Semantic Web and Intelligent Learning Management Systems

Goran Šimić, Dragan Gašević, Vladan Devedžić

FON – School of Business Administration, University of Belgrade
POB 52, Jove Ilića 154, 11000 Belgrade, Serbia and Montenegro
gshimic@eunet.yu, gasevic@yahoo.com, devedzic@galeb.etf.bg.ac.yu

Abstract. This chapter emphasizes integration of the Semantic Web technologies in intelligent learning systems by giving a proposal for an intelligent learning management system (ILMS) architecture we named Multitutor. This system is a Web-based environment for the development the e-learning courses and for the use of them by the students. Multitutor is designed as a Web-classroom client-server system, ontologically founded, and is built using modern intelligent and Web-related technologies. This system enables the teachers to develop tutoring systems for any course. The teacher has to define the metadata of the course: chapters, the lessons and the tests, the references of the learning materials.

1 Introduction

Two groups of the adaptive education systems are the most frequently used on the Web. Those are Adaptive Hypermedia (AH) and Intelligent Tutoring Systems (ITSs). The AH systems are focused on non-linear and adaptable structure of the educational materials [4]. AH systems provide to the user easy navigation, referencing and global view to the content. Also, they provide presentational adaptation techniques (the conditional or stretch text, variants of pages and fragments, and frames linked to the concepts). Both of them (AHS and ITS) are narrow focused on the specific area of one domain. While the AH systems have compact system design with high coupled components [3], the ITSs have high-level modularity. ITSs provide user (student) oriented design and much more pedagogical knowledge implemented in the system. Today there are many AH and ITS stand-alone systems that are used for similar educational tasks. The same knowledge is developed at the same time on the different places. This is the typically waste of domain experts' time. Therefore these systems are usually expensive and can not be used without license, payment or/and registering.

The learning management systems (LMSs) are much more successful in Web-enhanced education (related to a number of users). LMSs are integrated systems that support a number of teachers' and students' needs. LMSs provide a teacher to compose their courses from newly created and existed learning units (so called learning objects - LO). These objects are modeled and described by standard structure and metadata. This means that LOs would be reused in many courses and for different purposes. The standardization means that an LO could be found on the different locations on the Web, and semantically can be connected in the number educational structures in the same time.

Intelligent LMSs (ILMSs) are bridge the gap between the modern approach to Web-based education based on learning management systems and powerful but underused intelligent tutoring and adaptive hypermedia technologies [4]. The reusing ITS supported domains in more courses can be realized by the well-described knowledge. This knowledge has to be expressed in a precise, machine-interpretable form and enables the interoperable application components to process LO data both on the syntactic and semantic level [6]. The Semantic Web, a recent Web community effort, is a promising technology for improving semantic interoperability of LOs [19]. The main part of the Semantic Web are domain ontologies that should provide a formal description for a shared domain conceptualization. As the new Web generation, the Semantic Web has better conditions for composing and reusing learning materials. The Semantic Web can be seen as an opportunity to enhance the metadata associated to learning materials, expanding the possibilities of current e-Learning specifications and standards.

In this paper we try to explain the main characteristics of the ILMSs, and show our approach to create an ILMS called Multitutor as a Semantic Web enabled system. In the next section we give an overview of ILMSs and identify their shortcomings regarding interoperability. Section three explains motivation as we as the Multitutor architecture while section four shows the Multitutor implementation in detail. Section five discusses how can we benefit from current research in using Semantic Web technologies for e-learning.

2 ILMS – General Concepts and Applications

Nowadays, there are many different ITSs and LMSs. But the educational needs are not yet satisfied. There is no interoperability between these systems. The main problem is that every kind of data on the Web is poorly structured. The existing structures do not have a standardized format. In the last years the community tries to define the ontology of different kinds of knowledge [16]. The great task is that the existing systems accept those standards and modify their data and applications accordingly to standard representations and interfaces.

