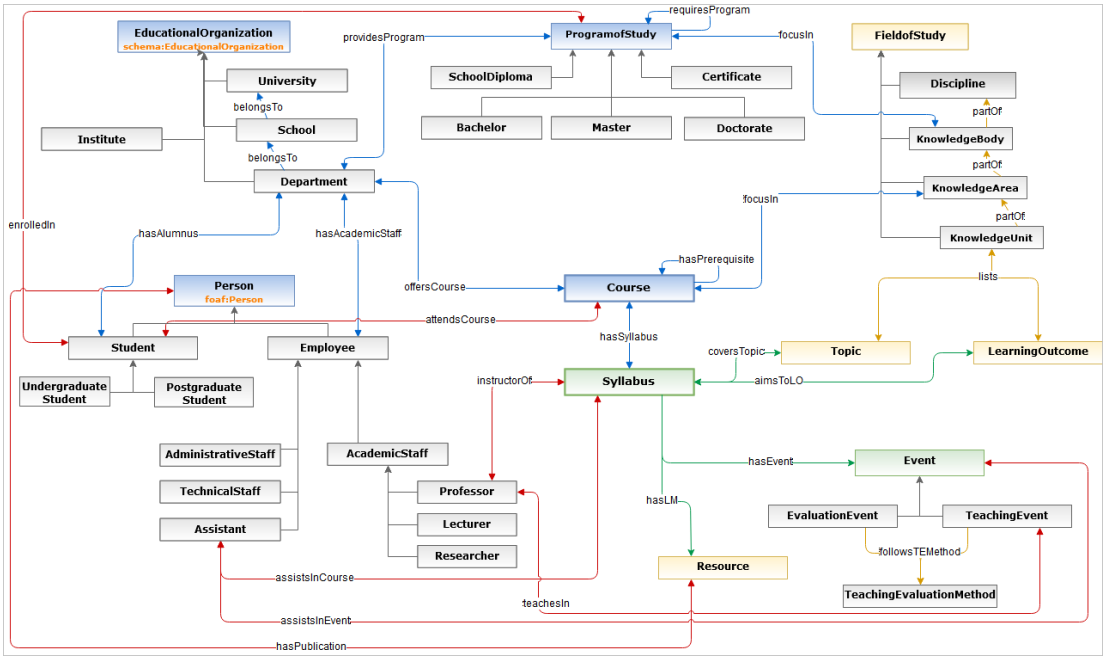


Figure 88: CCSO taxonomy and relationships

It also includes 54 objects properties for establishing relations between concepts, 40 of which are participating in 20 pairs of inverse object properties, and 76 data properties for describing concepts characteristics in detail (*Figure 89*).

The generated ontology is also available online to be downloaded and reused at: http://xworks.gr/ontologies.

