Protus 2.0: Ontology-based semantic recommendation in programming tutoring system

Boban Vesin^a,*, Mirjana Ivanović^b, Aleksandra Klašnja-Milićević^a, Zoran Budimac^b

^aHigher School of Professional Business Studies, Vladimir Perića-Valtera 4, 21000, Serbia
^bFaculty of Science, Department of Mathematics and Informatics, Trg Dositeja Obradovica 3, 21000 Novi Sad, Serbia

A R T I C L E   I N F O

Keywords:
Semantic web
Ontology
Tutoring system
Recommendation systems
Personalization
Adaptation rules

A B S T R A C T

With the development of the Semantic web the use of ontologies as a formalism to describe knowledge and information in a way that can be shared on the web is becoming common. The explicit conceptualization of system components in a form of ontology facilitates knowledge sharing, knowledge reuse, communication and collaboration and construction of knowledge rich and intensive systems. Semantic web provides huge potential and opportunities for developing the next generation of e-learning systems. In previous work, we presented tutoring system named Protus (PRogramming TUtoring System) that is used for learning the essence of Java programming language. It uses principles of learning style identification and content recommendation for course personalization. This paper presents new approach to perform effective personalization and reuse of knowledge. Developing such reusable ontologies is an important goal of ontology research.

Ontologies allow specifying formally and explicitly the concepts that appear in a concrete domain, their properties and their relationships (Gascueña, Fernández-Caballero, & González, 2006). Furthermore, they are useful in many domains: and especially in educational environments, as they enable people and/or software agents to share a common understanding of the knowledge structure. Moreover, they permit to reuse knowledge, it means that, it is not necessary to develop ontology from scratch if another ontology is available for use in the modeling of the current domain.

Although ontologies have a set of basic implicit reasoning mechanisms derived from the description logic which they are typically based on (such as classification, relations, instance checking, etc.), they need rules to make further inferences and to express relations that cannot be represented by ontological reasoning (e.g., in a learning domain it could be necessary to express the fact that a topic A is a prerequisite of topic B to make the right suggestions to the learner). Thus, ontologies require a rule system to derive/use further information that cannot be captured by them, and rule systems require ontologies in order to have a shared definition of the concepts and relations mentioned in the rules. Rules also allow adding expressiveness to the representation formalism, reasoning on the instances, and they can be orthogonal to the description logic on which ontologies are based on (Henze, Dolog, & Hejdl, 2004).

Learning technology is nowadays subject to intensive standardization efforts that have led to different specifications defining the
functions and information formats provided by different components (Aroyo & Mizoguchi, 2003). However, differences persist and in some cases, the production of proposed standards and protocols leads to increased heterogeneity. Further, the methodologies used to describe resources and activities are diverse, resulting in semantic interoperability problems and conversations between elements that have still not been addressed. Therefore, important aspect of our approach is the use of Sharable Content Object Reference Model – SCORM (SCORM, 2006), as a guideline format for content development, and for implementation of several ontologies for describing the domain, learner model, personalization actions, teaching strategy, communication and learner interface. Ontologies are included in Protus 2.0 and they are written in OWL. To support the development of the ontologies and the translation in OWL, we use the open source tool Protégé-4.1 (Protégé, 2012). Rules have to be expressed using standard semantic and formalisms as well as ontologies, based on the idea that an open format supports scalability. In our project, we exploit SWRL (SWRL, 2012), a Semantic web Rule Language combining OWL and RuleML (OWL, 2012). SWRL allows interoperability with major rules systems such as JESS (Carmagnola, Cena, Geni, & Torre, 2005).

In our previous work, we presented general tutoring system named Protus – PRogramming TUtoring System that is used and tested for learning basic elements of Java programming language (Vesin, Ivanović, & Budimac, 2009). Developed and implemented tutoring system is based on principles of Learning styles recognition and content recommendation for course personalization (Vesin, Ivanović, Budimac, & Pribela, 2008; Vesin, Ivanović, Klašnja-Miličević, & Budimac, 2011). Learning style can be defined as a combination of cognitive, affective and other psychological characteristics that serve as relatively stable indicators of the way a learner perceives, interacts with and responds to the learning environment (Popescu, Badica, & Trigano, 2007). Experimental results presented in Klašnja-Miličević, Vesin, Ivanović, and Budimac (2011b) show that testing the learners’ learning styles has the potential to improve the quality of an intelligent tutoring system, as well as keep the recommendation up-to-date.

The main objective of this paper is to present advantages and new functionalities of the Protus 2.0 – a new version of Protus tutoring system that will completely rely on Semantic web standards and technologies. Implemented architecture improves the ontology utilization, where the representation of each component is made by a specific ontology. This way, it will be possible to promote a clear separation of concerns of the components of tutoring system, to make explicit the communication among the components, to specify support to build the components and to emphasize the gains of the use of Semantic web in the development of tutoring systems. This will also allow better interoperability and reusability of Protus 2.0 components in the future.

The rest of the paper is organized as follows. In the Section 2 appropriate related work is analyzed and discussed. Implemented adaptation techniques and technologies used are presented in Sections 3 and 4, respectively. Section 5 describes the representation of components according to Semantic web technologies within proposed system, and discusses use of all implemented ontologies. The overview of implemented adaptation rules are illustrated in Section 6. That section also contains results of adaptation from the learners’ point of view. Section 7 concludes the paper and indicates directions of our further research.

2. Related work

For quite a while, a lot of research efforts have been focused on applying Semantic web technologies to different aspects of e-learning (Jovanović et al., 2007). A wide range of educational software that implements ontology-based components has been developed in recent period, but the most of these systems use ontologies only for representation of concepts, knowledge or learners data (Fernández-Breis et al., 2012; Gascueña et al., 2006; Lee, Lee, & Sun, 2002; Jia et al., 2011).

The prototype system named SMARTIES is a totally ontology-aware system which fully utilizes the qualities of ontology, computationally as well as conceptually (Mizoguchi, Hayashi, & Bourdeau, 2007). In this moment, the ontology focuses only on the abstract design of learning contents and has not been yet related to domain knowledge or learning objects to concretize the abstract design. The Personal Reader (Dolog & Nejdl, 2007) is another important result in the e-learning field. The Personal Reader provides a framework for designing, implementing and maintaining Web content readers, which provide personalized enrichment of Web content for each individual user. This system uses the Semantic web to personalize and enrich e-learning contents. It presents a service architecture relying on RDF (Resource Description Framework) and ontologies to exchange information about learning resources, the domain, and learners. Learning resources are described by means of shared ontologies (Dublin Core and Learning Objects Metadata), while reasoning and adaptation are realized by using TRIPLE, a rule-based query language for the Semantic web. Architecture for personalized e-learning based on Semantic web technologies was proposed in Henze et al. (2004). The authors propose usage of several ontologies for building adaptive educational hypermedia systems. In the architecture, this system integrates personalization services such as a recommendation and a link generation and thereby provides a personalized access to learning resources. However, their ontology does not represent the teaching strategy's functionality of a resource.