The ILMS structure is based on the structure of both ITSs and LMSs. As with ITSs, in the ILMS there are modeling and representation of relevant aspects of knowledge. This means that it contains the knowledge about a student, the domain, the pedagogy and the communication, that are involved. The general concepts that support the above knowledge aspects are implemented as components of ITS architecture. There are five basic ITS modules: *student model, domain knowledge, pedagogical module, expert model and communication model* (for details about each of these modules see [1]). ITSs have high intelligent performances. The level of intelligence of an ITS is proportional to the possibility of the student model to describe the skills and knowledge of the real student. The educational contents that the system delivers to the student are based on this model. If the student model contains wrong or incomplete students' profile, the ITS actions would complicate the student learning efforts. Today, this model has to support more sophisticated student properties. These properties are: student interests, educational goals, motivation, social and cultural environment, predisposition, psychological characteristics and many others. If system reactions are based only on the students' results, the system behavior will not be appropriate to the real students' needs. The student model is the ITS meta-knowledge about the students (in general). The concrete instances of the student model represent the systems' knowledge about the individual students. An ITS is better if it contains more stereotypes of students' model. Reusability of these entities can be supported by a student ontology.

The cost of high intelligent performance is that many ITSs are strongly focused on one domain. Most of ITSs have a disadvantage that their knowledge base (KB) is only used inside the concrete ITS environment. Therefore these systems do not need a standard representation of their domain knowledge. Usually, a KB is implemented through the rules or constraints. It is also annotated in one kind of script files that are readable only for specified ITSs. This KB can not be used frequently by other systems. Only ITSs that support appropriate script format can reuse this knowledge. Another problem is that the knowledge is no described by standard format.

On the other side ILMS inherit the design (building) of learning materials and management abilities from LMSs. While ITSs are concerned about the adaptation to learning possibilities of one student, LMSs are mainly focused on reusability of LOs, and execution of collaborative and administration tasks. ITSs are educational software, which is finalized, and they enable students to improve their skills and knowledge. If a teacher wants to change the learning contents, (s)he has to use an appropriate authoring tool. LMS s support this scenario.

LMSs provide a complete platform in the areas of logging, assessing, planning, delivering contents, managing records and reporting. They improve both the self-paced and the instructor-led learning processes. All these activities are represented to the end user (or organization) as a group of Web services. The LMS architecture has a layered organization as it is shown in [13]. LMSs are poorly Web oriented systems that are hosted on both Web and application servers. In fact, LMSs are high-distributed systems over the Web. One course presents an integrated structure of many learning resources that can be hosted on different Web locations. The same resources can be combined with others in different courses. Also, more student groups can learn many courses at the same time. In these conditions, the system must have powerful management features. This means that an ILMS needs specialized ITS properties and the capacity to perform the described administration, integration and

distribution tasks as LMSs. To be more precise, an ILMS has the aggregated structure of the LMS framework enriched by embedded core of ITS (see Figure 1).

The ILMS general architecture consists of three basic parts: *administration tools*, *teacher tools,* and *student tools*. The administrative tools support the realization of different management tasks. For example: maintenance of student and teacher records, administration of the domain knowledge and the system security protection.

The teacher tools of the system help teachers to create LOs, combine them with existing LOs and compose the courses. A teacher is responsible for entering students' data and giving the system students' profiles (by creating a specific student model). Domain experts can design the domain ontology that should describe and structure the knowledge. The teacher package provides the monitoring of student results that teachers can use to track student sessions with an ILMS.