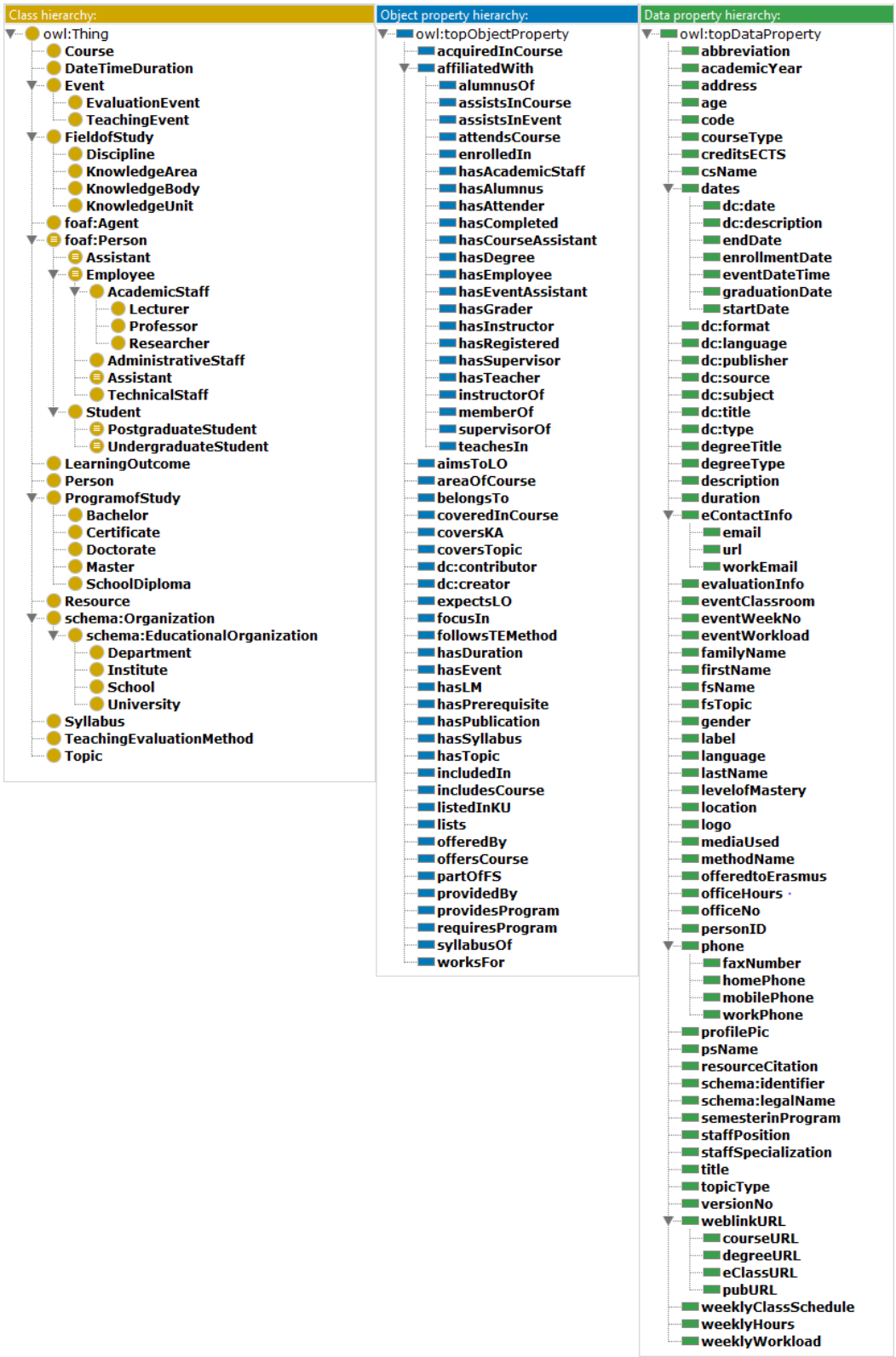


Figure 89: Classes and Properties of CCSO in Protégé

Figure 90 depicts a thumbnail of CCSO Taxonomy which includes Classes of CCSO, their properties and their relationships. A larger image can be found online at http://xworks.gr/ontologies [45].