Our research is closely related to the essences of QBSL system (Dehors & Faron-Zucker, 2006). It is web-based intelligent learning system that completely relies on Semantic web technologies and standards. It reuses a large set of learning resources taken from the web and has been used as an online support system for lab sessions of Java programming course. Protus 2.0 system implements recommendation techniques in personalisation process.

In Jia et al. (2011), learning ontology and prototype system have been developed and used for constructing formal and machine-understandable conceptualization of the performance-oriented learning environment. Authors in Fernández-Breis et al. (2012) presented a system that has been developed with teachers of a Secondary School that uses ontologies to support the development and management of the educational curriculum. They described the software platform called Gescur that implements the management processes that allows the execution of the planning, direction and control of the educative curriculum.

Other proposals of ontologies and their usage for several aspects of the e-learning systems, such as learner model and preferences, domain ontology, task ontology, and others, can be found in Aroyo and Mizoguchi (2003), Sosnovsky et al. (2008) and Latham, Crockett, McLean, and Edmonds (2011). Systems SmartTrainer, developed at Osaka University, Japan, and AMS, developed at University of Twente, The Netherlands, are presented in Aroyo and Mizoguchi (2003). They are based on a Domain Model, describing the structure of the information content within the system and a User Model that represents a learner's preferences, knowledge, goals and activity history. Authors showed that the ontology concept appears to be suitable solution for knowledge systematization within the tutoring tools. In Latham et al. (2011) authors proposed architecture for developing a novel conversational intelligent tutoring system called Oscar that leads a tutoring conversational and dynamically predicts and adapts to a student's learning style. This system imitates human tutor by using knowledge of learning styles and learner behavior to predict learning style.

Authors in Sosnovsky et al. (2008) reported an experiment on integration of domain models of two different adaptive systems and therefore provide us with ideas for organization and structure of do-
main ontology. An experimental Java programming course ontology – Java Learning Object Ontology – JLLO was built and introduced in Lee, Yen Ye, and Wang (2005). It is used as a guideline in developing the learning objects of introductory Java courses and in organizing these learning objects in an adaptive learning environment.

Several studies show how semantic rules can be used in combination with ontologies for providing adaptation process in e-learning systems (Chi, 2009; Popescu et al., 2007; Sicilia, Lytras, Sances-Alonso, Garcia-Barriocanal, & Zapata-Ros, 2010). Authors in Sicilia et al. (2010) address the representation of the main elements of instructional models using the formal ontology language OWL, which can be used in conjunction with the SWRL rule language. That paper has presented the foundations of using ontologies that describe formally the sequence and structure of learning activities, the roles and their participation in the activities, and the learning objects and/or services used by each role in each activity. Study given in Chi (2009) utilizes a knowledge-intensive approach to create a general sequencing knowledge base within elementary school mathematics course. That approach included two components:

- ontology was used to represent abstract views of content sequencing and course materials and
- added semantic rules were used to represent relationships between individuals.

Authors in Popescu et al. (2007) provided precise description of modeling rules for identification of learning styles and adaptation rules based on learning styles and/or learning preferences. In order to make the presentation independent of particular rule representation formalism, they have chosen to express rules in an informal pseudo-code notation.

These works show that the most relevant difficulty in the knowledge modeling for e-Learning systems is related to the creation and maintenance of Semantic web structures (such as ontologies) which can be exploited not only to organize learning objects and to state their inter-relationship but also to build personalized learning paths and to maintain up to date learners’ cognitive states.

In the structure of previously mentioned systems, the use of ontology focuses mainly on learning objects and their related aspects. Besides, that does not facilitate the definition and communication between the other components of the system’s architecture. Architecture for tutoring system supported by several ontologies is described in this paper as a way of addressing and offering acceptable solution for the mentioned problems. We further enhance our previous work by not only adapting the content modeling but also to build personalized learning paths and to maintain up to date learners’ cognitive states.

The first time that learners use the Protus 2.0, system asks them to fill the questionnaire that contain the ILS questions based on the learning style-model by Felder and Silverman (1998) concerning the different Cognitive Styles of Learning (CSL) learners may have, which were described in Klasičnjak Milicević et al. (2011b). Based on those CSL, a GUI interface was developed to enable the learner himself/herself to categorize his/her CSL, set his/her learning goals and characteristics of work environment and the kind of course he/she wants to take. At run-time, the learner model is updated taking into account the learning activities.

### 3.1. Adaptation process in Protus 2.0

The new learner signs up by using the registration form in order to create an initial personal profile. Each profile stores personal information supplied directly by the learner, like: last name, first name, login, previous knowledge, preferences, etc. (known as static information), and information about interests, dominant meaningful words, and behavior (known as dynamic information). The learner may change static information at any time by editing it.

The first time that learners use the Protus 2.0, system asks them to fill the questionnaire that contain the ILS questions based on the learning style-model by Felder and Solomon (1996) to predict their own learning styles. Initial learning style is calculated based on the learners’ answers and recorded in appropriate learner model.

During sessions, learners visit various resources and solve various tasks. When the learner completes the sequence of learning materials, the Protus 2.0 system evaluates the learner’s acquired knowledge (Fig. 1). Tests contain several multiple-choice questions and code completion tasks (Vesin et al., 2009). Protus 2.0 provides feedback on learners’ answers and gives the correct solutions after the test. The learners’ ratings can be interpreted according to the percentage of correct answers. Two learners are said to be similar to each other if they are evaluated by the system with the same ratings for a similar navigational sequence. Recommendation process can be carried out according to these ratings based on the collaborative filtering approach that is described in our previous work (Klašnja-Miličević et al., 2011b).

There can be different a sequence of resources that depends on navigational sequence determined for the learning style of the particular learner. In next sections we will present implemented architecture of Protus 2.0 as well as the set of proposed SWRL rules for altering navigation sequences and recommendation of resources. The benefit of representing teaching strategies as SWRL rules is that the strategies’ computations would be explicitly represented in the ontology, and could be viewed and edited, as well as reasoned about by other applications.
3.2. Structure of learning objects in Protus 2.0

The learning content is divided into six units, each of which consists of several lessons. Every lesson (out of eighteen) contains three basic parts: theory session (tutorials), examples and tests (Fig. 2). Tutorials can contain various resources with different purposes (introductory, syntax rule presentation, block diagrams, etc.). To each lesson an unlimited number of tutorials, examples and tests can be attached. Their number can be increased by teachers using an appropriate authoring tool. Protus 2.0 contains various forms for adding new learning material: tutorials, examples and tests (Vesin et al., 2009).

