Modeling the Knowledge Domain of the Java Programming Language as an Ontology

Aggeliki Kouneli[1], Georgia Solomou[1], Christos Pierrakeas[1,2], and Achilles Kameas[1]

[1] Educational Content, Methodology and Technology Laboratory (e-CoMeT Lab), Hellenic Open University (HOU), Greece
[2] Dept. of Computer Science in Administration and Economy Technological Educational Institute (TEI) of Patras, Greece
{kouneli,solomou,pierrakeas,kameas}@eap.gr

Abstract. Java is a very popular programming language and many study programs in Informatics worldwide include courses particularly designed for its learning. It is considered as the best paradigm for introducing students with object-oriented programming and concepts. Considering Java's popularity, we initially make an attempt to model this language by using a quite expressing and rich knowledge representation structure, like is ontology. Our aim is to capture the semantics of Java concepts in a way that would render them utilizable by intelligent e-learning applications. Because the construction of an ontology is not an easy task, we follow very specific steps when building the Java ontology. We then take advantage of an already implemented model describing the structure of learning outcomes and combine it with our ontology, with a view to offer a more effective way in organizing the course of Java in the Hellenic Open University.

Keywords: ontologies, java, programming languages, distance education, learning outcomes

1 Introduction

In distance education, the delivery of knowledge is accomplished through learning management systems or other similar in scope applications and mechanisms. The extension of these systems with semantic-aware technologies could benefit both learners' and tutors, by providing them with intelligent applications and tools.

Ontologies are a very popular knowledge representation technique that provides applications with the ability to process semantics. They enable knowledge reuse and sharing across applications and groups of people and appear with many applications in various fields. Especially in the field of education, ontologies have been used for representing knowledge domains, teaching strategies, student profiles, as well as for competence and learning goal's modeling [1]. Besides, as stated by Mizoguchi et al. in [6], ontologies render education systems smart and reflective, deliver explicit knowledge, standardize the vocabulary of a knowledge domain, and make communication easier and knowledge reusable.

Having these in mind, through this work, we make an attempt to create an ontology with the view to capture all knowledge delivered by a distance learning course. Our goal is to incorporate the resulting model in an appropriately designed intelligent mechanism able to process education-specific ontologies. Such a mechanism, capable of combining and inferring knowledge, could provide both learners and tutors with alternative and more effective paths in learning.

In section 2 we make reference to the ontologies' exact definition and to a methodology for building ontological representations. Section 3 gives an outline of all concepts and relationships used for modeling the Java programming language. In section 4 we apply this ontology in order to create and organize the learning outcomes for the course of Java. Conclusions follow, in Section 5.

2 About Ontologies

In this section we propose a methodology, for creating well-structured ontological representations.

2.1 Definition and Structure

The computer and information science borrows the word 'ontology' from Philosophy and assigns to it a more specific and technical definition. Gruber in [3] defines an ontology as "*an explicit specification of a conceptualization*" and outlines its utilization as a means to represent a specific domain of knowledge or discourse in a more typical way.

Ontologies, representing the knowledge and capturing a particular domain's semantics, constitute a formalism for perceiving and processing information, sharing knowledge, allowing its reuse and thus enabling communication between heterogeneous and distributed systems.

The basic structural elements of an ontology are a) *classes*, representing the concepts related to a specific domain of knowledge, b) *properties*, expressing types of interactions among the domain concepts and further divided into *object properties* and *datatype properties*, c) *instances*, representing specific entities that are members of a class and d) *axioms* that express true facts about the ontology entities.

2.2 A Methodology for Building Ontologies

The task of building an ontology requires significant effort and collaboration between ontology engineers and domain experts. Many different methodologies, describing the process of developing an ontology have been proposed in literature, albeit none of them has been standardized yet. Some well-known methodologies that mainly apply to the development of domain ontologies are proposed in [2], [7] and [8].

Based mostly on the quite popular approach of Noy and McGuinness, described in [7], we adopt the following five phases for building our domain ontology (see Fig. 1):

1. *Specification*: This is the starting phase where all necessary knowledge needs to be specified by experts of the knowledge domain based on various characteristics about the resulting ontology, like the level of formality, its purpose and scope, the intended end-users, etc.
2. *Conceptualization*: During this phase the important terms of the knowledge domain are enumerated, finally leading to the design of the domain's concept map.
3. *Implementation*: Includes the transformation of the previously created conceptual model into a formal computable model, by utilizing a typical language, like OWL.
4. *Evaluation*: The phase of evaluation implies the ontology's check in terms of clarity, consistency and reusability.
5. *Documentation and maintenance*: The last phase, where it is necessary to document all implemented concepts and relations in the ontology for future reference and maintenance reasons.

