An Ontological Approach in Learning Programming Languages

The case of Java and C

Christos Pierrakeas[1,2], Georgia Solomou[1], Achilles Kameas[1]

[1]Educational Content, Methodology and Technology Laboratory (e-CoMeT Lab)
Hellenic Open University (HOU)
Patras, Greece

[2]Dept. of Computer Science in Administration and Economy
Technological Educational Institute (TEI) of Patras
Patras, Greece

{pierrakeas, kameas}@eap.gr

***Abstract*— Learning Objects (LOs) constitute a novel approach in the organization of educational content and their usage continuously gains ground in the field of distance education. On the other hand, ontologies are rich knowledge representation structures that can be utilized for modeling many aspects of learning. In this paper, we examine the application of ontologies for modeling the knowledge domain covered by a distance learning course and the exploitation of such a representation for discovering correlations among LOs and effectively combining them when designing a course. We then present how we can take advantage of the ontological models for two very popular programming languages – that is Java and C – in an attempt to provide tutors with the possibility to better organize educational material regarding the field of computer programming.**

ontologies; learning objects; distance education; programming languages; Java; C;

I. INTRODUCTION

In distance education knowledge is delivered to learners from distance, by the aid of learning management systems or other similar in scope applications and mechanisms. The extension of these systems with semantic-aware technologies could redound to both learners' and tutors advantage, by providing them with intelligent applications and tools.

More specifically, ontologies appear as a very prominent knowledge representation technique in the field of education. They are considered a means to facilitate knowledge reuse and sharing across applications and groups of people and according literature they have been used for representing knowledge domains, teaching strategies, student profiles, as well as for modeling competence and learning goals [2]. In all these cases, they render e-learning applications capable of processing semantics, understanding and combining information and producing new facts. Therefore, ontologies can be exploited by e-learning systems to make them "smarter".

Through this work, we make an attempt to create an ontology for representing the knowledge delivered by a distance learning course, offered by the Hellenic Open University. We opted for the knowledge domain of computer programming, given that the learning of at least one programming language is the common requisite by all study programs in Informatics. We focused on Java and C, because these two languages constitute representative examples in the field of object-oriented and procedural programming, respectively.

The purpose of such an effort is the subsequent utilization of the produced ontological model by e-learning applications and hence the creation of advanced services for both learners and tutors. Our particular aim is to help tutors in designing and organizing the LOs for their course, as well as learners by giving them the opportunity to interact with intelligent learning mechanisms and thus becoming able to better understand and perceive knowledge regarding the field of computer programming.

The rest of the paper is organized as follows: In section II we define both the meaning of an ontology in computer science and of a LO in distance education. Section III researches the correlation of a knowledge domain ontology with LOs. In section IV we give an overview of two implemented ontologies about Java and C and present the possible benefits of using them when designing a course about computer programming. Conclusions follow, in Section V.

II. BACKGROUND

In this section we clarify the meaning of an ontology, as used in computer and information science, and outline its structure. We then figure out the most important characteristics of a LO in the context of distance education and assign to it a more explicit definition.

A. About Ontologies

Gruber in [3] defines an ontology as "*an explicit specification of a shared conceptualization*" and outlines its utilization as a means to represent a specific domain of knowledge or discourse in a more typical way. The World Wide Web Consortium[1] (W3C) uses the term 'ontology' in order to describe various structures of interrelated concepts, from simple taxonomies to metadata schemes and logical theories. In particular, according to W3C "*An ontology defines the terms used to describe and represent an area of knowledge. <…> Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them <…>. They encode knowledge in*

[1] http://www.w3.org/

a domain and also knowledge that spans domains. In this way, they make that knowledge reusable." [5].

Normally, a typical ontological representation consists of a) c*lasses*, which correspond to the knowledge domain's concepts, b) *properties*, which express types of interactions among the domain concepts and are further divided into *object properties* (connect two entities) and *datatype properties* (assign a value to an entity), and c) *instances* or *individuals* representing specific entities and belonging to some of the ontology classes. Assertions about true facts are stated through ontology *axioms*, whereas with *functions* special kinds of relations among individuals may be assigned.