4. Standardization and technology used

With the development of the Semantic web (Swartout & Tate, 1999) the use of ontologies as a formalism to describe knowledge and information in a way that can be shared on the web is becoming common. Adoption of the standard for the Ontology Web Language (OWL) (Tran, Cimiano, & Ankolekar, 2006) is propelling this trend toward large scale application in different domains. However, the utility of the ontologies is limited by the processing mechanisms that are smoothly integrated with this form of representation. Therefore there is an effort on the way to formalize the logic layer for ontologies. The Semantic web Rule Language (SWRL, 2012) is proposed as an important step in this direction, building on the experience of the previous work on RuleML (The Rule Markup Initiative, 2012). The availability of standardized rule language for the Semantic web will make it possible to use both ontologies and rules as a basis for innovative applications that are connected to the Semantic web.

4.1. SWRL

SWRL (Sicilia et al., 2010) is a language specifically targeted to introduce inference rules in knowledge models represented in OWL. Semantic web Rule Language (SWRL, 2012) is probably the most popular formalism in Web community for expressing knowledge in the form of rules. Specifically, SWRL is based on a combination of Web Ontology Language (OWL, 2012) and Rule Markup Language (The Rule Markup Initiative, 2012) and has been proposed as a W3C candidate standard for formalizing the expression of rules in Web context.

Rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as “whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold”.

The main advantage of SWRL is the simplicity it offers, while extending the expressiveness of OWL. Another benefit of SWRL is its compatibility with OWL syntax and semantics, since they are both combined in the same logical language.

It is worthy to mention that most of the existing rule-based applications for the Web have adopted SWRL approach in order to express rules. SWRL is neither a highly expressive language (e.g., no negation is available) nor a decidable one, but it remains simple (Papataxiarhis, Tsetsos, Karali, Stamatopoulos, & Hadjieftymiades, 2010).

4.2. Protégé

Protégé is an open source ontology editor and knowledge-based framework (Protégé, 2012). The Protégé OWL Plugin provides a SWRL editor, which enables the formalization of SWRL rules in conjunction with OWL ontologies. It provides graphical user interface for easy development and management of ontology. Moreover, the SWRLTab within it provides editor and validation tool to develop inference rules. In addition, other ontology can be imported to achieve knowledge reuse.

Besides the fact that Protégé is a good, intuitive and widely used tool, it also provides an open-source Java API that enables to access the ontological model and to use Protégé Forms from a Java environment.

4.3. Jess

Usually, the SWRL rules are translated to existing rule systems (e.g., Jess) that handle the reasoning tasks partially, since they are not aimed to manage knowledge expressed in terms of first-order logic or its subsets (Papataxiarhis et al., 2010). Jess (Java Expert System Shell) (2012) is a Java framework for editing and applying rules, since it contains a scripting environment and a rule engine, as well. Recently, the evolution of rule technologies on the Web has led Jess to rebound its practical value in the community of Web developers. Moreover, the fact that Jess is a Java-based system facilitates its integration with a number of Web programming paradigms like Java servlets or applets. Finally, it supports backward-chaining and some additional features like...
procedural attachments (Papataxiarhis et al., 2010). Finally, Jess was accessed via SWRL-Jess bridge. Jess is also a rule-based inference engine that can support RDF, OWL and SWRL inference. For this inference it must use SWRLTab, which is one of the OWL Plug-ins for Protégé (Lee, Kim, Lee, & Lee, 2007).

5. Protus 2.0 architecture

Protus 2.0 is a tutoring system designed to support learning processes in different courses and domains but with intention to be used for learning programming languages (Fig. 3). In spite of fact that this system is designed and implemented as a general tutoring system for different programming languages, the first completely implemented and tested version was used for learning introductory Java programming course (Klašnja-Milicević et al., 2011a). Java is chosen because it is wide used programming language at our University, and as it is a clear example of an object-oriented language therefore suitable for the teaching of the concepts of object-orientation. It is an interactive system with primary goal to allow learners to use teaching material prepared within appropriate introductory programming course. But above that, it also includes part for testing student’s acquired knowledge.

Ontology is relatively new concept that provides a common vocabulary of an area and defines, with different levels of formality, the meaning of the terms and the relationships between them. During the last decade, increasing attention has been focused on ontologies and their use in applications related to areas such as knowledge management, intelligent information integration, education and so on. Ontology engineering aims at making explicit the knowledge contained within software applications, and within enterprises and business procedures for a particular domain. Ontology engineering, as a set of tasks related to the development of ontologies for a particular domain, offers a direction towards solving wide range of problems brought by semantic obstacles. According to that, ontology engineering could be a key aspect for improvements and better quality and success of our, already developed tutoring system but in traditional manner. Educational ontologies for different purposes were included in the new version of the system, such as for presenting a domain (domain ontology), building learner model (learner model ontology), presenting of activities in the system (task ontology), specifying pedagogical actions and behaviors (teaching strategy ontology), defining the semantics of message content languages (communication ontology) and specifying behaviors and techniques at the learner interface level (interface ontology) (Devedžić, 2006). A repository of ontologies must be built to achieve easier knowledge sharing and reuse, more effective learner modeling and easier extension of whole system. Ontologies are structured following the SCORM e-learning standard (SCORM, 2006). This ontological representation enables not only to represent meta-data but also reasoning in order to provide the best solution for each individual learner.

General ideas, of redefined ontology-based architecture for Protus 2.0 have been presented in Vesin et al. (2011). This architecture is based on experiences gained from similar web-based learning systems (Chen, 2008; Merino & Kloos, 2008; Šimić, 2004) and architecture for ontology-supported adaptive web-based education systems suggested in De Bra, Aroyo, and Chepegin (2004), Jia et al. (2011), Mizoguchi and Bourdeau (2000) and Fernández-Breis et al. (2012). All of the proposed architectures are highly modular with four central components: the application module, the adaptation module,
the learner model and domain module. In all of proposed architectures, the adaptation module is explicitly separated from the domain module, but another component is introduced in Protus 2.0 as in (Mizoguchi & Bourdeau, 2000) – the application module. This module is used for storing adaptation rules used in further personalization process. Fig. 4 depicts the general architecture of Protus 2.0, redesigned and extended version of Protus system.