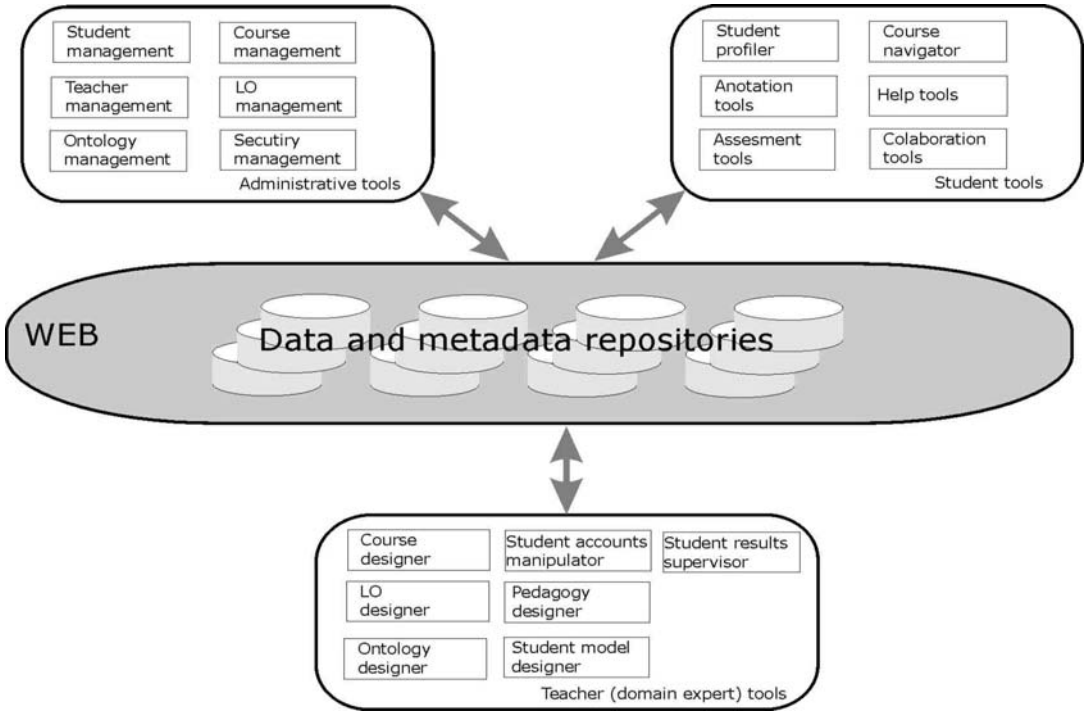

Fig. 1. The ILMS Architecture

The student tools generally help students to master the knowledge. The system enables a student to declare his interests, favorites, predisposition and real skills. These data help the system to initiate a student model and determine a student stereotype. While the student uses the system, different tools provide her/him navigation through the learning space, marks for important things, contextual help and skills measurement. The student can also collaborate with other students, teachers and experts. This is a way that an ILMS provides high cohesion and synergy of efforts from all the subjects in the learning process. The system knowledge is transparent and distributed on the Web. It becomes possible to use concepts of the Semantic Web integration process in the adaptive composing of learning materials. Different specialized pedagogical knowledge becomes accessible for all interested systems over the Semantic Web. Note also that current LMSs like Blackboard CourseInfo or WebCT cannot be easily made intelligent educational systems not only because they lack ontological support [7]. They also lack intelligent learner modeling, reasoning and adaptivity, although they do provide presentation and management of learning material and scenarios, as well as database management and administration of learners.

3 Multitutor: An ILMS

In this section we are trying to present an ILMS named Multitutor. This system is a product of three years research efforts. We started with a single user application, so called *Code Tutor* [18]. This is a

small Web-based tutor designed for fast students' briefing in the area of radio-communications. Our learners are telecommunication college students. The first version of Code Tutor has been actively used in classroom since mid-2001. The teachers' opinion is that it is very useful tool, and the students favor this kind of learning.

These facts have motivated us to build a new version, which will provide students to communicate with the system through standard Web browsers. The entire system is implemented in Java, using many different current technologies: the CLIPS tool was used for building ES knowledge base files, i.e. Code Tutor's domain knowledge, Java-based ES shell *Jess* was used to interpret these files, *Java™ Servlet* technology to implement the system's interactions with the students, *Apache HTTP* server to store static HTML pages, *Apache JServ* to interpret the servlets, and *XML* technology to generate course description files that Code Tutor uses to provide recommendations to the students. Code Tutor is actually Web-enabled and Web-ready, intended primarily for use in the classroom, rather than a full-fledged Web-based ITS built to be used adaptively over the Web.

Our opinion is that we developed a domain independent system that provides a useful environment for many courses. This way, we avoided the disadvantage of a rare use of the system. Our goal is to attract many teachers to use Multitutor. Therefore we expect a faster development of this system.