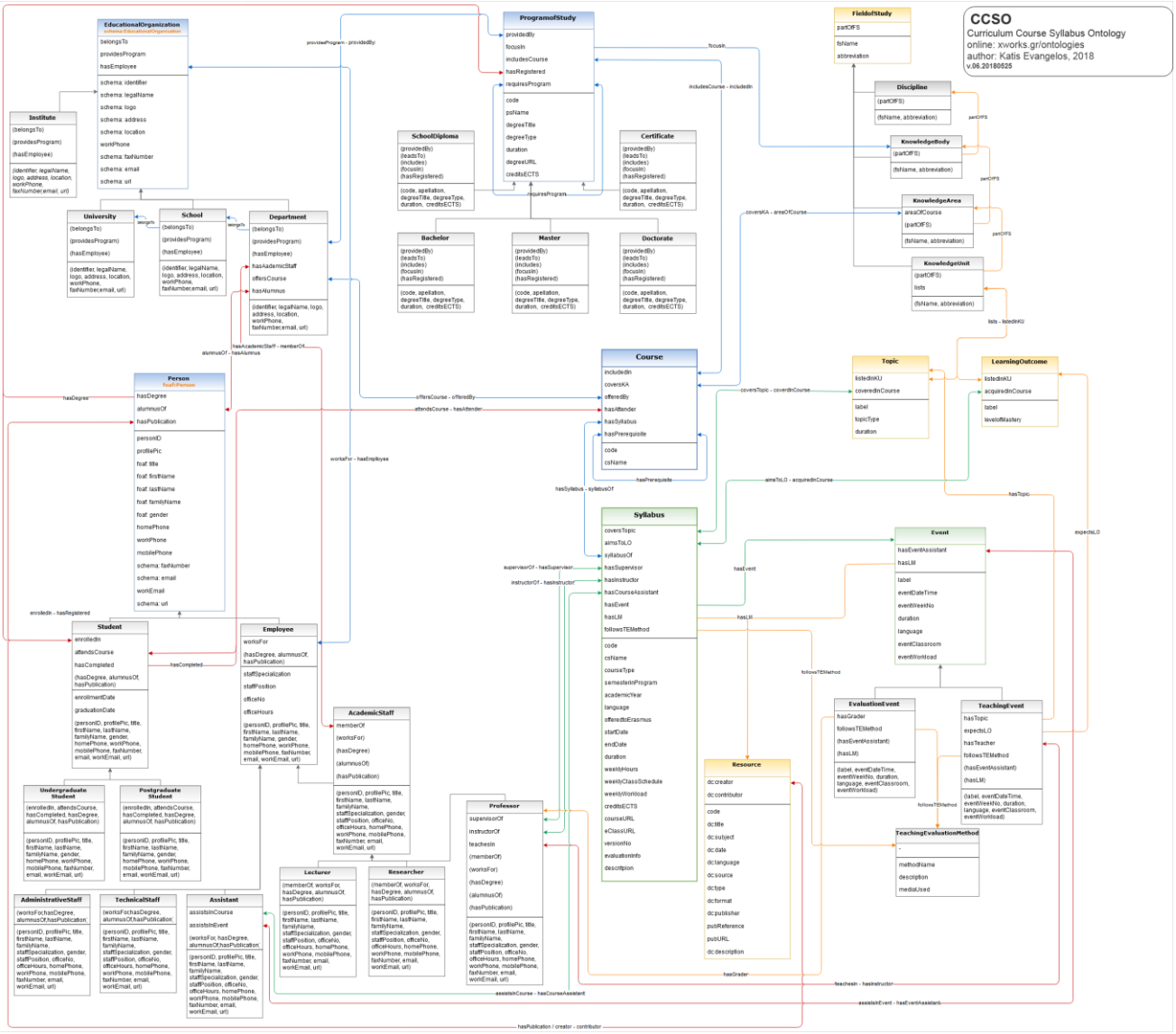


Figure 90: CCSO Taxonomy with Classes, their properties and relationships

All entities are enriched with additional annotation information, written in two languages, English and Greek, giving to ontology a bilingual character, as shown below in *Figure 92*. In addition, the ontology enriched with a large number of instances in most of its classes.

[45] CCSO Taxonomy full version: http://xworks.gr/ontologies/CCSO_Taxonomy.png

4.3. DOCUMENTATION

The last phase is the documentation of the ontology. Effective knowledge sharing and reuse, requires adequate documentation. *Skuce* [101] pointed out that one of the main barriers to effective knowledge sharing is the inadequate documentation of existing ontologies and knowledge bases. To address this issue, *Skuce* proposed all important assumptions must be documented, both the main concepts and the primitives used to describe the concepts.

Undoubtedly, providing sufficient documentation enhances the comprehension of the ontology and makes it easier to be applied and reused by other users and applications. Based on experience, many authors leave the documentation as the last task but that result in poorly documented ontologies, with few or almost no comments. As a consequence, the understanding and reuse of the ontology, even though it is consistent of terms and semantics, is becoming difficult, especially for users not involved in their design (e.g. when a user search for existing ontologies in the domain of his interest).

Considering this, documentation, like evaluation too, should be performed in parallel with the whole ontology development lifecycle [102]. This approach has been followed in this thesis· we performed the documentation along with the ontology development, from the specification of the ontology purpose to the evaluation phase, as shown in *Figure 91* where the dotted lines indicate constant interaction. We provide internal and external documentation with various pieces of information for the developed ontology.

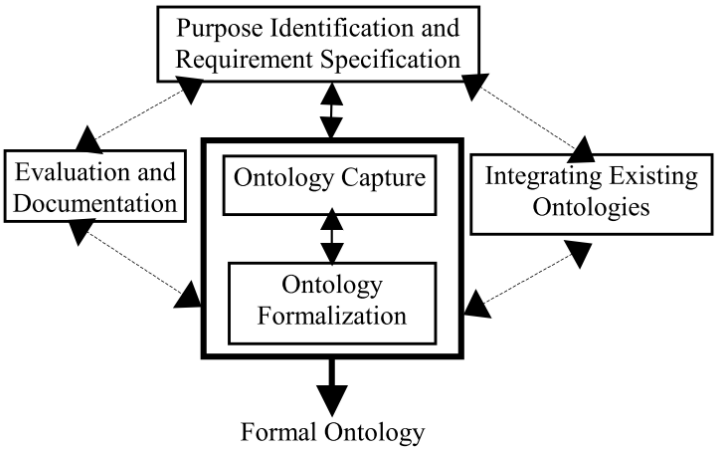


Figure 91: Ontology Development Process (source: Falbo *et al.*[102])

Internal documentation includes information annotated in ontology elements as metadata. Using the five pre-defined annotation properties that OWL provides, we attached useful information on ontology elements. The property *'rdfs:comment'* used to attach description and clarification on all classes as well as object properties and data properties. We also used the property *'rdfs:label'* to add meaningful names to ontology elements. All classes and properties have been written in two languages, English and Greek, giving to ontology a bilingual character. *Figure 92* illustrates the annotations of one class (point 1, yellow title bar), one object property (point 2, blue title bar) and one data property (point 3, green title bar).

