The MATHESIS Ontology: Reusable Authoring Knowledge for Reusable Intelligent Tutors

Dimitrios SKLAVAKIS [a,1] and Ioannis REFANIDIS [a]
[a] *Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece*

Abstract. This paper describes the MATHESIS Ontology, which is part of the MATHESIS project that aims at the development of an intelligent authoring environment for reusable model-tracing math tutors. The purpose of the ontology is to provide a semantic and therefore inspectable and reusable representation of the declarative and procedural authoring knowledge necessary for the development of a model-tracing tutor as well as of the declarative and procedural knowledge of the tutor under development. While the declarative knowledge is represented with the basic OWL components, i.e. classes, individuals and properties, the procedural knowledge is represented via the *process model* of the OWL-S web service description ontology. By using OWL-S, every authoring or tutoring task is represented as a composite process. Based on such an ontological representation, a suite of authoring tools will be developed at the final stage of the project.

Keywords. authoring systems, ontologies, semantic web, model-tracing tutors

Introduction

Intelligent tutoring systems and especially model-tracing tutors have been proven quite successful in the area of mathematics [1]. Despite their efficiency, these tutors are expensive to build both in time and human resources. The main goal of the ongoing MATHESIS project is to develop authoring tools for model-tracing tutors in mathematics, with knowledge re-use being the primary characteristic of the authored tutors as well as of the authoring knowledge used by the tools.

The MATHESIS ontology is an OWL ontology developed with the Protégé-OWL ontology editor. Its development is the second stage of the MATHESIS project. Aiming at the development of real-world, fully functional model-tracing math tutors, the project is being developed in a bottom-up approach. In the first stage the MATHESIS Algebra Tutor was developed in the domain of expanding and factoring algebraic expressions [2]. The tutor is web-based, using HTML and JavaScript. The authoring of the tutor as well as the code of the tutor were used to develop the MATHESIS ontology in a bottom-up way, as it will be described later.

The rest of the paper is structured as follows: Section 1 gives a brief presentation of the process model of OWL-S. Section 2 describes the part of the ontology that

[1] Corresponding Author: Dimitrios Sklavakis, Department of Applied Informatics, University of Macedonia, Egnatia 156, P.O. Box 1591, 540 06 Thessaloniki, Greece; E-mail: dsklavakis@uom.gr.

represents the model-tracing tutor(s), while Section 3 describes the representation of the authoring knowledge. Section 4 presents related work and, finally, Section 5 concludes with a discussion about the ontology and further work to complete the MATHESIS project.

1. The OWL-S process model

OWL-S is a web service description ontology designed to enable the tasks of (semi-) automatic discovery, invocation, composition and interoperation of Web services. It provides a language for describing service compositions. Every service is viewed as a Process. There are three subclasses of Process, namely the AtomicProcess, CompositeProcess and SimpleProcess.

Composite processes are decomposable into other composite or atomic processes. Their decomposition is specified by using control constructs such as Sequence and If-Then-Else. Any composite process can be considered as a tree whose non-terminal nodes are labeled with control constructs. The leaves of the tree are invocations of other processes, composite or atomic. These invocations are indicated as instances of the Perform control construct.

2. Tutor Representation in MATHESIS ontology

The MATHESIS project has as its ultimate goal the development of authoring tools that will guide the authoring of real-world, fully functional model-tracing math tutors. This means that during the authoring process and in the end, the result will be program code that implements the tutor, i.e. the ontology must be able to represent the program code. For this reason, in the first stage of the MATHESIS project the MATHESIS Algebra tutor was developed to be used as a prototype target tutor.

The MATHESIS Algebra tutor is a Web-based, model-tracing tutor that teaches expanding and factoring of algebraic operations: monomial and polynomial operations, identities, factoring. It is implemented as a simple HTML page with JavaScript controlling the interface interaction with the user and implementing the tutoring, domain and student models. Therefore, it is the representation of the HTML and JavaScript code that forms the low-level MATHESIS ontology of the tutor as described below.

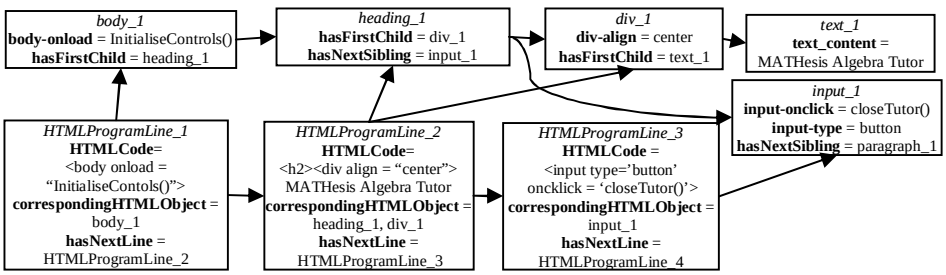

Figure 1. The HTML code and the corresponding DOM representation

2.1. Representation of the HTML Code of the Tutor

The representation of the HTML code and the corresponding Document Object Model (DOM) of the user interface are shown in Figure 1. Each line of the HTML code is represented as an instance of the HTMLProgramLine class having three properties: the HTMLCode, hasNextLine and correspondingHTMLObject. The last one points to the HTMLObject defined by the HTML code.