Therefore, ontologies conceptualize and define the knowledge of a particular domain through commonly acceptable terms, typical axioms and constraints, capturing in this way the domain's semantics. They constitute a formalism for perceiving and processing information, sharing knowledge, allowing its reuse and thus enabling communication between heterogeneous and distributed systems.

B. *Learning Objects and their characteristics*

LOs constitute a novel approach in organizing educational content, which is found in the core of a whole new instructional design paradigm developed in the field of distance learning [7]. The main idea is to decompose the educational content into smaller chunks and construct self-contained learning units (i.e. LOs). These learning units can then be combined in almost infinite ways in order to create collections and build sections, lessons, or courses.

LOs can be reused in different educational contexts whereas the order in which they are presented to the learner (i.e. the learning path), can vary depending on the learner's needs. Additionally, a LO-based approach gives the ability to update learning content more quickly.

Despite the continuously increasing use of LOs in the field of distance learning, there is no common definition of them. A LO may range "*from anything to everything*" [7], which hinders the construction of sharable, interoperable and reusable LOs.

A common and widely acceptable definition, though, is necessary, as it would set a framework for LO's structure and provide a basis for clarifying some of its important characteristics, such as size, content and relationship with learning outcomes and the concepts of a knowledge domain.

Considering the literature, as well as the LO's functional requirements for accessibility, reusability and interoperability, summarized in [11], we proposed the following definition: *"A LO is a self-contained and independent unit of digital educational content, which is associated with one or more learning outcomes and it has as primary aim the ability of reuse in different educational contexts"* [9].

The above definition dictates that a LO should have educational content, it should be associated with learning outcomes and needs to be described by an appropriate set of metadata. The association of a LO with the expected outcomes of the learning process implies also a correlation among LOs and the knowledge domain concepts, as explained in the next section. This kind of interrelationship can be proved particularly useful when designing and developing courses that are based on LOs.

III. The Role of Ontologies

Because ontologies provide advanced possibilities in knowledge modeling, their proper utilization by intelligent mechanisms in the field of distance education could provide tutors with a useful guide during the course design process. A key aspect in the process of designing a distance learning course is the proper organization of its educational material. The better the educational material is organized and delivered to learners, the more effectual the learning process becomes.

As already mentioned, a prominent approach for handling the educational content of a course is its methodical arrangement into LOs. A LO is constructed so as to satisfy the learning process's expected outcomes and typically refers to just a small piece of knowledge, compared to the overall knowledge domain that is being covered by a course. This piece of knowledge that a LO deals with, is referred to as the LO's *subject*.

Usually the subject of an LO is described with a set of keywords, derived from its knowledge domain. These keywords can be used as search terms and thus lead to LO's discovery by the learner or the application.

We propose to select the keywords used for characterizing a LO's subject from an ontology modeling the concepts of the knowledge domain, as shown in Fig. 1. Then a richer network of interrelations among LOs would be produced, as a result of the existing ontological interrelations among the knowledge domain concepts. What is more, the LOs' easier discovery and retrieval would become possible, given that the semantics of each term, used to express a LO's subject, would be taken into account during the search process.

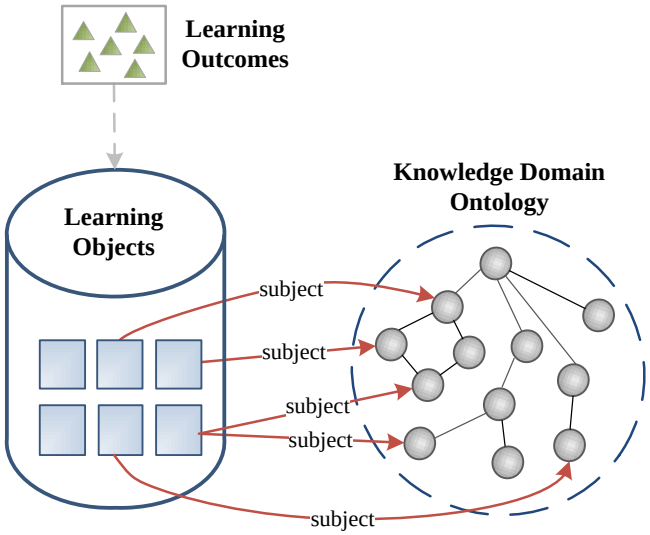

Figure 1. Connection of LOs with the knowledge domain concepts

Apart from the already known advantages of using ontologies in the field of education, mentioned in [8], their particular utilization for modeling a knowledge domain in the context of a distance learning course can have additional benefits, which can be summarized as follows:

- Having a sound and complete representation of a knowledge domain, the tutor can have a clear picture of the exact knowledge being covered by the educational material. Then the tutor could detect possible deficiencies of the available material and could also discover the additional LOs that should be developed in order to fully satisfy the learning outcomes of the course.
- By using an ontology for representing a knowledge domain, all declared terms come with an explicit meaning, whereas all types of interdependencies among them are expressed via appropriate properties. This information may constitute a valuable guide for the tutor in the process of defining the sequencing of concepts that are going to be treated by a course.
- By correlating a LO with specific concepts of the domain ontology - through the *subject* property - interrelationships among the course's LOs may be inferred. Such an effect can provide tutors with the possibility to discover more effective learning paths that would better correspond to learner's individual needs.
- Because LOs are explicitly characterized via ontology concepts instead of mere text keywords, their reusability in different educational contexts becomes possible.

IV. A Case Study on Java and C

In this section, we briefly describe the ontologies we deployed so as to support the learning process of the C and Java programming languages in the Hellenic Open University. Our main goal is to illustrate how these two ontological representations can be combined together in an attempt to achieve a better organization of LOs that have been designed for learning Java and C, as well as discover cases of potential reusability.

A widely-adopted methodology, proposed in [10], was utilized for building our ontologies. This methodology constitutes the basis for any other ontology building process and indicates first the collection of all necessary knowledge, enumerating as well the terms related to the domain of interest, and then the gradual creation of the class hierarchy, followed by the definition of necessary properties and restrictions.

For typically representing our ontological models, we adopted the most recent version of the Web Ontology Language which is a W3C standard, namely OWL 2.

A. The knowledge domain ontologies

As a first step for modeling an ontology of programming language, we had to extract all knowledge from a formal guide and build a glossary of terms. The book entitled '*The C Programming Language*' [6], constituted for us a complete guide to highlight and figure out the main concepts of ANSI standard C programming language. In the case of Java, we used its formal guide, known as '*The Java Tutorial*', provided by the Oracle Corporation[2].

1) An ontology for C

As far as the C programming language is concerned, the knowledge extraction process resulted in a glossary of terms, a part of which is presented in Table I.

This list was disintegrated and analyzed in order to build the class hierarchy. Some of the terms, depending on their meaning, were converted to either classes or sub-classes in the ontology schema. The rest of them were placed as members of the constructed classes, accordingly. The two upper level classes in the C ontology are *Keyword* and *LanguageElement*. Under the *LanguageElement* class lie the core elements of the C language, captured by the following classes: *Array*, *DataType*, *Function*, *Operator*, *Statement*, *Structure*, *Union* and *Variable* (see Fig. 2).

TABLE I. Glossary Of Terms for C Programmng Language

Data Type	*Loop*	*Switch*	*Do-while loop*	*Constant*
Integer	*Malloc*	*Input*	*While Loop*	*File*
Character	*For loop*	*Scanf*	*Logical Operator*	*Glossary*
Float	*Operator*	*Output*	*OR Operator*	*Syntax*
Data Structure	*Statement*	*Printf*	*AND Operator*	*String*
Variable	*If-else*	*Function*	*Relational Operator*	
Define	*If*	*Pointer*	*Standard Function*	