We tried to design an authoring tool that is a part of the Multitutor system. The component called *Course Designer* (Figure 2) is designed for this purpose. This tool is accessible to the teachers that want to create their course. We also attempted to formalize the course ontology by using standard describing and structuring format. Our selection is XML as a well-structured format for wide area purposes. The Multitutor system would be sorted in teacher-oriented tools. It provides a course creation without implementation details and course design using appropriate wizards. The Multitutor is a Web-based client-server system. This means the learning content is distributed to the students via the Web server. The user is on the client side and (s)he accesses to the learning resources using the Web browser. The Client sends the request through HTML page. The Web server forwards this request to the application server. The application server processes the request and returns the results usually in the form of dynamically generated HTML page. The Web server dispatches this page to the appropriate client.

The students can access any Web portal where they have an account. There are three actors in the use-cases of Multitutor: *administrator, teacher*, *and student*. The administrator executes management tasks in the system. The teacher tasks are well known. A teacher can create his own courses. These courses can be about different domains. Like as in the LMS, every moment the teacher can monitor his students' results. He can modify the learning contents during the students learning. Students are organized into groups (classes) and they access to the courses accordingly to their group. Their communication with the system (logging the system, customizing the interface, learning the course chapters, solving the tests and accepting the skills level and recommendations) runs over the Web browser. The system is designed to support changeable navigation possibilities to the student. It provides the dynamic creation of the learning materials.

The servlet engine represents the application server. The servlets (java classes) play the role of the front end of the application. They can refer the functional calls to the middle layer classes. As shown on the model, the core of the system is the tutor concept. The tutor is the main part of the system architecture. It is the system coordinator, dispatcher and monitor at the same time. The pedagogical strategies are implemented in the tutor. It analyzes the data of the student model (model of particular student) and uses its teacher knowledge to require the proper learning contents. The expert module maintains the references of domain knowledge and rule base. The reasoning machine processes the request of the tutor and composes the learning content. This content can include the text, the picture or some other multimedia. In the test phase the content is represented by the test sets or by the problems that students have to solve. These contents the tutor sends back to the servlets.

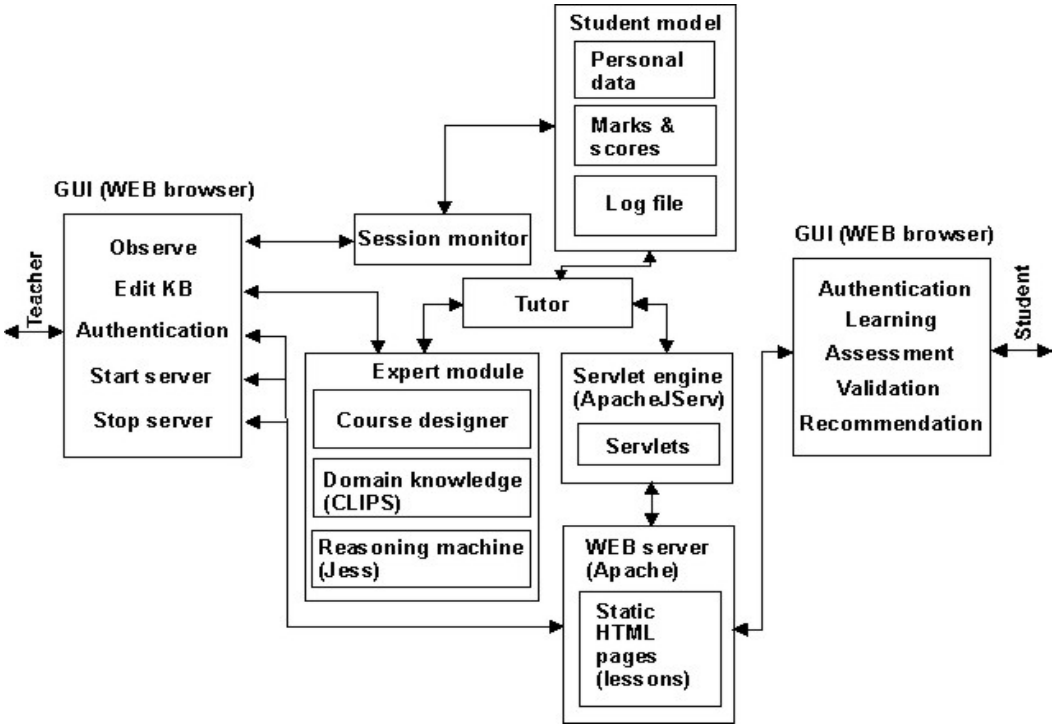

Fig. 2. The Multitutor architecture

4 Implementation – Multitutor

Based on the low coupling components of the system architecture, the entities are grouped (like a packages) by the functions and data contentment. This section tries to explain the distribution of the metadata.