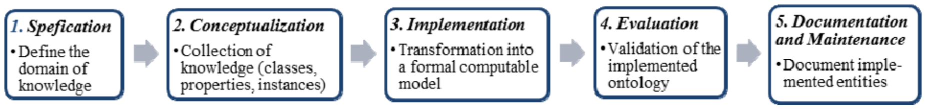

1. Spefication
• Define the domain of knowledge
2. Conceptualization
• Collection of knowledge (classes, properties, instances)
3. Implementation
• Transformation into a formal computable model
4. Evaluation
• Validation of the implemented ontology
5. Documentation and Maintenance
• Document implemented entities

Fig. 1. The five steps in the ontology building methodology

3 The Java Ontology

By following our ontology building methodology, we constructed an ontological model for the knowledge domain of the Java Programming Language. The ontology was constructed by the aid of Protégé and we were based on the most recent version of the Web Ontology Language and W3C standard, OWL 2[1].

3.1 Concepts in Java

As a first step, we collected the main concepts of Java based on the formal guide of the Java programming language, known as the *The Java Tutorial*, and provided by the Oracle Corporation[2].The main top concept, derived from this conceptual analysis, is captured by the *JavaElement* class, whereas other top concepts are ii) *Keyword* and iii) *LiteralValue*.

As is shown in Fig. 1, the *JavaElement* class is divided into twelve different subclasses, each one describing key elements and concepts met in the vocabulary and syntax of the Java programming language. More specifically, the *DataType* class encompasses the notion of data types in Java, such as *primitive* data types and *non-primitive (reference)* data types an consists of the respective classes. The latter is further divided into all possible structures for expressing a reference data type in Java, that is *Array*, *Class* and *Interface*.

[1] `http://www.w3.org/TR/owl-primer/`
[2] `http://docs.oracle.com/javase/tutorial/`

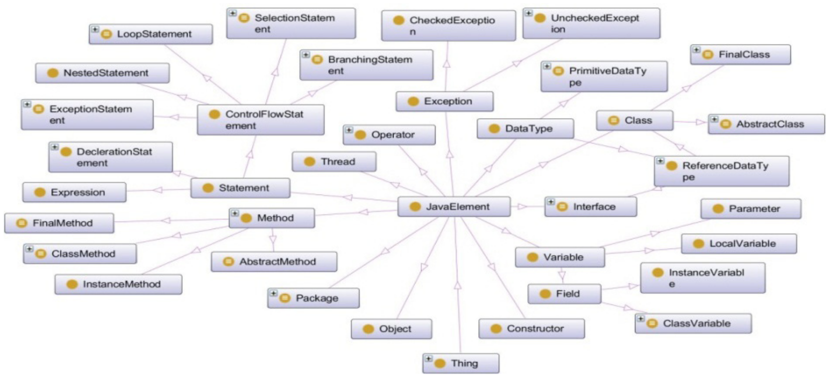


Fig. 2. Part of the class hierarchy of the implemented ontology about the Java programming language.

Operators constitute another basic element, met not only in Java, but also in any other programming language, and so a class with this name was included in our model. Operators were grouped into the following categories (transformed into classes) and set under the *Operator* class: *ArithmeticOperator*, BitwiseOperator,

AssignmentOperator, *ConditionalOperator*, *ShiftOperator*, *UnaryOperator* and *RelationalOperator*.

Variables are another essential element in a programming language and its notion in Java has been expressed via a separate class (*Variable*). Since Java contains four types of variables, i.e. *instance* variables, *class* variables, *local* variables and *parameters*, we took consideration for all these by setting respective classes in our model (*ClassVariable, InstanceVariable, LocalVariable* and *Parameter*) and placed them as children of the main *Variable* class.

The class *Statement* represents the smallest standalone element of a Java program. Under this class, three subclasses were created (*ControlFlowStatement, DeclarationStatement, Expression*) reflecting the various types of statements that a programmer may use in order to write source code in Java. The *ControlFlowStatement* class is further refined in a number of more specific types, each represented by a subclass and populated with a set of individuals corresponding to keywords used for their declaration.