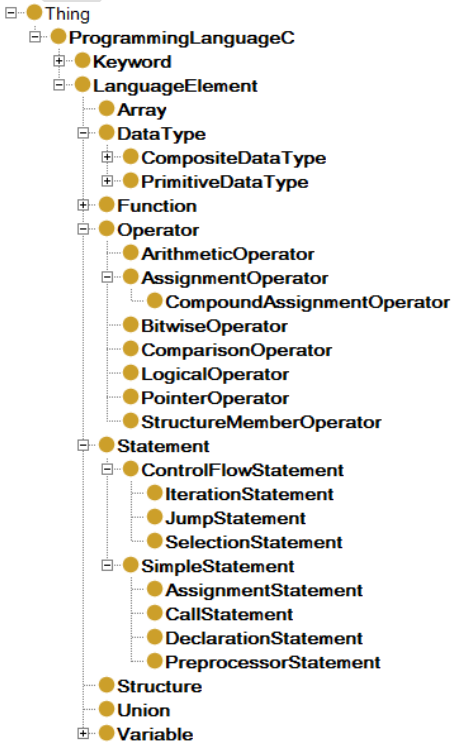

Figure 2. The class hierarchy of the C ontology

[2] http://docs.oracle.com/javase/tutorial/

Operators are a structural element typically supported by any programming language. In our ontology for C, we make provision for operators with the *Operator* class, which is comprised of the following sub-classes: *ArithmeticOperator*, *AssignmentOperator*, *BitwiseOperator*, *Comparison-Operator*, *LogicalOperator*, *PointerOperator* and *StructureMemberOperator*.

Operators participate in the creation of statements, which are the building blocks of a program code. A statement is an expression that can be executed so that a specific action (e.g. an assignment or a flow control) may take place. The *Statement* class in our C ontology represents exactly such kind of expressions in a program and is further divided into *SimpleStatement* and *ControlFlowStatement*.

The *SimpleStatement* class consists of the *AssignmentStatement*, representing statements in C that carry out computation and assign values to variables, the *DeclarationStatement*, used to declare variables and types, *CallStatement*, calling the execution of a function, with or without arguments, and the *PreprocessorStatement*, which express a kind of statements declared by the symbol "#" at the beginning of the source file and handled by the compiler before the program is actually compiled.

The *ControlFlowStatement* class was created to represent those statements that are only executed in case several conditions are met (sub-class *BranchingStatement*) as well as statements indicating repetition of a block of code for zero or more times, according a specific condition (sub-class *LoopStatement*).

The object properties in the C ontology are summarized in Table II. They express relationships among concepts of the C language as indicated by the Domain and Range restrictions.

TABLE II. OBJECT PROPRTIES IN THE C ONTOLOGY

Property Name	Domain	Range
hasFunctions	*LanguageElement*	*StandardLibraryFunction*
hasOperator	*LanguageElement*	*Operator*
uses	*Alphabet*	*Variable*

2) An ontology for Java

In our proposed ontological model about Java, the three top concepts, derived from the conceptual analysis, are captured by the following classes: i) *JavaElement*, ii) *Keyword* and iii) *LiteralValue*. The notion of control flow statements and operators, mentioned earlier in the context of the C ontology, constitute part of the Java language as well. So, the corresponding classes *ControlFlowStatement* and *Operator* have been created and placed under *JavaElement* (see Fig 3). What differs, though, is their division into sub-classes, given that Java and C don't support the same sub-categories for these structures.

More specifically, in the Java ontology, the *ControlFlowStatement* class apart from *Branching-* and *Loop- Statement* includes also *Exception-*, *Nested-* and *Selection- Statement*. As far as operators are concerned, although some are common for both languages (e.g., arithmetic and comparison operators), for the rest of them a different categorization is followed, hence the necessary classes have been created.

Since Java contains four types of variables, i.e. *instance variables*, *class variables*, *local variables* and *parameters*, we have made provision for respective classes in our model (*ClassVariable*, *InstanceVariable*, *LocalVariable* and *Parameter*). These classes have been set as children of the top *Variable* class.

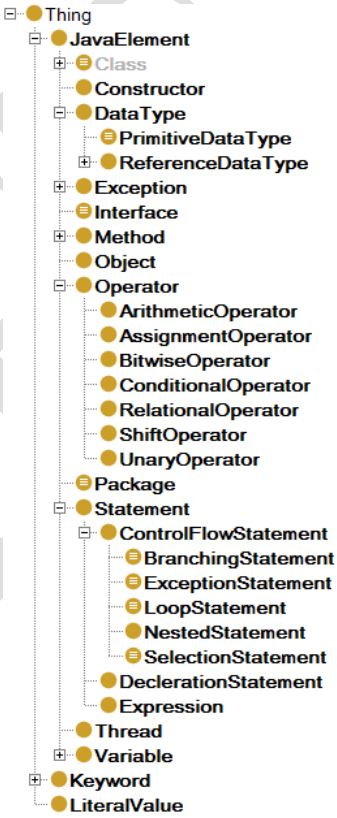

Figure 3. The class hierarchy of the Java ontology.