The domain module presents storage for all essential learning material, tutorials and tests. It describes how the information content is structured. The learner model is a collection of both static and dynamic data about the learner. The system uses that information in order to predict the learner’s behavior, and thereby adapt to his/her individual needs. The application module performs the adaptation. To be exact, the adaptation module follows the instructional directions specified by the application module. These two components are separated in order to make adding new content clusters and adaptation functionalities easier. Within session monitor component, the system gradually re-builds the learner model during the session, in order to keep track of the learner’s actions and his/her progress, detect and correct his/her errors and possibly redirect the session accordingly. At the end of every session, the learner model is updated and then used along with other information and knowledge to initialize the next session for the same learner. The adaptation module contains rules for supporting the adaptive functionality of the system. An example for adaptive functionality is to decide whether a learner has sufficient knowledge to study a lesson/document (recommended for learning).

It is important to note that the original architecture of Protus did not bring any kind of homogenous representation of components. Each one was represented by different formats, using a variety of tools. The purpose, of our current research activities, is to represent each component of the system in form of the ontology. Each component will be responsible for specific tasks. According to that, the level of abstraction of this architecture will be higher (Jacinto & Parente de Oliveira, 2008). This approach will make it easier to understand the role of each component and, consequently, to promote interoperability among the components of the architecture. The developed system is highly modular, which allows better flexibility and future replacement of various components as long as they comply with the current interface. In the rest of the section, details of construction of different kinds of ontologies employed in Protus 2.0 system will be separately addressed.

5.1. Domain ontology

One of the main goals of the learning process is to understand and to acquire a body of knowledge for a given domain (Razmerita, Nabeth, Angehrn, & Roda, 2004). Domain model presents storage for all essential learning material, tutorials and tests. It describes how the content intended for learning has to be structured (Bork, 2001). Often the domain model can be structured as a taxonomy of concepts, with attributes and relations connecting them with other concepts, which naturally leads to the idea of using ontologies to represent this knowledge.

The complete Java course in Protus 2.0 contains several Concepts (lessons). Therefore, Java course contains: an introductory lesson, syntax, loop statements, execution control, etc. (Fig. 5). Each concept can be assigned any number of different resources (text files, images, animations, etc.). All resources are assigned depending on their Resource type. So we have: theory, examples, assignments, exercises, syntax rules, and so on. Therefore, Protus 2.0 contains different presentation methods (that supports different learning styles) for every lesson. For example, if learner has preference for Visual presentation method, Protus 2.0 will choose resource of appropriate Resource type.

In Fig. 5, an excerpt of a domain ontology covering basics of Java programming concepts with subConceptOf relationships between these concepts has been shown. This figure depicts the root concept with some of its sub-concepts: Syntax, LoopStatements, ExecutionControl, and Classes. The LoopStatements concept is further specialized and fine-grained into ForStatement, WhileStatement, and DoWhileStatement. Each concept can be assigned any number of different
resources. Specification of other relations between concepts will be useful for further personalization purposes.

An excerpt of ontology as resource topology is depicted in Fig. 6. The ontology depicts resource types in the programming domain. The most general resource type is DomainResource. DomainResource has three subtypes: CourseMaterial, AdditionalMaterial and ExaminationMaterial. Classes CourseMaterial and AdditionalMaterial represent the theoretical and practical explanations, respectively, that are displayed to the learners. ExaminationMaterial can be further specialized to Task and Exam. The Exam is consisted of various Tasks (Ivanović et al., 2008). CourseMaterial can be further specialized into Introduction, BasicInfo, Goals, Theory, SyntaxRule and ProjectAssignements, which corresponds to the main concepts and essential elements of a programming language course.

Previously mentioned ontologies further provides information for the Task ontology and the Teaching strategy ontology which will be explained in more details in Sections 5.2 and 5.4 respectively. This way, when the learner accesses a course, Protus 2.0 can infer which concepts could be suitable to be presented to the learner.

5.2. Task ontology

Task ontology is a system of vocabulary for describing problem solving structure of all existing types of tasks domain independently. It complements the domain ontology by representing semantic features of the problem solving (Devedžić, 2006). Task ontology specifies domain knowledge by giving roles to each object and relations between them. This ontology does not describe the content taught by the learning material. Instead, each class of the ontology stands for a particular instructional role for a learning concept.

Task ontology shows the role of specific resource from domain ontology. For example, if resource has fact or definition role it is used to increase basic knowledge and if its role is example, then it is used to increase learner's practical skills.

An excerpt of task ontology of resources in Protus 2.0 is depicted in Fig. 7. The ontology represents learning material grouped by the resources. The class Concept is used to annotate a unit of knowledge which is represented by some Resource. Details about resources are kept in Resource class instances. Each instance of Resource class contains basic information on individual resources, which are used for the subsequent selection of appropriate resources in the process of personalization. Specific type and role is determined for every resource.

Like in Dehors and Faron-Zucker (2006), concepts and resources are related by the hasResource property. Concepts can be arranged by the hasPrerequisite property. The hasPrerequisite property is proposed for navigational purposes. It allows pointing out concepts that must be known before starting to study a concept, and the concepts for which it is a prerequisite. Concept will not be covered unless that the prerequisite condition is satisfied. There can be
different sequence of resources that depends on navigational sequence determined for particular learner.

Resources play certain roles in particular resource fragments. For example, some resources represent the crucial information, while the others just represent a mean to provide additional information or a comparison. In the proposed ontology, we represent these facts by instances of ResourceRole class and its two properties: hasRole and supports. For example, resources like BasicInfo and Example have different roles. The role of the first is to represent introductory information for lesson and the role of the former is to provide additional information. On the other hand, both concepts support adaptation to learner with Reflective style of learning (Klašija-Miličević et al., 2011b). Resource properties can be further extended by assigning a ResourceType. Similarly, the resources roles can be further extended by specifying their types. Concepts, their types and resources form task ontology of Protus 2.0 system.

5.3. Learner model ontology

The learner model stores personal preferences and information about the learner’s mastery of domain concepts (Ullrich, 2004). The information is regularly updated according to the learner’s interactions with the content and is used by the Teaching strategy ontology to draw conclusions and decisions. The ontology illustrated in Fig. 8 offers the opportunity to map all information about the learner, starting from confidential data, like password, to a knowledge evolution history.

The class Learner is built from three components: Performance, PersonalInfo, and LearningStyle. These three classes are related to association through hasPerformance, hasInfo, and hasLearningStyle properties.