Each HTMLObject has the corresponding HTML properties as well as the hasFirstChild and hasNextSibling which implement the DOM tree. Therefore, there are two representations of the HTML code enabling a bottom-up creation of the ontology (from HTML code to DOM) and a top-down (from the DOM to HTML code).

2.2. Representation of the JavaScript Code of the Tutor

The representation of the JavaScript code is shown in Figure 2. Each line of the JavaScript code is represented as an instance of the JavaScript_ProgramLine class having three properties: the javascriptCode, hasNextLine and hasJavaScriptStatement. The last one points to a JavaScript_Statement instance which is an AtomicProcess of the OWL-S process model.

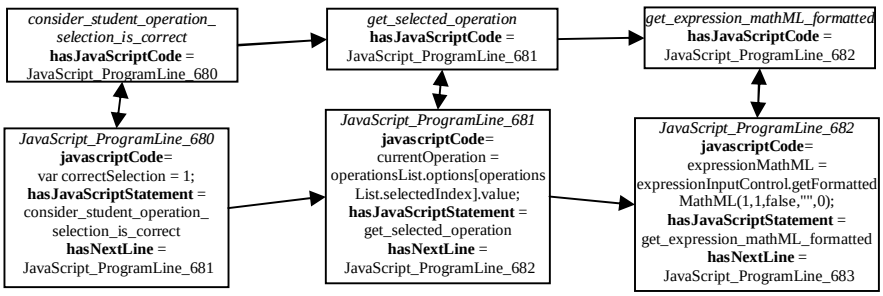

Figure 2. The JavaScript code and the corresponding JavaScript_Statement Atomic processes

Once again, there are two representations of the JavaScript code enabling a bottom-up creation of the ontology (from JavaScript code to JavaScript_Statement atomic processes) and a top-down (from the JavaScript_Statement atomic processes to JavaScript code).

2.3. Representation of the Tutoring Model

Being procedural knowledge, the model-tracing algorithm is represented as a composite process named Model_Tracing_Algorithm, shown in Figure 3. Each step of the algorithm is also a composite process. For example the Task_Execution_Expert_Process step can be described by an algorithm that performs other composite processes. These processes are instances of subclasses of the Task_Execution_Expert_Process class, shown in Figure 4. During the authoring of a specific tutor, the authoring tools will parse the tree of the Model_Tracing_Algorithm composite process and invoke for each tutoring task a corresponding authoring task represented also as a composite process in order to implement the tutoring task for the specific tutor (described in Section 3).

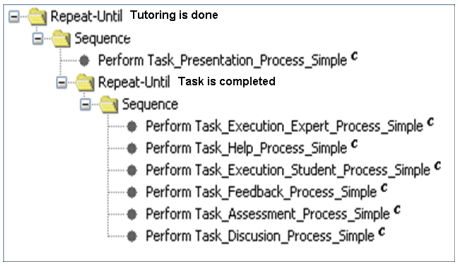

Figure 3. The Model_Tracing_Algorithm process

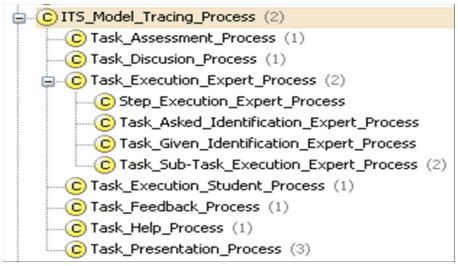

Figure 4. The model-tracing processes taxonomy

2.4. Representation of the Domain Expert Model

In a model-tracing tutor the Domain Expert Model executes the next step of the problem and produces the correct solution(s) to compare them with the student's proposed solution. If the solution step is simple, then it is represented as an instance of the atomic process JavaScript_Statement (see Section 2.2). If the step is complex, then it is represented as a composite process. This analysis ends when the produced composite processes contain only atomic processes, i.e. JavaScript_Statement instances.