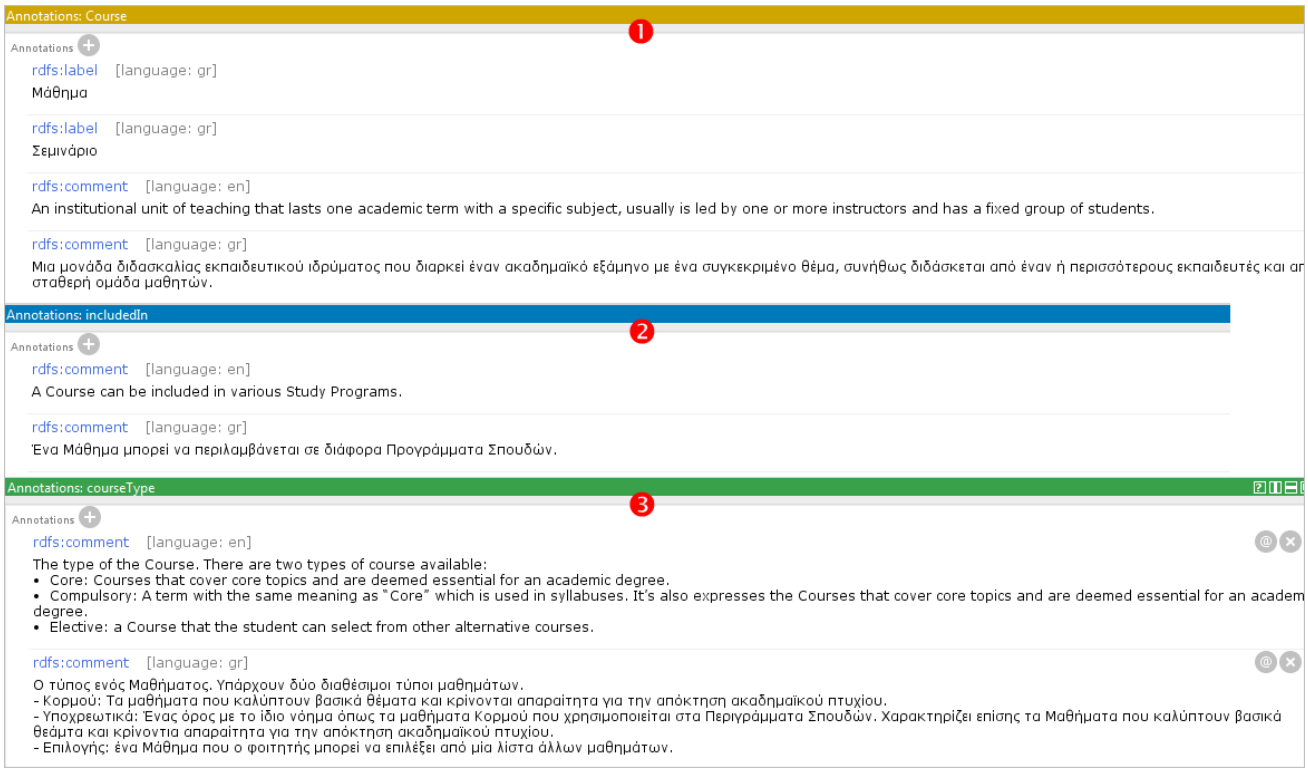


Figure 92: Annotations of *Course* class, *includedIn* object property and *courseType* data property

External documentation includes an extended document that describes in detail step-by-step the whole ontology development process, including purpose, class definitions, description of class properties, summary of definitions, and descriptive tables for each class with all relevant to class information such as description, super/sub-classes, properties, inverse classes, disjoints, *etc*. The evaluation process is also fully described with screenshots and examination of results that ontology returns in SPARQL queries based on the competency questions.

CHAPTER 5. CONCLUSION

5.1. SYNOPSIS

Ontologies have a wide range of use in the area and many educational processes and learning activities can benefit Education and the whole education field. They can be used, among others for the semantic annotation of educational entities for knowledge reuse and interoperability.

Curriculum, Course and Syllabus are considered as top-level entities in an educational organization. They include important information and play significant role in many educational processes and learning activities.

In this thesis we modeled the domain of a tertiary education curriculum conceptualizing core knowledge structures like Curriculum, Course and Syllabus.

We presented the ontology developed that semantically annotates these educational entities with sufficiency and richness in order to support rich services on top. For example, an ontology based web application can be developed that enhances the searcheability and discoverability of useful information about institution and its educational offerings. Moreover, Course Management System can benefit from the semantically annotated syllabi and offer improved services to teachers and students such as rich information about courses, course alignment and recommendation, searching and matching for courses that cover specific knowledge areas that students interested in, discover courses that aim to specific learning outcomes *e.tc*.

A literature review of ontologies in Education as well as in the context of Curriculum and Syllabus is presented.

We described in detail the methodology and development process of the educational ontology. All conceptualized elements are presented and analytically documented with their properties and relations.

Also the evaluation process of the ontology has been analyzed, which included technical evaluation against common errors and pitfalls in ontology developments as well as quality

examination. Quality examination tested the response of the ontology against a list of questions that introduce scenarios in terms of its application.

Finally, we presented our work and discussed how it can be used to improve curriculum management and development and enable syllabus semantic searching, matching, interlinking and recommendation.

5.2. DIFFICULTIES

As already has been mentioned, a significant number of methodologies, languages and tools have been developed for ontology engineering and development. However, it is true that building a good ontology that meets its requirements is not an easy task.

The great richness in literature about *Semantic Web* and *Ontologies* as well as Ontologies in Education [27], [66] offer significant help in tackling possible problems however difficulties may arise during ontology development [79], [83]. In addition to that, it takes time and much work to explore and study all these interesting works.

Existing ontologies have been important guides in ontology construction, especially in the stages of ontology design and main concepts identification. However, there are two obstacles encountered in educational ontologies that have been found in this survey. The first is that educational ontologies presented in literature, have not their ontology files available online, thus only the research documentation that was available was used [38], [63], [34], [75], [79], [82]. On the other hand, the majority of relevant ontology files were found in this survey with search engines and repositories, lack of documentation which makes it difficult to understand and reuse them.

Notwithstanding is the fact that the absence of one, or at least a few but reliable curated centralized ontology repositories, effects unclassified and un-evaluated content. Despite of the ability to use some generic or ontological search engines as well as existing ontology repositories (*more about that in 3.4.1*), finding good and relevant ontologies, can be a quite difficult and time-consuming process.