Class LearningStyle represents the preferred learning style for particular learner. This class offers four categories to the dimensions
of the Felder–Silverman Learning Style Model (sequential/global, active/reflective, visual/verbal and sensing/intuitive) (Felder & Solomon, 1996).

At runtime, learner interacts with a tutoring system. These interactions can be used to draw conclusions about possible his/her interests, goals, tasks, knowledge, etc. These conclusions can be used later for providing personalization. Ontology for learner observations should therefore provide a structure of information about possible learner interaction. Fig. 9 depicts such ontology as a part of Learner model ontology. Learner performance is maintained according to a class Interaction. Interaction is based on actions taken by specific learner, during specific Session. Interaction implies a Concept learned from the experience, which is represented by conceptUsed property. Interaction has a certain value for Performance, which is in this context defined as a floating point number and restricted to the interval from 1 to 5. This ontology is responsible for updating the Learner model ontology.

In order to accomplish this goal, the first task is to have a model of the learner with regard to learning style, therefore a learner modeling component is necessary (Popescu et al., 2007).

5.4. Teaching strategy ontology

Authoring of adaptation and personalization is actually authoring of learner models and applying different adaptation strategies and techniques to ensure efficient tailoring of the learning content to the individual learners and personalized task and navigation sequencing (Aroyo & Mizoguchi, 2003). The major goal of learning systems is to support a given pedagogical strategy (Dehors & Faron-Zucker, 2006). In this scope pedagogical ontologies can be associated with reasoning mechanisms and rules to enforce a given strategy. Often this strategy consists of selecting or computing a specific navigation sequences among the resources. Thus, formal semantics are required here to enable such computation.

Fig. 10 shows how the adaptation is carried out by the Teaching strategy ontology. The decisions are drawn on the basis of the information contained in the Condition class (that is generated by the information about learning style and performance of the learner) as well as teaching goals and previous behavior patterns. Class AdaptationType contains information about type of adaptation that is currently under way: styleMatch, adaptLink or navigation. These conditions are composed of data coming from several other components such as Learner model ontology, Task ontology and Domain ontology.

Personalization presents the choice of the most appropriate learning pattern or resource that will be recommended to the learner. This action is depending on many conditions but it implies only one decision. The decision determines what concept and resource the system is going to present for the learner.

5.5. Interface ontology

Interface ontology is result of the final stage of communication among the different components of the architecture. Fig. 11 shows the steps performed within the Interface ontology. System reads a decision from the Teaching strategy ontology, and based on that decision it creates navigation sequence of resources recommended for particular learner and generates an interface view to the learner. For example, if Decision includes recommending certain resource for learner than presentation will contain Resource of...
specific Resource type. On the other hand, if Decision contains data about recommended navigational pattern, than Protus 2.0 adds recommended Resource to current Navigational sequence.

Interface ontology can be used to specify the content of the pages or to standardize content and query vocabulary.

6. Implemented rules

Implemented ontologies have a set of basic implicit reasoning mechanisms derived from the description logic which they are based on but they need rules to make further inferences and to create useful relations on the represented pieces of information. Thus, rule systems require taxonomies in order to have a shared definition of the concepts and relations mentioned in the rules, and taxonomies require a rule system to derive/use further information that cannot be captured by them. Rules allow also adding expressiveness to the representation formalism, to reason on the instances, and they can be orthogonal to the description logic on which taxonomies are based on (Henze et al., 2004).

The proposed rules consist of an antecedent (body) and a consequent (head), each of which consists of a (possibly empty) set of atoms. Informally, meaning of the rule is: if the antecedent holds (is "true"), then the consequent must also hold. An empty antecedent is treated as trivially holding (true), and an empty consequent is treated as trivially not holding (false). Rules with an empty antecedent can thus be used to provide unconditional facts; however such unconditional facts are better stated in OWL itself, i.e., without the use of the rule construct. Non-empty antecedents and consequents hold if all of their constituent atoms hold, i.e., they are treated as conjunctions of their atoms. A typical SWRL rule is of the following form:

\[ a_1 \land a_2 \land \ldots \land a_n \land \rightarrow b_1 \land b_2 \land \ldots \land b_m \]

where \( a_i \) and \( b_i \) are OWL atoms of the following forms:

- **Concepts**, e.g., \( C(x) \), where \( C \) is an OWL description, in general, and \( x \) is either a variable, an OWL individual (facts about class membership, property values of individuals or facts about individual identity) or a data value.
- **Object properties**, e.g., \( P(x, y) \), where \( P \) is an OWL property and \( x, y \) are either variables, individuals or data values.
- **Datatype properties**, e.g., \( P(x, y) \), where \( P \) is an OWL property, \( x \) is variable or individual, while \( y \) is a data value.
- \( B(x_1, x_2, \ldots) \), where \( B \) is a built-in relation and \( x_1, x_2, \ldots \) are either variables, individuals or data values.
- \( \text{sameAs}(x, y) \) or \( \text{differentFrom}(x, y) \) where \( x, y \) are either variables, individuals or data values.

While the abstract SWRL syntax is consistent with the OWL specification, and is useful for defining XML and RDF serializations, it is rather verbose and not particularly easy to read (SWRL, 2012). In the rest of the section we will, therefore, often use a relatively informal human readable form similar to that used in many published papers.

In this informal syntax, a rule has the form:

\[ \text{antecedent} \Rightarrow \text{consequent} \]

where both antecedent and consequent are conjunctions of atoms written as \( a_1 \land \ldots \land a_n \).

Variables are indicated using the standard convention of prefixing them with a question mark (e.g., \( ?x \)). Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

\[ \text{parent}(?x, ?y) \land \text{brother}(?y, ?z) \Rightarrow \text{uncle}(?x, ?z) \]

In this syntax, built-in relations that are functions can be written in appropriate notation for functions, i.e., \( \text{op:numeric-add}(?x, 3, ?z) \) can be written instead as

\[ ?x = \text{op:numeric-add}(3, ?z) \]
During learning sessions, the most important system's task is execution of rules and the subsequent adaptation of the learner interface. Each rule is evaluated and, if all conditions hold, the body (action) of the rule is executed.

6.1. Adaptation rules

SWRL rules are one of the most popular forms of knowledge representation, due to its simplicity, comprehensibility and expressive power (Romero, Ventura, Hervas, & Gonzales, 2006). There are different types of adaptation rules depending on the knowledge they store. They are referred to as: decision rules, association rules, classification rules, prediction rules, causal rules, optimization rules, etc. Rules used in Protus 2.0 can be categorized as:

- **Learner modeling rules** that add knowledge about a learner, inferring new learner features from other already existing features of that learner. They are necessary for the identification of the learner style based on observed learning preferences.
- **Adaptation rules** that define the strategies of adaptation, taking into account domain features, system adaptation goals, user features, context and used presentation methods. They are necessary for content adaptation based on learning style and/or learning preferences (Popescu et al., 2007).