As far as the Java methods are concerned, they are captured by the *Method* class that contains the following subclasses: *AbstractMethod*, *FinalMethod*, *ClassMethod* and *InstanceMethod*. Furthermore, provision for all object-oriented concepts, such as *object*, *constructor*, *interface*, *class* and *packages* has been made in our ontology.

The *Exception* class stands for a specific element in Java, which expresses problematic and erroneous situations when compiling or running a Java program. All exceptions that cannot be caught are considered members of the *UncheckedException* class. This is further divided into *RunTimeException* and *Error* classes. A runtime exception aims to capture all those exceptions that are internal to the application, in contrast to an external error case, which is included in the the *Error* class. 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*, expressing reserved words with a special use within a program. Those keywords can be visibility modifiers (members of the *Modifier* class) or even terms that control the access privileges of various Java elements (members of the *AccessControlModifier* class). On the other hand, *LiteralValue* is a top class containing literals that have a special meaning for the Java compiler.

3.2 Description of Properties

The various types of interaction among ontology concepts are expressed through respective relations, known as *properties*. Properties are divided into *datatype properties* which link an individual to a particular value and *object properties* that are relationships between individuals.

We have defined 7 properties. More specifically, the pair-wise inverse properties *isSubclassOf* and *isSuperclassOf* are used to declare a parent-child relation between two Java classes. They are irreflexive, meaning that no individual can be related to itself via such a property. *hasMember* and *isMemberOf* are also inverse properties and used for relating Java interfaces, methods or packages. The *instance* property connects a Java class with its members, whilst the *hasModifier* property correlates any individual with a certain modifier. Finally, to determine the particular relation of a Java concept with a reserved keyword, the *isDefinedByKeyword* property is used.

4 The Role of the Java Ontology in Defining Learning Outcomes

All courses of the Hellenic Open University (HOU), which is an educational institution in Greece, specialized in lifelong learning, have been designed according distance learning principles. This means that each piece of its provided educational material has been directly related to learning outcomes. Learning outcomes play an important role in the instructional design process and especially in when organizing a distance learning course. According to [5], learning outcomes are statements that show "*what a learner is expected to know, understand and be able to demonstrate after completion of a learning process*" and should accompany any educational material serving the purposes of a study program offered from distance.

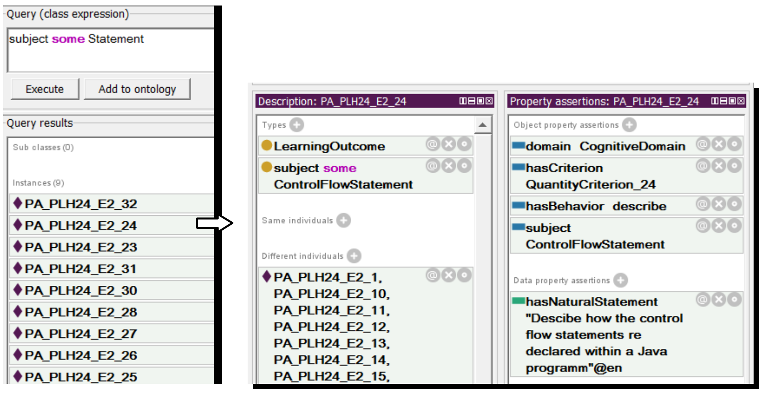
The knowledge domain of the Java programming language is covered by the HOU's course module of 'Software Engineering', which is offered as part of the HOU's study program in Informatics.

In a previous work, described in [4], an ontological model has been introduced for representing learning outcomes' structure and classification. This model has been successfully applied for the reconstruction of 32 learning outcomes referring to Java *Operators*, *Arrays* and *Statements*.

Through this work, we took advantage of the aforementioned model and augmented it so as to include our proposed ontology about Java. Our aim was to enhance this mixed schema with the possibility to process, organize and correlate learning outcomes according to their subject and thus lead to the creation of effective learning paths.

For expressing the subject of a learning outcome (i.e., the particular concept of the knowledge domain to which this learning outcome refers) the datatype property *subject* was used in the previous introduced model. Consequently, any request for classifying or correlating learning outcomes according their subject, was actually based on a string match process. By augmenting this model with the Java ontology, *subject* was transformed into an object property. Consequently, each reference to a Java concept in the combined schema was replaced by its corresponding entity in the Java ontology. As a result, a richer network of relationships among learning outcomes was produced, being thus possible to retrieve and infer more semantically enriched results.