Java methods are captured by the *Method* class, which is further divided into the following subclasses: *AbstractMethod*, *FinalMethod*, *ClassMethod* and *InstanceMethod*. All object-oriented concepts, such as *object*, *constructor*, *interface*, *class* and *packages* are also present in our ontology.

The *Exception* class expresses problematic and erroneous situations in a Java program. All exceptions that cannot be caught are considered members of the *UncheckedException* class. This is further divided into *RunTimeException* (exceptions internal to the application) and *Error* classes (external error cases). Any other exception belongs to the *CheckedException* class which is sibling of *UncheckedException*.

Finally, one of the top classes in the Java ontology is *Keyword*, containing reserved words with very specific use within a program. Those keywords can be visibility modifiers (*Modifier* class) or even terms that control the

access privileges of various Java elements (*AccessControlModifier* class). *LiteralValue*, another top class, contains literals with special meaning for the Java compiler.

Finally, the types of relationships among Java elements are modeled by a set of object properties, summarized in Table III.

TABLE III. OBJECT PROPRTIES IN THE JAVA ONTOLOGY

Property Name	Domain	Range	Inverse Property
isSubClassOf	*Class*	*Class*	*isSuperClassOf*
isSuperClassOf	*Class*	*Class*	*isSubClassOf*
instance	*Object*	*Class*	-
hasMember	*Package or Interface or Class or Method*	*JavaElement*	*isMemberOf*
isMember	*JavaElement*	*Package or Interface or Class or Method*	*hasMember*
isDefinedByKeyword	*JavaElement*	*Keyword*	-
hasModifier	*JavaElement*	*Modifier*	-

B. Retrieving and Reusing LOs

In this section, we show how we can take advantage of the implemented ontological models about Java and C in the context of e-learning applications, able to handle ontologies. To this end, we make use of Protégé and combine the two proposed ontologies in one integrated schema. This schema is then augmented by an additional class, named *LearningObject*, which captures the notion of LOs. The object property *subject* was used for expressing the main subject to which a LO refers. So, all members of the *LearningObject* class are connected to an appropriate concept of the Java or C ontology, via this property.

As a starting point, we populate the combined ontology with real individuals and in particular with LOs regarding the course of Java. We, then, check if any LO exists, dealing with the Java operators or Java control flow statements. Our aim is first to discover and then reuse the retrieved LOs in different instructional context, e.g. in the process of learning operators in the C programming language.

In order to retrieve LOs that refer to all type of control flow statements in Java, including branching and loop statements, we need to run the following query:

subject some **Java:ControlFlowStatement**[3]

The results we managed to retrieve though this semantic search process were *LO_Java_3* and *LO_Java_2*, as shown in Fig. 4.

By checking the subject of obtained individuals, though, it reveals that none of them explicitly refers to Java control flow statements but to narrower in meaning concepts. More specifically, *LO_Java_3* refers to the branching statement '*if*' and *LO_Java_2* discusses the loop statement '*for*'. Nevertheless, the retrieval of these particular individuals becomes possible because in the context of our ontological model both branching and loop statements have been declared as sub-classes of the *ControlFlowStatement* class. But if it was not for these ontological representations and a simple text search had been performed instead, our query would return nothing, given that there is no LO making explicit reference to control flow statements.

Moving a step further, we may consider reusing retrieved LOs concerning control flow statements in Java, in the process of designing a course about a similar topic in C. By this way, a network of interrelationships among LOs could be build, allowing their utilization even in different educational contexts.