5.3. FUTURE WORK

Nowadays, the importance of ontology is well-received in many domains and sciences with computer science and education to present increasing perspective [27], [66]. As to the CCSO, we think of some promising future directions. As a first step, the ontology can be used by students attending "Semantic Web" course included in the post graduate study program of *Department of Informatics Engineering* in local institution. Students can help to catch study programs and courses taught in our institution and fill ontology with real data. In this way, more users will be able to explore its capabilities, test its achievements and bring into focus omissions and improvements.

As a step forward a web application, based on CCSO, can be developed that will allow institution employees as well as students to navigate and query data stored in the ontology. In this way, employees, instructors and students can be better informed about several educational offerings in our institution.

Moreover, *Open eClass* - the Course Management System that is used in our institution – can benefit from the semantically annotated syllabi and offer improved services to teachers and students for better searching and exploring of syllabus information.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Sci. Am.*, vol. 284, no. 5, pp. 28–37, 2001.

[2] N. Spivack, "Web 3.0: The Third Generation Web is Coming," 2013.

[3] G. Antoniou, Groth, Paul, and Frank van Harmelen, *A Semantic Web Primer*, Thrid Edit. Cambridge, London, England: MIT Press, 2012.

[4] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

[5] W. N. Borst and W. N. Borst, "Construction of Engineering Ontologies for Knowledge Sharing and Reuse." Centre for Telematics and Information Technology (CTIT), Netherlands, 1997.

[6] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: principles and methods," *Data Knowl. Eng.*, vol. 25, no. 1–2, pp. 161–197, 1998.

[7] M. R. Genesereth and N. J. Nilsson, "Logical Foundations of Artificial Intelligence," *Morgan Kaufmann*, vol. 58, 1987.

[8] T. Gruber, "Ontology," *Encyclopedia of Database Systems*. Springer-Verlag, 2008.

[9] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods and Applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–136, 1996.

[10] Y. Sure and R. Studer, "Methodology, Tools & Case Studies for Ontology based Knowledge Management," University of Karlsruhe, 2003.

[11] O. Lassila and D. McGuinness, "The Role of Frame-Based Representation on the Semantic Web," *Electron. Artic. Comput. Inf. Sci.*, vol. 6, no. 5, p. 10, 2001.

[12] N. Guarino, "Formal Ontology and Information Systems," in *First International Conference (FOIS'98)*, 1998, vol. 46, pp. 3–15.

[13] A. Gómez-Pérez and R. Benjamins, "Overview of Knowledge Sharing and Reuse Components: Ontologies and problem-solving methods," in *IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, 1999.

[14] F. Gandon and G. Schreiber, "RDF 1.1 XML Syntax," *W3C*, 2014. [Online]. Available: https://www.w3.org/TR/rdf-syntax-grammar/.

[15] D. Brickley and R. V. Guha, "RDF Schema 1.1," *W3C*, 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/.

[16] D. Fensel, F. Van Harmelen, M. Klein, H. Akkermans, H. Schnurr, R. Studer, J. Hughes, U. Krohn, and J. Davies, "On-To-Knowledge : Ontology-based Tools for Knowledge Management," in *eBusiness and eWork 2000 Conference*, 2000, pp. 18–20.

[17] D. Fensel, F. Van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider, "OIL: An ontology infrastructure for the semantic web," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 38–45, 2001.

[18] D. L. McGuinness, R. Fikes, J. Hendler, and L. A. Stein, "DAML+OIL: an Ontology Language for the Semantic Web," *IEEE Intell. Syst.*, vol. 17, no. 5, pp. 72–80, 2002.

[19] A. Gómez-Pérez and O. Corcho, “Ontology Languages for the Semantic Web,” *IEEE Intell. Syst.*, vol. 17, no. 1, pp. 54–60, 2002.

[20] J. Cardoso and A. L. N. Escórcio, “Editing Tools for Ontology Construction,” *Semant. Web Serv. Theory, Tools Appl.*, pp. 1–27, 2007.

[21] (SCBIR) Stanford Center for Biomedical Informatics Research, “Protégé - A free, open-source ontology editor and framework for building intelligent systems,” *Stanford University*, 2009. [Online]. Available: https://protege.stanford.edu.

[22] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, “The Evolution of Protégé: An Environment for Knowledge-Based Systems Development,” *Int. J. Hum. Comput. Stud.*, vol. 58, no. 1, pp. 89–123, 2001.

[23] R. Sivakumar and P. V Arivoli, “Ontology Visualization Protégé Tools – A Review,” *Int. J. Adv. Inf. Techn ology*, vol. 1, no. 4, pp. 1–11, 2011.

[24] I. S. Group, “HermiT OWL Reasoner,” *University of Oxford*. [Online]. Available: http://www.hermit-reasoner.com.

[25] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, “HermiT: an OWL 2 reasoner,” *J. Autom. Reason.*, vol. 53, no. 3, pp. 245–269, 2014.