4.1. The Initial System Data

When the system is in use, the tutor module creates a separate instance for every logged student and updates them during the student sessions. The Web server is responsible for delivering the learning contents to a particular student. The initial data that Multitutor uses during the starting phase are stored in the same place (in one file). This file contains the data about the teachers, courses and student groups.

These data provide two things: one is about the registered users (teachers and students) that can use the system, and the other is the path to the course ontology. The initial data are structured to relate teachers, classes (student groups) and courses. The conceptual model (Figure 3) that abstracts these relations and it can be translated in the basic system ontology [5].

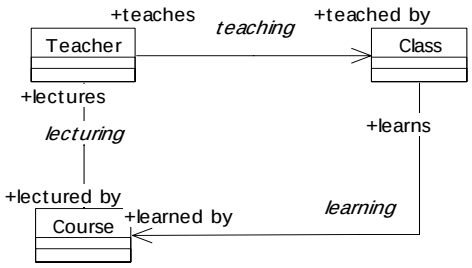

Fig. 3. The general concepts of the learning process

The teacher concept is used in the teacher application. There are two cases: when the teacher creates the course, or when he searches the students' results. This way the teacher looks at the results of his

students (classes) only. This model can be converted in an ontology schema that is readable for another part of the application logic. We used XML Schema to create the ontology vocabulary. All the elements are globally defined in the XML Schema definition document. A relation between classes is not defined as an attribute of a class, but as an independent entity, which have a certain domain and range.

4.2. The Basic Concepts of the Course Ontology

The course is an aggregated structure that contains the learning material, the references and the content for assessment. The learning material is structured on the learning objects, which are named *chapters* and *lessons*. Every course is divided on the chapters. Every chapter is divided on the lessons. The lesson is the basic learning unit. One lesson is related to one LO. The learning object is an aggregated structure that consists of the following classes: domain *concept*, *explanation* of the concept, the *learning content* and the *test set* (see Figure 4). This way one LO can be used to create many lessons in the different courses. The LO describes one concept of domain. The concept is related to the explanation, one or more test sets and to the learning contents. The *LearningContent* class represents the multimedia content of the learning object. Depending on different students' knowledge levels the different content will be presented to the student. The concept is self-related. This means one concept is the analogy of some other. The lesson is self-related too. One lesson is the prerequisite to the some other.

The test set is the collection of the questions and related answers that the system uses to assess the students' knowledge about one concept. The Multitutor offers the answers to the student. The answers have the marks or the true/false statement. This means the level has to be precisely defined by the course creator (teacher). One LO on the specified level can have number of questions. This way the student gets different questions every time when he repeats the test.

