Ontology and Query System Implementation of a Computer Science Program Using Grüninger and Fox’s Methodology

Oswaldo Jair García Franco, Erick Barrios González, Mireya Tovar Vidal, José de Jesús Lavalle Martínez, and Carmen Cerón Garnica

Faculty of Computer Science, Benemerita Universidad Autonoma de Puebla,
14 sur y Av. San Claudio, C.U., Puebla, 72570, Mexico

carmen.ceron@correo.buap.mx

Keywords: Ontology, Grüninger and Fox’s Methodology, OWL.

Abstract: This article shows the Grüninger and Fox’s methodology for the implementation of an ontology using OWL (The W3C Web Ontology Language). The ontology is created from a computer science program, it also shows some results obtained by a module created in python to add, delete, and consult information about students, courses and teachers using Owlready2 and RDFLib (packages for ontology-oriented programming in Python). The consistency of the ontology has been evaluated using the Protegé’s pellet reasoner and queries that answer specific competence questions defined in the design phase of the methodology. The results show the advantages of the ontology to manipulate and consult the information about the degree.

1 INTRODUCTION

The idea of adding semantic and ontological metadata to the network (semantic web) allows us to have more organized information, ensuring better searches. Ontologies define the basic terms and relationships that make up the vocabulary of a specific area, as well as the rules for combining terms and relationships to define extensions to the vocabulary (Neches, 1991), providing the means to explicitly describe the conceptualization behind the knowledge represented. That is why ontologies are intended to address a problem related to unorganized information. In the present document the design of an ontology is presented to organize information about courses, professors and students in a computer science degree. For the design of the ontology, the Grüninger and Fox’s methodology (Gruninger, 1995) helped us to define competency questions based on a scenario, and express them formally with its axioms, that is, the possible applications in which the ontology will be used. We have also relied on (Jean-Baptiste, 2021) for the creation of the ontology in OWL and the modules in Python. There are other works that have addressed similar problems. In (Doria, 2017) where the use of ontologies in the educational field is proposed. In (Mora Arciniega, 2016) an ontological model is developed for the representation of academic data and its publication with semantic technology and (Rosell León, 2016) where an UH-Ontology was designed for the management of heterogeneous data in universities. Our contribution is mainly focused on the management of information regarding undergraduate courses, it is hoped to provide an alternative to existing solutions. (Reyes Peña, 2018) implements an ontology for a master’s program in computer science, (Flores, 2017) where an ontological model is designed to represent the information about the professional practices or (Tovar, 2017) the social service of an educational institution, all using the methodology of (PyQt5, 2021) and SPARQL. We also have (Bravo, 2014) that performs the representation of the academic and
institutional context using ontologies but using SQWRL as a query language.

1.1 Motivation

The implementation of an ontology in Computer Science degree will allow the administrative staff to access efficiently to detailed information about students, courses and professors which allows to have a scalable system in the future to store more data related to the study program.

2 METHODOLOGY

The Grüninger and Fox’s methodology (Gruninger, 1995) was applied to the proposed problem. It is composed of the following steps: Motivating scenario, informal competency questions, terminology, competency questions using formal terminology and axioms.

2.1 Motivating Scenario

The Computer Science degree is made up of an educational program described by a study plan document and a graphic map that shows the information about the courses. The study plan document describes the classification of courses in different levels: The basic level is made up of the General University Training area (FGU) and two disciplinary areas that correspond to Computer Science. The training level is made up of 5 disciplinary areas that correspond to: Disciplinary integration, basic sciences, computer science and technology that make up the formation of the computer science degree. Finally, the optative area is made up of the disciplinary optative courses and the complementary optative courses, which includes a range of 54 to 90 hours and 3 to 6 credits. Although the study plan document contains a brief description of the courses, the graphic map details, their name, their code, what are their schedules, laboratory hours, workhours, theory hours, areas, semesters to which that belong, what is their required courses (or if they are a requirement to take another course) and the number of credits obtained with each one. Within the curricular structure, the area of disciplinary Integration promotes the relationship of theory with practice and is made up of two subareas: professional practice and integration courses. It also considers within the categories of optional courses both the disciplinary and the complementary. A regular, leveling or regularization student can take basic and/or optative courses, and they are limited to a maximum of two courses of four credits each one. Both students and teachers have a name, age, email, telephone. Teachers have a worker number, a date of the first contract and can teach one or more courses, finally students have a registration, a date of entry and take several courses depending on the current period they are studying.

2.2 Informal Competency Questions

From the proposed scenario, to define the actions the actions carried out in the degree and the elements that make it up, the following set of questions are obtained:

- What is the id, name, email, telephone of each student and teacher?
- What is the name, credits, area of each course?
- What is the schedule in which each course is taught?
- How many areas are there in the degree and how many credits is each course in each area worth?