To determine the possibilities of this new, enhanced ontological model we run some semantic queries against it, through the DL query tab of Protégé. For example, if we want to obtain all learning outcomes concerning *Statements* in Java, we need to run the following query: `subject some Statement`. This query, expressed in the Manchester OWL Syntax, returns 9 results (see Fig. 3), each one being an instance of the *LearningOutcome* class. A more careful look, though, reveals that none of the retrieved individuals explicitly refers to Java *Statements* but to narrower in meaning concepts. For example, individual PA_PLH24_E2_24 refers to *Control Flow Statements* (see Fig. 3). Nevertheless, the retrieval of this particular individual becomes possible because *Control Flow Statement* has been declared as a subclass of the *Statement* class. The same holds for all other obtained results.


Query (class expression)
subject some Statement

Query results
Sub classes (0)
Instances (9)
PA_PLH24_E2_32
PA_PLH24_E2_24
PA_PLH24_E2_23
PA_PLH24_E2_31
PA_PLH24_E2_30
PA_PLH24_E2_28
PA_PLH24_E2_27
PA_PLH24_E2_26
PA_PLH24_E2_25

Description: PA_PLH24_E2_24
Types
LearningOutcome
subject some ControlFlowStatement
Same individuals
Different individuals
PA_PLH24_E2_1, PA_PLH24_E2_10, PA_PLH24_E2_11, PA_PLH24_E2_12, PA_PLH24_E2_13, PA_PLH24_E2_14, PA_PLH24_E2_15,

Property assertions: PA_PLH24_E2_24
Object property assertions
domain CognitiveDomain
hasCriterion QuantityCriterion_24
hasBehavior describe
subject ControlFlowStatement
Data property assertions
hasNaturalStatement "Descibe how the control flow statements re declared within a Java programm"@en

Fig. 3. Searching for learning outcomes concerning Java *Statements* (*left*). Ontological representation of instance PA_PLH24_E2_24 in Protégé (*right*).

5 Conclusions and Future Work

What we presented here was an ontology representing the most important notions in the Java programming language. To this end, we had in mind both the official Java Tutorial and the extent to which this domain is covered by the introductory Java course offered by the study program in Informatics of HOU.

This representation was implemented by following very specific steps (methodology). The resulting model captures the most important aspects of Java as classes and correlates them through a number of relationships. The aim of this approach was to produce a semantically enriched representation about a so popular knowledge domain in the field of Informatics, with a view to further exploit it within intelligent e-learning applications. For the time being, and in order to examine our ontology's added value in deploying intelligent applications, we combined it with an already implemented ontological schema describing learning outcomes. As a result, we managed to enhance this model in terms of learning outcomes discovery and retrieval.

Our future goal is to feed this combined schema in an appropriately designed e-learning application, able to handle semantics. By this way, we could probably enhance and alleviate the process of designing a course, by allowing ontology-based systems to process knowledge and lead to more effective decisions.

Acknowledgement. The research described in this paper was partly funded by the National Strategic Reference Framework programme 2007-2013, project MIS 296121 "Hellenic Open University".

References

1. Devedžić, V.: The Setting for Semantic Web-based Education. In: Semantic Web and Education. Springer, New York (2006)
2. Fernandez-Lopez, M., Gomez-Perez, A., Juristo, N.: METHONTOLOGY: from Ontological Art towards Ontological Engineering. In: Proceedings of the AAAI 1997 Spring Symposium Series on Ontological Engineering, pp. 33–40. AAAI Press, Stanford (1997)
3. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. J. Knowledge Acquisition 5, 199–220 (1993)
4. Kalou, A., Solomou, G., Pierrakeas, C., Kameas, A.: An Onotlogy Model for Building, Classifying and Using Learning Outcomes: In International Conference on Advanced Learning Technologies, Rome (to appear, 2012)
5. Kennedy, D., Hyland, A., Ryan, N.: Writing and using Learning Outcomes. Bologna Handbook, Implementing Bologna in your Institution, C3. 4-1, 1–30 (2006)
6. Mizoguchi, R., Ikeda, M., Sinitsa, K.: Roles of Shared Ontology in AI-ED Research: Intelligence, Conceptualization, Standardization and Reusability. In: International Conference on Artificial Intelligence in Education, pp. 537–544 (1997)

7. Noy, N., McGuinness, D.: 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)
8. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In: Proc. Of Workshop on Basic Ontological Issues in Knowledge Sharing in Conjuction with International Joint Conference on Artificial Intelligence, Montreal, Canada (1995)