The idea is that the adaptation rule for reaching the adaptation goal can be defined taking into account the knowledge domain, the learners' current knowledge, his/her preferences and learning style. Also, the definition of adaptation rules requires considering the set of available adaptation methods and techniques (such as hiding text/links, link annotations, presentation methods, altering navigation sequences, etc.) (Carmagnola et al., 2005).

In the rest of the session we will present adaptation approach implemented in Protus 2.0 with some typical rules for adapting an e-learning course to the needs of the learners with different Felder–Silverman learning styles.

6.2. Learning styles identification

There are over seventy identifiable approaches to investigate and/or describe learning style preferences. We used one such data collection instrument, called Index of Learning Styles (ILS) (Felder & Solomon, 1996). The ILS is a 44 question, freely available, multiple-choice learning styles instrument, which assesses variations in individual learning style preferences across four dimensions or domains. These are Information Processing, Information Perception, Information Reception, and Information Understanding. Within each of the four domains of the ILS there are two categories (see Table 1):

- Information Processing: Active and Reflective learners,
- Information Perception: Sensing and Intuitive learners,
- Information Reception: Visual and Verbal learners,
- Information Understanding: Sequential and Global learners.

The learning style can be investigated by offering the learner the free choice between example, activity or explanation at first, and by observing a pattern in the choices he/she makes.

Before initial session, and after learning style has been determined by the ILS (Fig. 12), current learning style category of the particular learner must be written in Learner ontology of Protus 2.0. For example, if system determines that learner belongs to Active category within Information Processing domain, Protus 2.0 ontology should be updated with that fact:

\[
\text{Learner(?x) \land hasLearningStyle(?x,?y) \\
\land hasCategory(?y,?z) \land \\
isCategoryOf(?z,active) \\
\rightarrow hasLearningStyle(?x,active)
\]

At the beginning of every session Protus 2.0 requests information about the status of the course from the Learner model ontology for the particular learner (Fig. 8). This data includes information about the current lesson and the learning style category of learner within each of the four domains of the ILS. Request for appropriate resources which will be presented to the learner, based on this data, is sent to the Application module. Further, all activities of learners are monitored, as well as all requests he/she send to the system.

Personalization actions and presentation of lessons are determined by ILS and learners performance. If conditions (that personalization is based on) are generated by the learning style of the learner (Fig. 8) then rule system of Protus 2.0 starts personalization based on appropriate condition. For learner with active learning style, generated condition is act, for learner with reflective learning style generated condition is ref, etc. Examples of rules that implement those actions are:

\[
\text{Learner(?x) \land hasLearningStyle(?x,active) \\
\land Generates(active, ?z) \\
\land Condition(?z) \rightarrow Condition(act)
\]

\[
\text{Learner(?x) \land hasLearningStyle(?x,?y) \\
\land LearningStyle(\text{reflective}) \land Generates(?y, ?z) \\
\land Condition(?z)\rightarrow Condition(ref)
\]

Generated conditions are used for determining further actions which will be described in detail later in this section. After conditions are determined, Protus 2.0 makes appropriate decision for presentation based on adaptation type. There are three adaptation types, so far implemented in Protus 2.0: styleMatch (matching appropriate learning styles), adaptInterface (displaying/hiding interface elements) and navigation (altering navigation through course). Several rules for making decisions about further personalization are:

\[
\text{Personalisation(?p) \land basedOn(?p,?c) \\
\land Condition(act) \land CurrentGoal(?g) \\
isTypeOf(?g,\text{styleMatch}) \rightarrow determines(?p,act100)
\]

\[
\text{Personalisation(?p) \land basedOn(?p,?c) \\
\land Condition(ref) \land CurrentGoal(?g) \\
isTypeOf(?g,\text{adaptInterface}) \rightarrow determines
\]

\[
\text{Personalisation(?p) \land basedOn(?p,?c) \\
\land Condition(ref) \land CurrentGoal(?g) \\
isTypeOf(?g,\text{navigation}) \rightarrow determines(?p,ref100)
\]

Protus 2.0 makes a decision based on the current learning style of the learner and current adaptation type. Therefore, active instances of Decision class are determined. Decision named act100 is activated when learner’s current learning style is active and styleMatch adaptation type needs to be implemented. Decision named ref101 is activated when learner’s current learning style is Reflective and adaptInterface adaptation type needs to be implemented, etc.
Further personalization depends of Decision made by previous rules.

6.2.1. Information Processing: Active and Reflective learners

Within Information Processing domain it could be distinguished example-oriented learners, named Reflectors, and activity-oriented learners, called Activists. Active learners tend to retain and understand information best by doing something active with it – discussing or applying it or explaining it to others. Reflectors are people who tend to collect and analyze data before taking an action. They may be more interested in reviewing other learners’ and professional opinions rather than doing real activities. In Protus 2.0 system, a learner with the active learning style is shown an activity first, then a theory, explanation and example (Klašnja-Milicević et al., 2011b).

For example, in case of a specific perception modality preference, the recommended action would be to present the learner first with the preferred media type and then with the alternative representation types. Several rules that implement sequencing of resources are:

\[
\text{Resource}(x) \land \text{isTypeOf}(x, \text{excercise}) \land \text{Resource}(y) \land \text{isTypeOf}(y, \text{example}) \land \text{Decision}(\text{act100}) \rightarrow \text{hasPrerequisite}(y,x) \\
\text{Resource}(x) \land \text{isTypeOf}(x, \text{theory}) \land \text{Resource}(y) \land \text{isTypeOf}(y, \text{example}) \land \text{Decision}(\text{act100}) \rightarrow \text{hasPrerequisite}(x,y)
\]

Purposes of the previous rules are to define prerequisites among resources. Therefore, the first rule defines that exercise is prerequisite for presenting an example and meaning of the second is to assign priority for presenting the theory over examples.

For the learner with the reflective style this order is different – he/she is shown an example first, then an explanation and theory, and finally he/she is asked to perform an activity.