3. Authoring Knowledge Representation in MATHESIS Ontology

As mentioned in Section 2.3, for each tutoring task of the model-tracing algorithm, there is a corresponding authoring task in the MATHESIS ontology, represented as a composite process. The authoring_task_execute_task_by_expert (Figure 5) for example corresponds to the Task_Execution_Expert_Process_Simple tutoring task (Figure 3). The define_data_structures_for_knowledge_components process, one of the composite processes that form this authoring task, is shown in Figure 6.

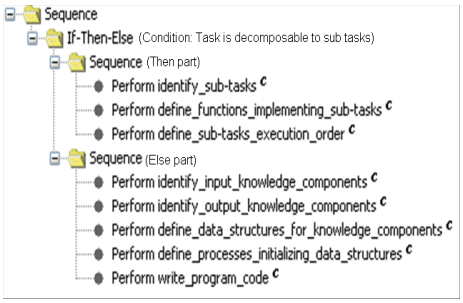

Figure 5. The authoring process for the Execute_Task_By_Expert tutoring task

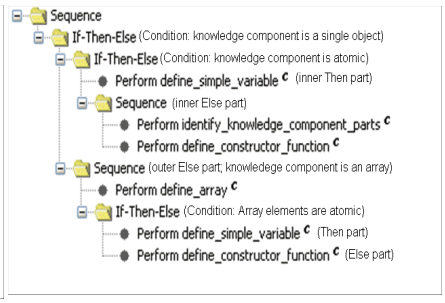

Figure 6. The authoring composite process define_data_structures_for_knowledge_componets

Based on all the above representations, the overall authoring process will have as follows: The tools will parse the model-tracing algorithm (Section 2.3). For each step of the algorithm, the corresponding authoring process will be called and traced. This authoring process will guide the author in creating recursively the various parts of the

tutor (Sections 2.1, 2.2, 2.4); as a consequence, any newly created tutor part becomes new knowledge in the ontology to be used later.

4. Related Work

The use of ontologies and semantic web services in the field of ITSs is relatively new. Ontological engineering is used to support authoring of instructional scenarios [3], provide educational feedback, or plan learning resources. However, there is a lack of semantic expressiveness and, more important, the difficult task of using and integrating low-level learning services to compose more complex ones, is not faced at all.

It is this difficult task that the MATHESIS project is trying to address using as low-level learning services the concept of problem-solving tasks and providing through the MATHESIS ontology a semantic description of these tasks and the way they can be combined to create more complex learning services (intelligent tutors).

5. Discussion and Further Work

In an overview of intelligent tutoring system authoring tools [4], it is suggested that authoring tools should support interoperability, re-usability, durability, scalability and accessibility. Even in this preliminary form, the MATHESIS ontology provides a proof-of-concept that it can serve as the basis for the development of authoring tools and implemented tutors that will match these criteria. The main reason for this claim is that the ontology provides an open, modular and multi-level representation (ranging from conceptual design to program code) of both authoring and tutoring knowledge.

Of course, it is expected that a lot of work has to be done: the ontology must be extended, refined and formalized. This will be done by representing the whole MATHESIS Algebra tutor into the ontology. Because of the tremendous workload this task entails, an initial set of authoring tools are being developed: parsers for HTML to/from MATHESIS interface model and for JavaScript to/from MATHESIS JavaScript_Statements/Program code translation; interpreters for the authoring and tutoring OWL-S processes; visualization tools for the authoring processes, the tutoring model (model-tracing algorithm) and the tutor parts being developed. Most of these tools will be implemented as Java plug-ins for the Protégé-OWL editor, accessing and updating the MATHESIS ontology.

References

[1] K.R. Koedinger, J.R. Anderson, W.H. Hadley, M.A. Mark, Intelligent Tutoring Goes to School in the Big City, *International Journal of Artificial Intelligence in Education* **8** (1997), 30-43.

[2] D. Sklavakis, I. Refanidis, An Individualized Web-Based Algebra Tutor Based on Dynamic Deep Model Tracing, *Proceedings of the 5th Hellenic Conference on Artificial Intelligence (SETN '08)*, LNAI **vol. 5138**, 389-394, Springer, Berlin/Heidelberg, 2008.

[3] R. Mizoguchi, Y. Hayashi, J. Bourdeau, Inside Theory-Aware and Standards-Compliant Authoring System, *Proceedings of the Fifth International Workshop on Ontologies and Semantic Web for E-Learning (SWEL '07)*, http://compsci.wssu.edu/iis/Papers/SWEL07-Proceedings.pdf.

[4] T.Murray, An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art, *Authoring Tools for Advanced Technology Learning Environments*, 491-544, Kluwer Academic Publishers, Netherlands, 2003.