In order to answer the behavior of the elements that make up the degree, the following set of informal competency questions are proposed:

- Which teachers teach a disciplinary optative course and what course do they teach?
- Which students take optative courses and what courses are?
- How many laboratory hours does each course in the general formation area have?
- What courses in the area of technology must a student take and how many credits does each have?
- What courses are formative and in what periods are they studied?

From the scenario and the competency questions, the classes (objects as mentioned in (Gruninger, 1995)), sub-classes, attributes are defined in the ontology design.

2.3 Terminology

2.3.1 Class Identification

The description of the classes found is presented in the Table 1, considering the competency questions.
Table 1: Class description.

<table>
<thead>
<tr>
<th>Class</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>Individual of the human species that includes in this case teachers and students</td>
</tr>
<tr>
<td>teacher</td>
<td>Person whose profession is to teach a science, in this case he teaches one or more courses of the degree in computer science.</td>
</tr>
<tr>
<td>student</td>
<td>Person who receives teachings from the teacher, is registered and takes one or more courses.</td>
</tr>
<tr>
<td>course</td>
<td>Course on a specific topic. Example: Differential calculus.</td>
</tr>
<tr>
<td>basic-course</td>
<td>It is a matter of general training, to ensure adequate mastery of the scientific instrument, of analysis in general and the fundamentals of methodologies of science and research in particular (Rivera, 2016).</td>
</tr>
<tr>
<td>formative-course</td>
<td>Its objective is to give the student deeper knowledge and specialization in the different areas of computer science (Rivera, 2016).</td>
</tr>
<tr>
<td>optative-course</td>
<td>It is a course whose objective is to deepen the student’s learning in the areas that have been of greatest interest to him (Rivera, 2016).</td>
</tr>
<tr>
<td>disciplinary-course</td>
<td>It is an optional course which covers the student’s graduation profile and is related to current, conceptual and procedural knowledge of the Bachelor’s Degree in Computer Science (Rivera, 2016).</td>
</tr>
<tr>
<td>desity-course</td>
<td>It is an optative disciplinary type of which only one course can be taken, and includes a range of 54 to 90 hours and 3 to 6 credits (Rivera, 2016).</td>
</tr>
<tr>
<td>complementary-course</td>
<td>It is an optional course whose objective is to offer the student the opportunity to deepen some of the areas of disciplinary knowledge (Rivera, 2016).</td>
</tr>
</tbody>
</table>

2.3.2 Defining Attributes

Based on the previously found classes and the scenario approach, Table 2 shows the description of the attributes or data property.

Table 2: Attributes description.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Class</th>
<th>Type</th>
<th>Range</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>person</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>age</td>
<td>person</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>email</td>
<td>person</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>telephone</td>
<td>person</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>id_teacher</td>
<td>teacher</td>
<td>String</td>
<td>8</td>
<td>1:1</td>
</tr>
<tr>
<td>admission_date</td>
<td>student</td>
<td>Date</td>
<td>Date</td>
<td>1:1</td>
</tr>
<tr>
<td>attended_semester</td>
<td>student</td>
<td>String</td>
<td>Positive Integer</td>
<td>1:1</td>
</tr>
<tr>
<td>student_id</td>
<td>student</td>
<td>String</td>
<td>8</td>
<td>1:1</td>
</tr>
<tr>
<td>schedule</td>
<td>course</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>credits</td>
<td>course</td>
<td>Integer</td>
<td>Positive Integer</td>
<td>1:1</td>
</tr>
<tr>
<td>semester</td>
<td>course</td>
<td>Integer</td>
<td>Positive Integer</td>
<td>1:1</td>
</tr>
<tr>
<td>theory_hours</td>
<td>course</td>
<td>Integer</td>
<td>Positive Integer</td>
<td>1:1</td>
</tr>
<tr>
<td>lab_hours</td>
<td>course</td>
<td>Integer</td>
<td>Positive Integer</td>
<td>1:1</td>
</tr>
<tr>
<td>work_hours</td>
<td>course</td>
<td>Integer</td>
<td>Positive Integer</td>
<td>1:1</td>
</tr>
<tr>
<td>id_course</td>
<td>course</td>
<td>String</td>
<td>4</td>
<td>1:1</td>
</tr>
<tr>
<td>name_course</td>
<td>course</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>area</td>
<td>course</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>optative_type</td>
<td>optative_course</td>
<td>String</td>
<td>Limited</td>
<td>1:1</td>
</tr>
<tr>
<td>is_desit</td>
<td>optative_course</td>
<td>boolean</td>
<td>1:1</td>
<td></td>
</tr>
</tbody>
</table>

2.3.3 Relationships Definitions Between Classes

In this section, we have defined the relationship between each class to establish a hierarchy in the ontology. Teachers and students belong to the class of person and course is divided into two different types: basic and formative; for the formative courses there are a subclass “Optional”, this subclass has the attribute “optative type” as well as for optional and disciplinary courses. In Table 3 we can see the relationships between classes or object property of the ontology. For example, a course is taught by a teacher, the cardinality is 1 to 1.
### 2.4 Competency Questions Using Formal Terminology

The formal competency questions place restrictions on which axioms will be included.