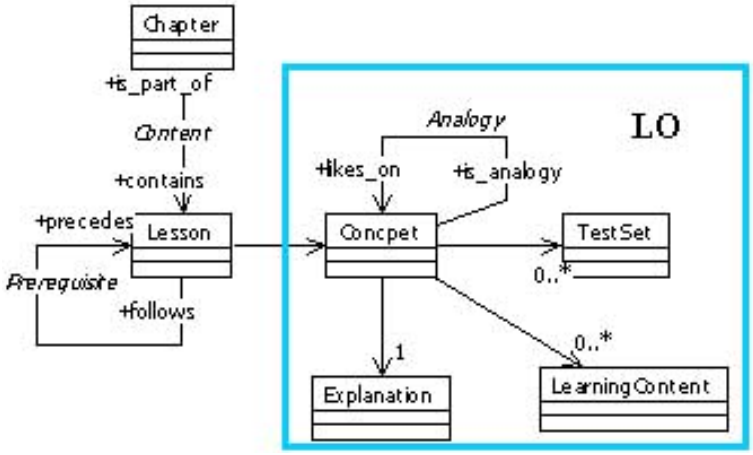

Fig. 4. The main concepts of the course ontology

The entities that are self-related can play different roles. In the next example (Figure 5), there are two lessons in the course *Physics* file (the chapters of the course are not shown). The analogy is similarly to prerequisite. This self-relation can be used when the student can not pass the tests about the main concept. Then the system tries to explain this concept by the similar one. If the student can not understand the concept of sound waves, the Multitutor helps him by the similar explanation about the water wave. The main goal of analogy is to explain the main concept on the other interesting way. The strong recommendation to the teachers is to use the simpler concepts for the analogies.

```
<?xml version="1.0"?>
<Ontocourse>
    <Course>
        <Name>Physics</Name>
        <!-- ...-->
        <Lesson>
            <Name>Sound Wave</Name>
            <Prerequisites>
                <Lesson>
                    <Name>Wave motion</Name>
                    <!-- ...-->
                </Lesson>
            </Prerequisites>
            <!-- ...-->
            <Concept>
                <Name>Sound Wave</Name>
                <Analogies>
                    <Concept>
                        <Name>Water Wave</Name>
                    </Concept>
                </Analogies>
                <!-- ...-->
            </Concept>
        </Lesson>
    </Course>
    <!-- ...-->
</Ontocourse>
```

Fig. 5. The fragment of the course data

4.3. The Student Model

The student model has a separate ontology that is shown in Figure 6. This structure has four parts: *the basic student data*, the student stereotype, *students' real skills* (based on the scores) and *the skills that are estimated by the system*. One student can have different skills because he studies many courses. The stereotype holds the sophisticated data about students' interests, favorites, interface customization, the rate of progression, the learning paths, but also data about the most frequently faults. The stereotype is very important for the determining of pedagogic strategy (in the pedagogic module).

The relations are uniformly propagated through the model in the student ontology. Multitutor sorts a student in one stereotype. The student skills are determined when the student starts to use the system. During the first session the student gets the questionnaire and the pretest. Those results are used to predict the student success and they are represented by the *ProjetedSkill* concept of the model. While the student learns the course the system monitors the students' navigation and time which is spent on the studying every particular concept. The student gets the tests and Multitutor serializes the results. The *MeasuredSkill* concept provides the correlations of the students' data. Those data are processed by the expert module and the conclusions are used by the pedagogical module to compose the next learning content.

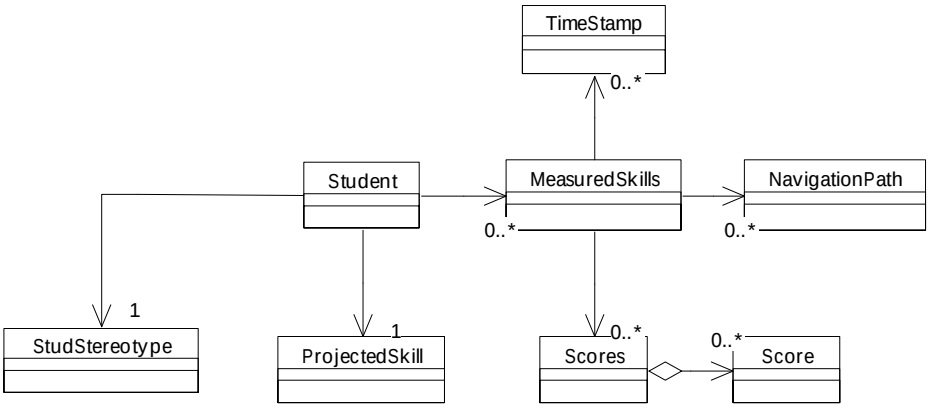

Fig. 6. The student ontology

4.4. Mutitutor Applications

In Multitutor we have developed the Code Tutor educational systems for teaching radio-communications that we have already mentioned. In order to illustrate how Multitutor can be used for Semantic Web learning applications we show a simple Petri net educational system. However, if we want to use Petri net model in Multitutor we should prepare suitable equipment. In our case we use the Petri net infrastructure for the Semantic Web [11] consisting of: Petri net ontology, P3 – a Petri net tool for creating learning materials and the Petri net Web Service.

