A case study of modeling the object oriented programming knowledge as an educational ontology

Mihaela Oprea¹

(1) Petroleum-Gas University of Ploiesti, Department of Automatics, Computers and Electronics, Bdul Bucuresti No 39, Ploiesti, 100680, ROMANIA
E-mail: mihaela[at]upg-ploiesti.ro

Abstract

Educational ontologies represent important resources for intelligent tutoring systems, e-learning systems and web-based education in universities. They can be developed by modeling the knowledge domain of each university course for all didactical activities: teaching, learning and examination. The paper presents a case study of modeling the object oriented programming knowledge as an ontology by following methodological guidelines of the EduOntoFrame framework. The resulted educational ontology, CS-OOP-Onto, was implemented in Protégé, a Java-based ontology editor. Basic concepts as well as advanced concepts of object oriented programming were included in CS-OOP-Onto, facilitating teaching and learning the Object Oriented Programming course at different levels of knowledge. The modular structure of the ontology allows its extensibility with specific knowledge from different object oriented programming languages such as C++, C# and Java.

Keywords: Educational ontology, Knowledge modeling, University course, Object oriented programming

1 Introduction

Intelligent tutoring systems as well as e-learning systems and web-based education in universities make use of various educational resources such as educational ontologies as means for knowledge sharing and interoperability. A university educational ontology represents a conceptualization of a university course. Such an ontology is composed by a vocabulary of terms from the university course, representing the course domain terminology, with terms definitions, and by a set of axioms with rules and constraints regarding the correct use of terms in various contexts. The development of an educational ontology follows either an existing methodology or an ad hoc chosen one. Usually, a collaborative ontology development methodology is applied, based on the cooperation of course domain experts (e.g. teachers) and knowledge (ontology) engineer.

The paper presents a case study on modeling the domain knowledge of the university course Object Oriented Programming from the Computer Science undergraduate study program as an ontology, by following the general methodological guidelines of the EduOntoFrame framework introduced in (Oprea, 2013) and the specific ones, described in (Oprea, 2015), for collaborative educational ontology development.

The paper is organized as follows. Section 2 presents a brief overview on some ontology development methodologies and frameworks, focusing on those applied to university educational ontologies development. Also, some important issues about modeling a domain knowledge as an ontology are discussed, with a special emphasize on university courses domains. A case study of object oriented programming knowledge modeling as an ontology (namely, CS-OOP-Onto) is described in section 3. The CS-OOP-Onto ontology was implemented in Protégé, a Java-based ontology editor. The final section concludes the paper.
2 Educational Ontology Development

2.1 Methodologies and Frameworks
Several methodologies and frameworks were proposed for ontology development, some of them being applied to educational ontologies development in universities. Examples are given by DILIGENT (Pinto et al., 2004), HCOME (Kotis and Vouros, 2006), UPON (De Nicola et al., 2009), UPON Lite (De Nicola and Missikoff, 2016), NeOn (Suárez-Figueroa et al., 2012), EduOntoFrame (Oprea, 2013; Oprea, 2015). A brief overview on these frameworks and methodologies is presented.

The DILIGENT methodology provides support to domain experts in a distributed setting to engineer and evolve ontologies. A fine-grained methodology approach based on Rhetorical Structure Theory for ontology engineering argumentation is used. Five activities are performed: (1) initial ontology building, (2) local adaptation of the ontology, (3) local ontology analysis, (4) local ontology revision, and (5) local ontology update. Two case studies are presented in (Pinto et al., 2004). One from the Biology domain and one in a Computer Science Department. DILIGENT is a collaborative methodology that was applied to educational ontologies development in universities.

HCOME (Kotis and Vouros, 2006) is a human centered ontology engineering methodology that was developed from a domain-specific perspective and has some features similar to DILIGENT. It can be applied to collaborative development of academic educational ontologies.

UPON (Unified Process for Ontology) is a reference ontology engineering methodology proposed in (De Nicola et al., 2009). It is based on the software engineering unified process and the UML approach, and was applied to large-scale ontology building in domains such as automotive, aerospace and furniture. The UPON Lite methodology (De Nicola and Missikoff, 2016) is the lightweight version of UPON, released in 2016, for rapid ontology engineering, based on semantic technology. It was designed for enterprise ontologies development.