<table>
<thead>
<tr>
<th>Active</th>
<th>Reflective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work in groups</td>
<td>Work alone</td>
</tr>
<tr>
<td>Preference to try out new material immediately (ask, discuss, and explain)</td>
<td>Preference to take time to think about a problem</td>
</tr>
<tr>
<td>Practical (experimentalists)</td>
<td>Fundamental (theoreticians)</td>
</tr>
<tr>
<td>Sensing</td>
<td>Intuitive</td>
</tr>
<tr>
<td>More patient with details</td>
<td>More interested in overviews and a broad knowledge (bored with details)</td>
</tr>
<tr>
<td>By standard methods</td>
<td>Innovations</td>
</tr>
<tr>
<td>Senses, facts and experimentation</td>
<td>Perception, principles and theories</td>
</tr>
<tr>
<td>Visual</td>
<td>Verbal</td>
</tr>
<tr>
<td>Preference to perceive materials as pictures, diagrams and flow chart</td>
<td>Preference to perceive materials as text</td>
</tr>
<tr>
<td>Global</td>
<td>Sequential</td>
</tr>
<tr>
<td>Prefer to get the big picture first</td>
<td>Prefer to process information sequentially</td>
</tr>
<tr>
<td>Assimilate and understand information in a linear and incremental step, but lack a grasp of the big picture</td>
<td>Absorb information in unconnected chunks and achieve understanding in large holistic jumps without knowing the details</td>
</tr>
</tbody>
</table>
Meanings of these rules are analogous to previously mentioned. From the learners’ point of view, results of the firing these rules are displaying resources in adequate order. Examples of interface for learners with active and reflective learning style are presented in Figs. 13(a) and (b), respectively.

Other form of personalization for Information processing is link annotation. Every resource is given a certain role. For example, if resource has fact or definition role it is used to increase basic knowledge (preference of the reflective learners) and if its role is example, than it is used to increase learner’s practical skills (preference of the active learners). If certain role is predefined for learners with active learning style, than that resource is recommended to learner:

\[
\text{Learner(?x) \land hasLearningStyle(?x,active) \land \begin{cases} \\
\text{supports(?r,active)} & \text{if } ?c \text{ is active} \\
\text{supports(?r,active)} & \text{if } ?c \text{ is active}
\end{cases} \rightarrow \text{isRecommended(?r, true)}
\]

Similar rules are used for adaptation to other learning styles, too. isRecommended is a data valued property atom that consists of an OWL data property (recommended). With previous rule, recommendation status of that resource is set to true, therefore changes in user interface will be made. In this case, Protus 2.0 seeks resources that are predefined for learners with active learning style. For example, a learner with the active learning style can participate in activities such as quiz, chatting, and discussion options. Therefore, system annotate appropriate link that provides communication options (Fig. 14(a)).

6.2.2. Information Perception: Sensing and Intuitive learners

Within Information perception domain sensing learners, called Sensors, tend to be patient with details and good at memorizing facts and doing hands-on (laboratory) work. On the other hand intuitive learners, named Intuitors may be better at grasping new concepts and are often more comfortable than sensing learners with abstractions and mathematical formulations. Sensors often like solving problems by well-established methods and dislike complications and surprises. On the other hand, Intuitors like innovation and dislike repetition. Sensors tend to be more practical and careful than Intuitors. Intuitors tend to work faster and to be more innovative than Sensors. For example, it is assumed that sensing learners will be interested in additional materials, therefore they may click the button for additional material on the interface (Fig. 14(b)). Rules that implement these actions are:

\[
\text{Learner(?x) \land hasLearningStyle(?x,sensing) \land \begin{cases} \\
\text{supports(?r,sensing)} & \text{if } ?c \text{ is sensing} \\
\text{supports(?r,sensing)} & \text{if } ?c \text{ is sensing}
\end{cases} \rightarrow \text{isRecommended(?r, true)}
\]

These adaptation rules set value of recommendation attribute of specific instances of resources class to true. In the above case, rules are used to recommend resources to Sensor learner. If Recommended attribute for resource is set to true, it gives an information to Teaching strategy ontology of Protus 2.0 that particular resource is appropriate to learner and it could be presented to him/her. Intuitors are provided with abstract material, formulas and concepts. Adequate explanations are in form of block diagrams or exact syntax rules. Example of rule that implements those actions is:

\[
\text{Learner(?x) \land hasLearningStyle(?x,intuitive) \land \begin{cases} \\
\text{supports(?r,intuitive)} & \text{if } ?c \text{ is intuitive} \\
\text{supports(?r,intuitive)} & \text{if } ?c \text{ is intuitive}
\end{cases} \rightarrow \text{isRecommended(?r, true)}
\]

This rule is used to recommend syntax rule resource to Intuitive learner that results in adding appropriate tab in tabbed pane (Fig. 14(c)).

6.2.3. Information Reception: Visual and Verbal learners

Within Information Reception domain Visual learners remember best what they see - pictures, diagrams, flow charts, time lines, and demonstrations (Klašnj-Miličević et al., 2011b). Verbal learners get more out of words – written and spoken explanations. Protus 2.0 recommends appropriate resources with the following rules:

\[
\text{Learner(?x) \land hasLearningStyle(?x,visual) \land \begin{cases} \\
\text{supports(?r,visual)} & \text{if } ?c \text{ is visual} \\
\text{supports(?r,visual)} & \text{if } ?c \text{ is visual}
\end{cases} \rightarrow \text{isRecommended(?r, true)}
\]

The first rule is used for recommending resources that contain pictures and diagrams to Visual learner (Fig. 15(a)) while former rule recommends resource with written explanation to Verbal learner (Fig. 15(b)).
6.2.4. Information Understanding: Sequential and Global learners

Within Information Understanding domain, Sequential learners tend to follow logical stepwise paths in finding solutions. On the other hand Global learners may be able to solve complex problems quickly or put things together in novel ways once they have grasped the big picture, but they may have difficulty explaining how they did it. Sequential learners prefer to go through the course gradually, in a linear way with each step following logically from the previous one, while Global learners tend to learn in large leaps, sometimes skipping learning objects and jumping to material that is more complex. According to these characteristics of learning styles, Sequential learners go through Protus 2.0 lessons by in advance predefined order (Klašnja-Milićević et al., 2011b) while the Global learners are provided with the possibility to freely jump through the courseware. To define order of concepts, next rules are implemented:

Learner(?x) ∧ hasLearningStyle(?x,sequential) → 
   hasPrerequisite(loopStatements,syntax)
Learner(?x) ∧ hasLearningStyle(?x,sequential) → 
   hasPrerequisite(executionControl,loopStatements)
Learner(?x) ∧ hasLearningStyle(?x,sequential) → 
   hasPrerequisite(classes,executionControl)
etc.