5 Future Improvements

We have so far shown the main features of the Multitutor system as well as examples of two learning applications developed in the Multitutor. We especially stressed how the Multitutor describes metadata regarding their interoperability. Accordingly, we have explained three XML Schemas that describe: 1. The whole system, 2. Courses, 3. Student models. However, the XML Schema mechanism itself has several weaknesses regarding the ontology description [14], so in the future Multitutor versions we should improve some of them. The main point is to use the Semantic Web ontology languages (e.g. RDF(S) and OWL) as well as e-learning initiatives and proposals based on those languages. Here we shortly elaborate some important experiences that can be useful for the future Multitutor improvements.

Edutella is a democratic (peer-to-peer) network infrastructure for search and retrieval of information about learning resources on the Semantic Web [17]. Brase and Nejdl showed how ontologies could be exploited to enhance LO metadata in Edutella [2]. They gave an example of an ontology developed in accordance with the ACM Computer Classification system (ACM CSS). This ontology was described with RDF, and used in the Edutella system. The ontology improved the searching for leaning objects and it would be a useful for Multitutor. The navigation through learning materials as well as their findabilty can be improved by topics maps [9]. Topic maps provide a language to represent the conceptual knowledge with which a student can distinguish learning resources semantically. Moreover, topic maps are very suitable for representing the course unit ontological structure.

The EU/ITS project ELENA (http://www.elena-project.org/) tries to provide solutions for personalization, openness, and interoperability in the context of smart spaces for learning [10]. This project emphasize that we should use appropriate standards to describe a learner profile. Examples of attempts to standardize a learner profile are IEEE Personal and Private Information (PAPI) (http://ltsc.ieee.org/wg2/) and IMS Learner Information Package (LIP) (http://www.imsproject.org/profiles/index.cfm). Taking into account these two standards the authors' of the Elena project developed the learner ontology. The ontology keeps information about appropriate learning resources which are relevant with respect to user interests, user performance in different courses within one domain or even different domains, user goals and preferences, etc. This ontology in the RDFS form is available at http://www.learninglab.de/~dolog/learnerrdfbindings/. Another useful direction for describing student models in Multitutor as well as on the Semantic Web is the User Modeling Markup Language (UserML) [12]. UserML is an ontology-aware XML vocabulary defined by the UserOL ontology.

Several Educational Modeling Languages (EMLs) have been recently emerged. One of EML definitions states that an EML is a semantic notation (i.e. metamodel or ontology) for units of learning to be used in e-Learning [15]. They have XML binding and they are pedagogically flexible. The final result of an EML should be an instructional model with the following segments: content, didactical (e.g. sequencing) and presentational [20]. These EMLs attempts can be used as guidelines how Multitutor courses can be described in the future. In fact, we can use an EML instead of the Multitutor's course ontology.

Note that the learning technology community lacks standardized-ontologies for all these described aspects. However, all these efforts give useful guidelines for the future improvements. We believe that a solid starting point for new Multitutor versions is to use RDFS defined annotations instead of current XML Schema based formats.

6 Conclusions

In this chapter we tried to explore development of ILMSs for the Semantic Web. As result of our research we developed Multitutor an ILMS that uses XML-based technologies (i.e. XML Schema and XSLT) in the combination with the well-proven tools for developing intelligent systems (i.e. Jess). Our first experience with Multitutor is encouraging from both students' and teachers' sides. However, our ILMS needs further changes in order to better exploit the Semantic Web benefits (e.g. we should use RDFS or OWL definitions of both course and student ontologies rather that current XML Schema definitions). Of course, some recent solutions of the use of ontology development and Semantic Web languages for e-learning (e.g. Edutella, Elena, UserML, Topic Maps, etc.) can be very useful in this direction. Note that many author in the e-learning community defined ontologies of different kinds of knowledge in the last years. But, this raises many problems for developers as if which solution is the most appropriate. Accordingly, the main challenge for the e-learning community is to adopt standard Semantic Web ontologies [8] that will be guidelines for the developers of LMSs/ILMSs.