1. Which teachers teach a disciplinary optative course and what course do they teach?

\[ \exists x \text{ teacher}, n \text{ course} \ (\text{taught_by}(\text{course}, \text{teacher})) \]

2. Which students take optative courses and what courses do they take?

\[ \exists x \text{ student}, n \text{ course} \ (\text{taken_by}(\text{course}, \text{student})) \]

3. How many laboratory hours does each course in the general formation area have?

\[ \exists l \text{ hours}, c \text{ course} \ (\text{lab_hours}(l, c)) \]

### 2.5 Axioms

Once the elements within the ontology have been defined, we proceed to define the axioms:

- A student cannot be a teacher at the same time.
  \[ \neg (\exists x \text{ student}, y \text{ teacher}) \]

- A student can only take a DESIT course.
  \[ \neg (\exists o \text{ basic course}, (\text{optative_course}(o) \land \text{basic_course}(b))) \]

- A optative course cannot be a basic course.
  \[ \neg (\exists s \text{ course}, (\text{is_desit}(s) = \text{True} = 1)) \]

- Optative courses are limited to 54 to 90 hours
  \[ \text{optative_course}(o) \land ((\text{theory_hours}(o, t) + \text{lab_hours}(o, l) + \text{work_hours}(o, w)) > 54 \land ((\text{theory_hours}(o, t) + \text{lab_hours}(o, l) + \text{work_hours}(o, w)) < 90)) \]

### 3 RESULTS AND DISCUSSION

The hierarchy of classes to represent the course, courses, students and teachers within the Computer Science degree is shown in Figure 1.
3.1 Implementation of the Ontology Systems

Equations should be placed on a separate line, numbered and centered. An extra line space should be added above and below the equation. Classes, attributes and relationships have been implemented using (Jean-Baptiste, 2021), functions have been created to insert and delete information from forms using a graphical interface designed with Pyqt5 (PyQt5, 2021).

The implementation of the queries uses two libraries to complement each other, in this case Owlready (Baptiste, 2019) does not support certain SparQL elements:
- ASK, DESCRIBE, LOAD, ADD, MOVE, COPY, CLEAR, DROP, CONSTRUCT queries.
- INSERT DATA, DELETE DATA, DELETE WHERE queries (you may use INSERT or DELETE instead).
- SERVICE (Federated queries)
- VALUES in SELECT queries
- Parentheses in property path expressions, e.g. ‘(a/rdfs:subClassOf)*’

When a query contains any element listed, RDRlib executes the query avoiding any kind of issues.

3.2 Answers to Competition Questions

To show that the ontology and the Python module can solve the competency questions, the ontology has been filled with example instances containing: 15 course instances or their subclasses (formative and basic), 2 teacher instances, and 2 student instances.

To answer the questions previously posed, it is necessary in the application interface to have a section to carry out the corresponding queries, the queries are made in SPARQL in the query section and the result can be seen in the box below.

The consistency of the ontology was tested before its implementation in the system through Protegé’s Pellet reasoner (Protégé, 2021) as it is shown in Figure 3.

4 CONCLUSIONS

The Grüninger and Fox’s methodology (Gruninger, 1995) allowed us to establish a structured semantic relation to define the classes, properties, relationships and the restrictions necessary for the implementation of the ontology that was evaluated through a consistency test using the Protegé’s Pellet module (Protégé, 2021) and the answers to the competence questions mentioned above in the methodology section. The use of two libraries Owlready2 (Jean-Baptiste, 2021) and RDFLib (RDFLib Team, 2019) turned out to be of great importance to obtain results of more complex queries from the ontology, since individually each one of the already mentioned libraries has important limitations, also contemplating that the performance of the application is limited to the
performance of the libraries. In a future work it is intended to use the results obtained in this paper to include postgraduate courses information, in addition to extending the system to improve the management of student enrolment. Finally, Python as the selected programming language streamlined the implementation for each of the ontology modules as mentioned in (Jean-Baptiste, 2021).

Figure 4: Answer to competence question 1.

Figure 5: Answer to competence question 2.

ACKNOWLEDGEMENTS

The authors would like to thank Benemerita Universidad Autonoma de Puebla, Mexico. The present work has been funded by the research projects VIEP-2021 and VIEP-2022 at BUAP.

REFERENCES


PyQt5 Reference Guide — PyQt 5.7 Reference Guide. (s. f.). https://doc.bccnsoft.com/docs/PyQt5/