[26] J. Cardoso, “The Semantic Web Vision: Where are we?,” *IEEE Intell. Syst.*, vol. 22, no. 5, 2007.

[27] D. Dicheva, S. Sosnovsky, G. Tatiana, and P. Brusilovsky, “Ontological Web Portal for Educational Ontologies,” in *12th International Conference of Artificial Intelligence in Education*, 2005, pp. 19–27.

[28] M. Grandbastien and B. Huyinh Kim Bang, “Ontologies for enabling Learning Objects retrieval: A case study,” in *Sixth Int’l Workshop on Ontologies and Semantic Web for E-Learning, in conjunction with Int. Conf. on Intelligent Tutoring Systems*, 2008, pp. 50–54.

[29] F. Van Assche, “Linking Learning Resources to Curricula by using Competencies,” in *First International Workshop on Learning Object Discovery & Exchange (LODE’07)*, 2007, pp. 80–91.

[30] G. Paquette, “An Ontology and a Software Framework for Competency Modeling and Management Competency in an Instructional Engineering Method (MISA),” *Educ. Technol. Soc.*, vol. 10, no. 3, pp. 1–21, 2007.

[31] E. Alberdi and D. H. Sleeman, “ReTAX: a step in the automation of taxonomic revision,” *Artif. Intell.*, vol. 91, no. 2, pp. 257–279, 1997.

[32] S. Sosnovsky and T. Gavrilova, “Development of educational ontology for C-programming,” *Inf. Theor. Appl.*, vol. 13, pp. 303–308, 1991.

[33] S. Bianchi and C. Mastrodonato, “Use of Ontologies to Annotate and Retrieve Educational Contents : the AquaRing Approach,” *Learn. Knowl. Soc.*, vol. 5, no. 1, pp. 211–220, 2009.

[34] M. Oprea, “An Educational Ontology for Teaching University Courses,” in *6th International Conference on Virtual Learning–ICVL*, 2011, no. 39, pp. 117–122.

[35] J. Mesaric and B. Dukic, “An Approach to Creating Domain Ontologies for Higher Education in Economics,” in *29th International Conference on Information Technology Interfaces, ITI 2007*, 2007, pp. 75–80.

[36] “SUMO (Suggested Upper Merged Ontology).” [Online]. Available: www.ontologyportal.org.

[37] B. Swartout, R. Patil, K. Knight, and T. Russ, “Toward distributed use of large-scale ontologies,” in *Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*, 1996, pp. 138–148.

[38] H. Chung and J. Kim, “An Ontological Approach for Semantic Modeling of Curriculum and Syllabus in Higher Education,” *Int. J. Inf. Educ. Technol.*, vol. 6, no. 5, pp. 365–369, 2016.

[39] R. Mizoguchi, K. Sinitsa, and M. Ikeda, “Task Ontology Design for Intelligent Educational / Training Systems,” in *Position Paper for ITS’96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs*, 1996.

[40] R. R. Amorim, M. Lama, E. Sánchez, A. Riera, and X. A. Vila, “A learning design ontology based on the IMS specification,” *Educ. Technol. Soc.*, vol. 9, no. 1, pp. 38–57, 2006.

[41] C. Knight, D. Gasevic, and G. Richards, “An ontology-based framework for bridging learning design and learning content,” *Educ. Technol. Soc.*, vol. 9, no. 1, pp. 23–37, 2006.

[42] À. Rius and E. García-Barriocanal, “An Ontology to Automate Learning Scenarios? An Approach to its Knowledge Domain,” vol. 4, 2008.

[43] B. Barros, M. F. Verdejo, T. Read, and R. Mizoguchi, “Applications of a collaborative learning ontology,” in *Mexican International Conference on Artificial Intelligence*, 2002, pp. 301–310.

[44] A. Schmidt and C. Kunzmann, “Towards a Human Resource Development ontology for combining Competence Management and Technology-Enhanced Workplace Learning,” in *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, 2006, pp. 1078–1087.

[45] A. Schmidt and C. Kunzmann, “Sustainable competency-oriented human resource development with ontology-based competency catalogs,” in *eChallenges*, 2007, vol. 2007.

[46] M.-A. Sicilia, “Ontology-based competency management: infrastructures for the knowledge intensive learning organization,” in *Intelligent learning infrastructure for knowledge intensive organizations: A semantic Web Perspective*, IGI Global, 2005, pp. 302–324.

[47] R. Vas, “Educational Ontology and Knowledge Testing,” *Electron. J. Knowl. Manag.*, vol. 5, no. 1, pp. 123–130, 2007.

[48] N. Press, *Understanding Metadata*, vol. 20. Baltimore: National Information Standards Organization (NISO), 2004.

[49] P. Barker and L. M. Campbell, “Metadata for learning materials: an overview of existing standards and current developments,” *Technol. Instr. Cogn. Learn.*, vol. 7, no. 3–4, pp. 225–243, 2010.

[50] J. Valaski, A. Malucelli, S. Reinehr, and R. Santos, “Ontology to Classify Learning Material in Software Engineering Knowledge Domain,” *ONTOBRAS-MOST*, pp. 37–48, 2011.