The NeOn methodology, which is detailed in (Suárez-Figueroa et al., 2012), allows collaborative development of ontologies. It uses the NeOn Glossary for Processes and Activities and is based on nine scenarios for ontologies building (a process viewed as the construction of an ontologies network, where different people can manage different resources). The NeOn methodology performs ontology engineering by reuse. Also, it is applied a merging process for the ontological resources. The methodology was applied to educational ontologies development.

EduOntoFrame is a general framework (described in (Oprea, 2013)) designed for the development of educational ontologies in a university. It is applied in all phases of a full university course didactical activity: teaching, learning and examination. Three main ontologies compose the course educational ontology: the course teaching ontology (Teach-Onto), the course learning ontology (Learn-Onto) and the course examination ontology (Exam-Onto). Each of these three ontologies has one or more course dependent ontologies and one course independent ontology. The independent ontologies contain terms corresponding to the didactical activity (e.g. teaching, learning, examination), including terms specific to pedagogical, methodological and psychological knowledge, that are characteristic to a university didactical activity. A collaborative version of EduOntoFrame was introduced in (Oprea, 2015) which makes use of ontology mapping. Our research work reported in this paper applies the methodological guidelines of EduOntoFrame.

2.2 Domain Knowledge Modeling as an Ontology
Modeling the knowledge of a certain expertise domain can be realized by identifying the fundamental (basic) knowledge and more advanced knowledge from that domain, and their corresponding concepts (basic and advanced), and relationships, followed by their inclusion in an ontology. These steps are detailed as follows.
Algorithm 1: Expertise Domain Knowledge Modeling

Input: expertise domain knowledge (exp-DK)
Output: ontology (D-Onto)

Step 1. identify and define the fundamental (basic) knowledge (B-DK) from exp-DK and their corresponding concepts (B-DK-Concepts) and relationships (B-DK-Rel);

Step 2. identify and define the more advanced knowledge (A-DK) from exp-DK and their corresponding concepts (A-DK-Concepts) and relationships (A-DK-Rel);

Step 3. include B-DK-Concepts, B-DK-Rel, A-DK-Concepts and A-DK-Rel in an ontology, D-Onto (i.e. in the ontology hierarchical structure – ontology tree);

Step 4. define the set of axioms (AS) with rules and constraints corresponding to the correct use of the concepts included in D-Onto;

Step 5. return D-Onto ontology.

The formal definition of D-Onto is given by relation (1).

\[ D-Onto = \{D-CR^{exp-DK}, AS\} \]  

where, \( D-CR^{exp-DK} \) given by relation (2), is the set of concepts (basic and advanced) and their corresponding relationships from the expertise knowledge domain exp-DK, and AS is the axioms set.

\[ D-CR^{exp-DK} = B-DK-Concepts \cup B-DK-Rel \cup A-DK-Concepts \cup A-DK-Rel \]  

We have applied the main steps of Algorithm 1 to university course knowledge modeling as an educational ontology. The case study is described in the next section. Examples of relationships between concepts are: is-a, ako, has, part-of, required-by, belongs-to etc. The D-CR\(^{exp-DK}\) set (given by relation (1)) is extended with pedagogical, methodological and psychological terms (\( Terms^{PMP} \)), specific to university course didactical activities of teaching, learning and examination. In this case, the formal definition of D-CR\(^{exp-DK}\) (named, D-CR\(^{UnivC-DK}\)) is given by relation (3).

\[ D-CR^{UnivC-DK} = B-DK-Concepts \cup B-DK-Rel \cup A-DK-Concepts \cup A-DK-Rel \cup Terms^{PMP} \]  