Where hasPrerequisite is a data valued property atom that defines prerequisites among resources. Based on the defined prerequisites, Protus 2.0 can make decision whether to present one lesson in time in sequential order (for Sequential learners) or to present links to all lessons at once to learner (in case of Global learners). In the first case, the interface elements for sequential navigation (in our case the buttons for Next/Previous resource/lesson) will be shown (Fig. 16). On the other hand, interface elements for non-sequential navigation will be added (Fig. 17) for Global learner.

6.3. Learner modeling rules

This section describes some examples of methods that can be used as simple cases for some legal interpretation of learning style modeling. The adaptive feedback is usually based solely on an initial assessment of the learning style profile, which is then expected to remain stable. However, research indicates that learning styles of an individual can vary depending on the task and update the knowledge base (Chi, 2009). The above SWRL rules can be executed using the JESS rules engine after providing the factual knowledge. After firing the rule, the inferred knowledge can be written back to the ontology repository and update the knowledge base (Chi, 2009).

Meaning of the previous rule is: if the learner interacts with specific concept and during that interaction he/she took the test and earned specific grade, than system should memorize that learner's performance. In addition, isLearned property of that particular concept should be set to true.

If learner does not provide required level of performance results within session with presentation method used for certain learning style category, his/her current learning style category will be modified by next rule:

Learner(?x) ∧ hasInteraction(?x,?y) ∧ 
   hasInteraction(?x,?c) ∧ Concept(?c)
   ∧ conceptUsed(?c,?p) ∧ Performance(?p)
   ∧ hasResult(?y,?p) ∧ 
   hasGrade(?p, grade) ∧ swrlb:lessThan(grade, required) → 
   hasLearningStyle(?x,visual)

Meaning of the previous rule is: if learner does not provide required level of performance results within session with presentation method used for certain learning style category, his/her current learning style category will be modified.

For example, if learner with Verbal learning style interacts with system and during that interaction he/she had accessed appropriate concept but not earned sufficient grade (required grade level is kept in global value required), than, learning style of that learner should be changed to other style from Information reception domain: Visual learning style. That implies that in next session, learner will be presented with resources that are defined to support that new learning style category.

Similar rules will be executed for other categories of learning styles (intuitive/sensing, global/sequential and active/reflective).

The above SWRL rules can be executed using the JESS rules engine after providing the factual knowledge. After firing the rule, the inferred knowledge can be written back to the ontology repository and update the knowledge base (Chi, 2009). The four preparation steps were:

1. developing an ontological knowledge model by gathering expertise related to learning objects and content sequencing
2. representing the domain model using OWL ontology
3. establishing individual relationships and adaptation rules using SWRL and
4. asserting factual knowledge based on the ontology schema.

The above processes provide the basis for building an initial ontological knowledge base. Both description logic reasoning and the rules engine were further applied to extend the inference power to establish a runtime knowledge base.
Based on the implemented ontologies and adaptation rules described earlier in the paper, Protus 2.0 personalize user interface for particular learner. Personalization includes presentation recommended components or links to them based on the rules described in the Section 6. Fig. 19 presents user interface of Protus 2.0, personalized for learner that belongs to Reflective, Intuitive, Verbal and Global categories within Informational processing, Information perception, Information reception and Information understanding domains, respectively.

In our opinion the main achievements of research results presented here are:

- Explicit representation of the ontologies and rules, encouraging their understandability, maintainability and reusability. The representation of each component in Protus 2.0 is made by a specific ontology, making possible a clear separation of the components of tutoring system (adaptation, application and
domain module, session monitor and learner model) and explicit the communication among them. Rules provide explicit definition of all personalization activities in the system.

- Separation of knowledge about learning styles as modularized sets of rules. For every personalization action in Protus 2.0, appropriate rules were defined as a consequence of learning style identification. Therefore, Protus 2.0 contain rules for presenting personalization options for all four domains of learning style investigation.

- Component-based definition of adaptive tutoring system that uses first-order logic to perform personalization. All personalization options are presented in the form of SWRL rules. These adaptation rules use data from domain, task, learner model and teaching strategy ontology to perform personalization.

- Facilitation of appropriate implementation of the rules in an adaptive tutoring system. With this approach we hide a lot of functionalities behind the rules. This way, maintenance of adaptation rules is sufficient for modifying personalization options in Protus 2.0.

These ontologies may serve as a good starting resource that can be further extended and modified in order to define completely at the ontology for the domain of adaptive systems.

7. Conclusion

In this paper we presented how Semantic web technologies and in particular ontologies can be used for building Java tutoring system. Architecture for such adaptive and personalized tutoring system that completely relies on Semantic web standards and technologies was presented. The form of several ontologies has been introduced which correspond to the components of a tutorial system: domain ontology, task ontology, learner model ontology and teaching strategy ontology. For generating presentation structures, learner interface ontology has been introduced.
We also presented an ontology-based approach to:

- employing web ontology language (OWL) to describe context knowledge,
- using Semantic web rule language (SWRL) to represent rule-based inference rules for context reasoning,
- use of JESS to make inference over the total environmental information.

This ontology-based approach allows implementation of adaptation customized to different requirements. The learner demand is derived from the knowledge contained in the ontology. Various conditions are captured in the body of SWRL rules. As a result of the firing of rules, recommendations in the form of various content presentations are generated, which can be used to implement the concept of adapted content and adapted navigation. In addition, the rules can be modified for specific adaptation requirements. For example, we could add new types of resources that will support some of the particular learning styles. In that case, we must modify appropriate adaptation rules in order to provide information to system, which resources to recommend to learner.

From the presented paper it follows that ontologies will fundamentally change the way in which systems are constructed. Today, knowledge bases are still built with little sharing or reuse – almost each one starts from a scratch. In the future, intelligent systems developers will have libraries of ontologies at their disposal. Rather than building from scratch, they will assemble knowledge bases from component drawn from the libraries. This should greatly decrease development time while improving the robustness and reliability of the resulting knowledge bases.

The explicit conceptualization of system components in a form of ontology facilitates knowledge sharing, knowledge reuse, communication and collaboration and construction of knowledge rich and intensive systems. There are some methodologies how to construct and maintain ontologies and how to use them in practical applications, but this is rather a set of recommendations then exact theories. Despite this, ontologies are being more and more widely used for construction of systems that require an explicitly encoded knowledge. Therefore, those systems are able to interoperate better providing learners with an extended access to information resources.

For future work, we plan to present results of utility of Proton 2.0 in classroom. Plan is also to incorporate additional courses from other domains and to consider the shared-knowledge among users so that learners could recommend and/or tag content to other learners. These actions will make the system more comprehensive and intelligent.

References