[51] J. Valaski, A. Malucelli, and S. Reinehr, “Recommending Learning Materials according to Ontology-based Learning Styles,” in *7th International Conference on Information Technology and Applications (ICITA 2011)*, 2011, pp. 71–75.

[52] J. Valaski, S. Reinehr, and A. Malucelli, “An ontology to support the classification of learning material in an organizational learning environment: An evaluation,” *Interact. Technol. Smart Educ.*, vol. 14, no. 1, pp. 67–87, 2017.

[53] M.-C. Lee, D. Y. Ye, and T. I. Wang, “Java Learning Object Ontology,” *Fifth IEEE Int. Conf. Adv. Learn. Technol.*, pp. 0–4, 2005.

[54] D. A. Koutsomitropoulos, G. D. Solomou, A. D. Alexopoulos, and T. S. Papatheodorou, “Knowledge Management and Acquisition in Digital Repositories-A Semantic Web Perspective.,” in *KMIS*, 2009, pp. 117–122.

[55] D. A. Koutsomitropoulos, A. D. Alexopoulos, G. D. Solomou, and T. S. Papatheodorou, “The use of metadata for educational resources in digital repositories: Practices and perspectives,” *D-Lib Mag.*, vol. 16, no. 1/2, 2010.

[56] K. Verbert and E. Duval, “ALOCOM: a generic content model for learning objects,” *Int. J. Digit. Libr.*, vol. 9, no. 1, pp. 41–63, 2008.

[57] Y. Poulakakis, K. Vassilakis, M. Kalogiannakis, and S. Panagiotakis, “Ontological modeling of educational resources: a proposed implementation for Greek schools,” *Educ. Inf. Technol.*, vol. 22, no. 4, pp. 1737–1755, 2017.

[58] R. Mizoguchi and J. Bourdeau, “Using Ontological Engineering to Overcome Common AI-ED Problems,” *Int. J. Artif. Intell. Educ.*, vol. 11, pp. 107–121, 2000.

[59] D. G. Sampson, M. D. Lytras, G. Wagner, and P. Diaz, “Ontologies and the Semantic Web for E-learning,” *Educ. Technol. Soc.*, vol. 7, no. 4, pp. 26–28, 2004.

[60] L. Aroyo, D. Dicheva, and A. Cristea, “Ontological Support for Web Courseware Authoring,” in *International Conference on Intelligent Tutoring Systems*, 2002, pp. 270–280.

[61] C. Pahl and E. Holohan, “Ontology Technology for the Development and Deployment of Learning Technology Systems - A Survey,” 2004.

[62] T. Kerkiri, A. Manitsaris, and I. Mavridis, “How e-learning systems may benefit from ontologies and recommendation methods to efficiently personalise resources,” *Int. J. Knowl. Learn.*, vol. 5, no. 3–4, pp. 347–370, 2009.

[63] H. S. Chung and J. M. Kim, “Ontology Design for Creating Adaptive Learning Path in e-Learning Environment,” in *International MultiConference of Engineers and Computer Scientists*, 2012, vol. I, pp. 14–17.

[64] J. M. Gascueña, A. Fernández-Caballero, and P. González, "Domain Ontology for Personalized E-Learning in Educational Systems," in *Sixth International Conference on Advanced Learning Technologies*, 2006, pp. 456–458.

[65] R. Mizoguchi, "A step towards ontological engineering," in *12th National Conference on AI of JSAI*, 1998, pp. 24–31.

[66] M. Al-Yahya, R. George, and A. Alfaries, "Ontologies in e-Learning: Review of the Literature," *Int. J. Softw. Eng. Its Appl.*, vol. 9, no. 2, pp. 67–84, 2015.

[67] A. Gomez-Perez, M. Fernández-López, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

[68] G. Adorni, S. Battigelli, and D. Brondo, "CADDIE and IWT: two different ontology-based approaches to Anytime , Anywhere and Anybody Learning," *J. e-Learning Knowl. Soc.*, vol. 6, no. 2, pp. 53–66, 2010.

[69] Y. L. Chi, "Developing Curriculum Sequencing for Managing Multiple Texts in e-Learning System," in *Proceedings of International Conference on Engineering Education*, 2010, pp. 1–8.

[70] M. C. Van der Wende, "The Bologna Declaration: Enhancing the Transparency and Competitiveness of European Higher Education," *J. Stud. Int. Educ.*, vol. 4, no. 2, pp. 3–10, 2000.

[71] M. Ronchetti and J. Sant, "Curriculum Management and Review: an ontology-based solution," Trento, Italy, 2007.

[72] J. S. Alomari, "Ontology for Academic Program Accreditation," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 7, pp. 123–127, 2016.

[73] E. Kontopoulos, D. Vrakas, F. Kokkoras, N. Bassiliades, and I. Vlahavas, "An ontology-based planning system for e-course generation," *Expert Syst. Appl.*, vol. 35, no. 1, pp. 398–406, 2008.

[74] E. Ministers of Education, *The Bologna Declaration of 19 June 1999*, no. May 1998. Bologna, 1999.