References

1. J. Beck, M. Stern, and E. Haugsjaa, "Applications of AI in Education," *ACM Crossroads*, Vol. 3, No. 1, 1996, pp. 11-15.

2. J. Brase, W. Nejdl, "Ontologies and Metadata for eLearning," *In S. Staab & R. Studer (Eds.) Handbook on Ontologies*, Springer-Verlag, 2004, pp. 555-574.

3. P. Brusilovsky, "Adaptive Hypermedia," *User Modeling and User-Adapted Interaction*, Vol. 11, No.1-2, 2001, pp. 87-110.

4. P. Brusilovsky, "A Distributed Architecture for Adaptive and Intelligent Learning Management Systems," *In Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems*, 2003,Sydney, pp. 5-13.

5. R. A. Calvo, "User Scenarios for the design and implementation of iLMS," *In Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems,* 2003, Sydney, pp. 14-22.

6. V. Devedžić, "Web Intelligence and AIED," *In Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems,* 2003, Sydney, pp. 23-33

7. V. Devedžić, "Key Issues in Next-Generation Web-Based Education," *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 33, No. 3, 2003, pp. 339-349.

8. V. Devedžić, "Think ahead: evaluation and standardization issues for e-learning applications," *International Journal of Continuing Engineering Education and Lifelong Learning*, Vol. 13, No. 5/6, 2003, pp. 556-566.

9. Ch. Dichev, D. D. Dicheva, and L. Aroyo, "Topic Maps for E-Learning," *International Journal on Advanced technologies for Learning*, ACTA Press, Vol. 1, No. 1, pp. 1-7, 2004.

10. P. Dolog, N. Henze, W. Nejdl, M. Sintek, "Personalization in Distributed eLearning Environments," *In Proceedings of the 13th International World Wide Web Conference*, NY, USA, 2004.

11. D. Gašević and V. Devedžić, "Reusing Petri Nets Through the Semantic Web," *In Proceedings of the 1st European Semantic Web Symposium*, Heraklion, Greece, 2004.

12. D. Heckmann, A. Krueger, "A User Modeling Markup Language (UserML) for Ubiquitous Computing," *In Proceedings of the 9th User Modeling Conference*, Johnstown, Pennsylvania, USA, 2003, pp. 393-397.

13. iCMG Learning Management System (LMS) Architecture (May 25, 2004) [Online]. Available: http://www.icmgworld.com/corp/ces/ces.lms.asp

14. M. Klein, "XML, RDF, and Relatives," *IEEE Intelligent Systems*, Vol. 16, No. 2, March/April 2001, pp 26-28.

15. R. Koper, "Educational Modeling Language: adding instructional design to existing specifications," *Workshop "Standardisierung im eLearning"*, Frankfurt, Germany, 2002.

16. R. Mizoguchi and J. Bourdeau, "Using Ontological Engineering to Overcome Common AI-ED Problems," International Journal of Artificial Intelligence in Education, Vol. 11, 2000, pp. 1-12.

17. M. Nilsson, M. Palmér, and A. Naeve, "The Edutella P2P Network - Supporting Democratic E-learning and Communities of Practice," *in McGreal, R. (ed.) Accessible education using learning objects*, Taylor & Francis Books Ltd., London, UK, 2003, to be published.

18. G. Šimić, V. Devedžić, "Building an intelligent system using modern Internet technologies," *Expert Systems with Applications*, Vol. 25, No. 2, 2003, pp. 231–246.

19. Lj. Stojanović, S. Staab, R. Studer, "eLearning in the Semantic Web," *In Proceedings of the World Conference on the WWW and the Internet (WebNet 2001)*, Orlando, Florida, USA, 2001.

20. F. Weitl, C. Süß, R. Kammerl, B. Freitag, "Presenting Complex e-Learning Content on the Web: A Didactical Reference Model," *In Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*, Montreal, Canada, 2002, pp. 1018-1025.