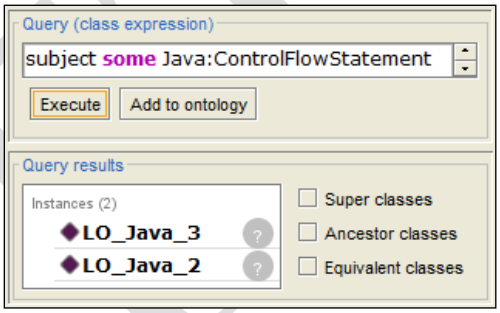

Figure 4. Query for *Control Flow Statements* in Java and retrieved results

V. CONCLUSIONS

The main purpose of this work was to propose a novel approach in building a course that could alleviate the task of designing the learning process and probably lead to more effective learning paths. This approach is based on the notion of a) *ontologies*, used to represent the network of concepts and relationships of the knowledge domain being covered by the course and b) *LOs*, used as a mean to organize the educational material into self-contained learning units, which are directly correlated to ontology concepts.

To discover the possible merits of such an approach, we first created the ontological representation of the Java and C programming languages, two domains of knowledge that have a lot in common and are offered as part of two distance learning courses in the Hellenic Open University. The process of building an ontology itself, is a demanding task, and although many methodologies have been proposed in literature - even for exactly creating a teaching ontology (e.g., in [3]) - we opted for the well-known methodology of Noy and McGuinness [10], which actually works as a basis for any other such ontology development process.

The resulted ontologies for the knowledge domains of Java and C provide tutors with a clearer picture about the domain concepts and their relationships. Therefore, by organizing the educational material into LOs and by correlating the LOs' subject with concepts in these ontologies, tutors can more easily discover cases where the educational material is problematic, by making limited or no reference to some concepts of great importance. On the other hand, because Java and C are two programming languages

[3] This is a semantic query, expressed in the Manchester OWL Syntax (http://www.co-ode.org/resources/ reference/Manchester _syntax/) via the DL-query tab of Protégé.

with many concepts in common (like for example *operators* and some kind of *statements*) LOs intended to serve the scope of the one course, can be probably reused in the context of the other. In addition, by connecting LOs with a domain ontology, the rich network of interrelations among the ontology concepts, is somehow "transferred" to LOs, thus providing tutors with useful information about alternative and more efficient learning paths. Most importantly, though, this kind of information can be automatically inferred and obtained, by exploiting semantic-aware techniques and tools.

ACKNOWLEDGMENT

This research described in this paper has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) (Funding Program: "HOU").

REFERENCES

[1] L. B. Baruque, F. Porto, and R. N. Melo, "Towards an Instructional Design Methodology Based on Learning Objects," Proc. International Conference on Computers and Advanced Technology in Education (CATE 2003), Rhodes, Greece, Jul. 2003.

[2] V. Devedžić, "The Setting for Semantic Web-based Education," Semantic Web and Education, Springer, New York, 2006.

[3] G. Ganapathy, and R. Lourdusamy, "Towards Ontology Development for Teaching Programming Language," Proc. International Conference on Data Mining and Knowledge Engineering, UK, Jul. 2011, pp 1845-1848.

[4] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," Journal of Knowledge Acquisition, vol. 5, issue 2, Jun. 1993, pp 199-220 .

[5] J. Heflin (ed), "OWL Web Ontology Language - Use Cases and Requirements," W3C Recoomenadion, Feb. 2004, URL: http://www.w3.org/TR/webont-req/#onto-def

[6] B. W. Kernighan, D. M. Ritchie, "The C Programming Language (2nd ed.),". Englewood Cliffs, NJ: Prentice Hall.ISBN 0-13-110362-8, 1988.

[7] R. McGreal, "Learning Objects: A practical Definition," International Journal of Instructional Technology and Distance Learning, vol. 1, Sep. 2004, pp 21-32.

[8] R. Mizoguchi, M. Ikeda, K. Sinitsa, "Roles of Shared Ontology in AI-ED Research: Intelligence, Conceptualization, Standardization and Reusability," Proc. International Conference on Artificial Intelligence in Education, 1997, pp. 537-544.

[9] G. Nikolopoulos, G. Solomou, C. Pierracheas, A. Kameas, "Modeling the Characteristics of a Learning Object for Use within e-Learning Applications," Proc. 5th Balcan Conference in Informatics (BCI 2012), Novi Sad, Serbia, Sep. 2012 (to appear).

[10] N. Noy, D. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.

[11] P. R. Polsani, "Use and Abuse of Reusable Learning Objects," Journal of Digital Information, vo. 3, no 4, Feb. 2003. URL: http://journals.tdl.org/jodi/article/viewArticle/89/88.