[75] G. Demartini, I. Enchev, J. Gapany, and P. Cudre-Mauroux, "The Bowlogna Ontology: Fostering Open Curricula and Agile Knowledge Bases for Europe's Higher Education Landscape," *Semant. Web*, vol. 4, no. 1, pp. 1–11, 2012.

[76] R. Mizoguchi, "Tutorial on ontological engineering Part 2: Ontology development, tools and languages," *New Gener. Comput.*, vol. 22, no. 1, pp. 61–96, 2004.

[77] BBC, "BBC Curriculum Ontology," 2013. [Online]. Available: https://www.bbc.co.uk/ontologies/curriculum.

[78] P. Brusilovsky and J. Vassileva, "Course sequencing techniques for large-scale web-based education," *Int. J. Contin. Eng. Educ. Life Long Learn.*, vol. 13, no. 1–2, pp. 75–94, 2003.

[79] S. Boyce and C. Pahl, "Developing Domain Ontologies for Course Content," *Educ. Technol. Soc.*, vol. 10, no. 3, pp. 275–288, 2007.

[80] R. Ramesh, M. Sasikumar, and S. Iyer, "Annotating the Domain ontology of a Course with its Syllabus and Learning Objectives," in *Learning and Teaching in Computing and Engineering (LaTICE), 2016 International Conference on*, 2016, pp. 130–131.

[81] A. Ameen, K. U. R. Khan, and B. P. Rani, "Creation of ontology in education domain," in *2012 IEEE Fourth International Conference onTechnology for Education (T4E)*, 2012, pp. 237–238.

[82] L. Zeng, T. Zhu, and X. Ding, "Study on Construction of University Course Ontology: Content, Method and Process," in *International Conference on Computational Intelligence and Software Engineering. CiSE 2009.*, 2009, pp. 1–4.

[83] N. F. Noy and D. L. McGuinness, "Ontology Development 101 : A Guide to Creating Your First Ontology," Stanford, 2000.

[84] M. Uschold and M. King, "Towards a Methodology for Building Ontologies," in *Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995, no. July, p. 15.

[85] R. Subhashini and J. Akilandeswari, "A Survey on Ontology Construction Methodologies," *Int. J. Enterp. Comput. Bus. Syst.*, vol. 1, no. 1, pp. 60–72, 2011.

[86] R. de A. Falbo, C. S. de Menezes, and A. R. C. da Rocha, "A Systematic Approach for Building Ontologies," in *Ibero-American Conference on Artificial Intelligence*, 1998, pp. 349–360.

[87] M. Gruninger and M. S. Fox, "Methodology for the Design and Evaluation of Ontologies," in *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, 1995, pp. 1–10.

[88] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: a search and metadata engine for the semantic web," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 2004, pp. 652–659.

[89] Y. Zhang, W. Vasconcelos, and D. Sleeman, "OntoSearch : An Ontology Search Engine 1 IKB : Identify Knowledge Base," in *Research and Development in Intelligent Systems XXI*, Springer, 2005, pp. 58–69.

[90] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle, "The SWRC Ontology - Semantic Web for Research Communities," *EPIA*, pp. 218–231, 2005.

[91] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle, "SWRC Ontology," 2005. [Online]. Available: http://ontoware.org/swrc.

[92] B. Declaration, "The Bologna Declaration of 19 June 1999," *Jt. Declar. Eur. Minist. Educ.*, 1999.

[93] J. Hefflin, J. Hendler, and S. Luke, "SHOE : A Prototype Language for the Semantic Web," *Heflin, Jeff, James Hendler, Sean Luke. "Shoe A prototype Lang. Semant. web." Linköping Electron. Artic. Comput. Inf. Sci.*, vol. 6, pp. 1–34, 2001.

[94] HQA, "HQA Accreditation Template," Athens, 2014.

[95] S. Draft, "Computer Science Curricula 2013," New York, NY, USA, 2013.

[96] D. Dicheva and C. Dichev, “Authoring educational topic maps: can we make it easier?,” in *Advanced Learning Technologies, 2005. ICALT 2005. Fifth IEEE International Conference on*, 2005, pp. 216–218.

[97] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe, S. Jupp, G. Moulton, R. Stevens, N. Drummond, S. Jupp, G. Moulton, and R. Stevens, “A Practical Guide to Building OWL Ontologies using Protege 4,” 2009.

[98] S. Draft, “Computer Science Curricula 2013.” ACM and IEEE Computer Society, Incorporated: New York, NY, USA, 2013.

[99] A. Gómez-Pérez, “Some Ideas and Examples to Evaluate Ontologies,” in *Proceedings. of 11th Conference on Artificial Intelligence for Applications*, 1995, pp. 299–305.

[100] M. Poveda-Villalón and M. C. Suárez-Figueroa, “OOPS - OntOlogy Pitfalls Scanner,” Madrid, 2012.

[101] D. Skuce, “Conventions for reaching agreement on shared ontologies,” in *Proceedings of the 9th Knowledge Acquisition for Knowledge Based Systems Workshop*, 1995.

[102] R. Falbo, G. Guizzardi, and K. Cristina Duarte, “An Ontological Approach to Domain Engineering,” in *14 International Conference on Software Engineering and Knowledge Engineering*, 2002, vol. 27, pp. 351–358.