Figure 1 shows the general block diagram of the expertise domain knowledge modeling as an ontology with our approach, while Figure 2 shows the modular structure of UnivCD-Onto, i.e. the D-Onto ontology in the case of a university course domain knowledge modeling with EduOntoFrame framework.
The UnivCD-Onto ontology contains three ontologies, Teach-Onto, corresponding to the university course teaching activity, Learn-Onto, corresponding to the university course learning activity and Exam-Onto, corresponding to the university course examination activity. According to EduOntoFrame, each ontology is composed by one or more modules (i.e. ontologies) as follows. Teach-Onto contains four modules: CBS-Onto (course basic subject ontology corresponding to university course basic knowledge modeling), CAS-Onto (course advanced subject ontology, corresponding to university course advanced knowledge modeling), CPS-Onto (course prerequisite subject ontology corresponding to university prerequisite courses knowledge modeling), and BT-Onto (basic teaching ontology). Learn-Onto contains two modules: CPA-Onto (course practical applications ontology) and BL-Onto (basic learning ontology). Exam-Onto contains two modules: CE-Onto (course examination ontology), and BE-Onto (basic examination ontology). One of the main advantages of using EduOntoFrame is given by the modularization of D-Onto, that allows its extensibility and easy update.

Summarizing, the domain knowledge modeling in the case of a university course is performed during EduOntoFrame framework application for collaborative educational ontology development. The inclusion of pedagogical, methodological and psychological terms in D-Onto is a step specific to EduOntoFrame. Our approach allows the collaboration between knowledge engineer and domain experts (i.e. teachers). Examples of other research work in the area which consider the collaborative educational ontologies development by modeling the course domain knowledge are reported in (Boyce and Pahl, 2007), (Panagiotopoulos et al., 2012) and (Kouneli et al., 2012).
3 Case Study

We have considered for the case study of domain knowledge modeling as an ontology the expertise domain of the university course Object Oriented Programming (OOP) from the Computer Science (CS) undergraduate study program, that was modeled as the CS-OOP-Onto ontology.

Following Algorithm 1 and the EduOntoFrame framework, we have derived the basic and advanced concepts from the OOP domain, as well as their relations. Examples of basic concepts are: class, object, class member, class interface, class implementation, constructor, destructor, class data, class method, message, abstraction, encapsulation, information hiding, class section, public, private, protected etc. The basic concepts represent the OOP fundamental knowledge. Examples of advanced concepts are virtual function, polymorphism, overloading, overriding, inheritance, single inheritance, multiple inheritance, templates, virtual class, abstract class. Figure 3 shows a part of the basic and advanced concepts hierarchy of OOP. The default relations are is-a and ako, in the case of concepts taxonomy (e.g. the relation type between ClassMethod and DataMember with ClassMember is is-a) and has, part-of, belongs-to in the case of ontology component modules (e.g. the relation type between OOP-BasicConcepts and OOP-AdvancedConcepts with CS-OOP-OntoRoot is part-of).

The general OOP principles were taken from the OOP textbook (Budd, 1998). The knowledge specific to a certain object oriented programming language (e.g. C++, C#, Java) can be integrated in CS-OOP-Onto, by extension.

The implementation of the CS-OOP-Onto was done in Protégé 4.3. Figure 4 shows some data properties from the data property hierarchy of the ontology, while some object properties from the object property hierarchy of the ontology are shown in Figure 5. Examples of such entities are className (data property) and hasClassMember (object property). Figure 6 shows a part of the CS-OOP-Onto ontology class hierarchy in Protégé 4.3 with selected basic and advanced concepts.

![Figure 3. The OOP basic and advanced concepts hierarchy (selection)](image-url)
Figure 4. Data property hierarchy for CS-OOP-Onto (selection from Protégé 4.3)

Figure 5. Object property hierarchy for CS-OOP-Onto (selection from Protégé 4.3)

Figure 6. The CS-OOP-Onto class hierarchy (selection from Protégé 4.3)
4 Conclusion

The paper presented a case study of modeling the knowledge from the Object Oriented Programming university course as a university educational ontology (CS-OOP-Onto) by following the EduOntoFrame framework and Algorithm 1 that was described in section 2. The main benefit of using the developed educational ontology (which models the OOP knowledge) is given by its modular structure which facilitates extensibility and easy update, being a proper educational resource for web-based education, e-learning and intelligent tutoring systems implemented in universities.

References