7. Intelligent Virtual Teaching

Goran Šimic, Dragan Gaševic, Zoran Jeremic, and Vladan Devedžic

FON – School of Business Administration, University of Belgrade, POB 52, Jove Ilica 154, 11000 Belgrade, Serbia and Montenegro

There are two major groups of existed adaptive education systems: Adaptive Hypermedia (AH) and Intelligent Tutoring Systems (ITS). While the atention of the ITS is directed on the pedagogoical activities, and knowledge and learner modeling, the LMS are enpowered to accomplish different administraiton tasks: management of knowledge, course and student groups. The interoperabillity between different ITS and LMS requires the standardized data representation and technology which support those standard formats. The Semantic Web could be seen as an opportunity to solve the problems of interoperabillity.

7.1 Introduction

Using current Internet technology to support learning in the classroom is recently becoming much easier and much more feasible than it used to be. If a network of computers or workstations is available in a classroom (the same is on the global network), it is easy to install and use Apache, Tomcat, or another Web server. It can easily distribute HTML pages generated statically or dynamically by an educational application. Client computers/workstations should only have an Internet browser. Hardware and software requirements for the client machines are minimal.

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 [6]. 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 [5], the ITSs have high-level modularity. ITSs provide the 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.

G. Šimic et al.: *Intelligent Virtual Teaching*, StudFuzz **178**, 171–202 (2005)
www.springerlink.com

The learning management systems (LMSs) are much more successful in Web-enhanced education which is related to the number of users. LMSs are integrated systems that support a number of teachers' and students' needs. Teachers can use a LMS to develop Web-based course materials and tests, to communicate with students and to monitor their progress. Students can use it for learning and collaboration. Although the adaptive education systems perform every function much better than an LMS, today there is a complete dominance of LMSs over adaptive educational systems.

LMSs provide a teacher to compose their courses from newly created and also 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 with the number educational structures at the same time.

Intelligent LMSs (ILMSs) are the bridge between the modern approach to Web-based education (based on learning management systems) and powerful but underused intelligent tutoring and adaptive hypermedia technologies [6]. The ITS reusing of an supported domain in great deal of courses can be realized by the well-described knowledge. This knowledge has to be expressed in a precise, machine-interpretable form which enables the interoperable application components to process LO data, as well as on the syntactic and semantic level [9]. The Semantic Web, a recent Web community effort [2], is a promising technology for improving semantic interoperability of LOs [35]. The main parts of the Semantic Web are domain ontologies. Those ontologies should provide a formal description for a shared domain conceptualization [19]. As the new Web generation [6], the Semantic Web has better conditions for composing and reusing learning materials. The Semantic Web could be seen as an opportunity to enhance the metadata associated to learning materials, expanding the possibilities of current e-Learning specifications and standards [14].

We are going to try in this chapter to explain the main characteristics of the ILMSs, and to show our approach to create an ILMS called Multitutor, as a Semantic Web enabled system. In the next section we are going to give an overview of ILMSs and identify their shortcomings regarding interoperability. Section three explains the Multitutor architecture. Section four shows the Multitutor implementation in detail. In section five we show three courses developed in Multitutor: Code Tutor for teaching radio-communications, Design Pattern Tutor and a Petri net teaching system. Section six discusses how could we benefit of using Semantic Web technologies in development of e-learning systems.

7.2 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 [29]. 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 both structures - ITSs and LMSs. As with ITSs, in the ILMS there are modeling and representing relevant aspects of knowledge. This means that it contains the knowledge about a student, the domain, the pedagogy and the communication.

The general concepts that support the above knowledge aspects are implemented as components of ITS architecture. There are five basic ITS modules (Figure 1): *student model, domain knowledge, pedagogical module, expert model and communication model* [1]. The communication model is an interface for a student - system communication. This module provides the possibility that more users can be in the session with the system at the same time. Also the communication model dispatches appropriate learning contents to individual users.

The pedagogical module is a tutoring part of an ITS. Different learning strategies and teaching tactics would be implemented in this module. The pedagogical module is responsible for the decisions about every individual student. This module profiles the student and determinates the student model stereotype. During the student sessions, the pedagogical module measures the students' skills and knowledge, and updates the student model. The system changes its behavior according to the students' skills and knowledge.

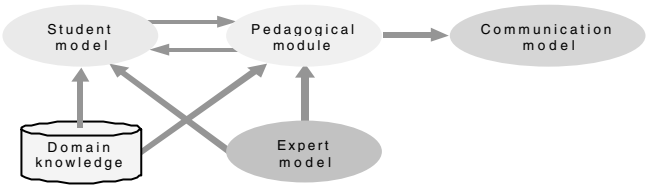

Fig. 1. General concepts of the ITS architecture

ITSs have high intelligent performances. The level of intelligence of an ITS is proportional to the measurement in which the student model describes the real student. The system delivers the educational content to the student 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 metaknowledge 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's also annotated at one kind of script files and which are readable only for specified ITSs. This KB can not be used by other systems. Only ITSs, that support appropriate script format, can reuse this knowledge. Another problem is that knowledge is not 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 softwares, finalized, and enable the 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 the both - the self-paced and the instructor-led learning processes [23]. All these activities are represented to the end user (or organization) as a group of Web services. The architecture of LMS is more complex than in the ITS case (Figure 2). As Web services these systems are more transparent and they have more security mechanisms. LMSs are poorly Web oriented systems that are hosted on both Web and application servers.

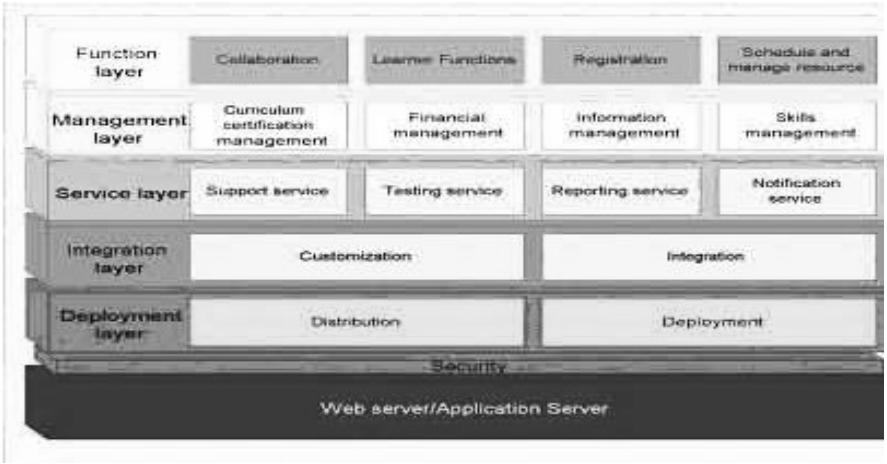

Fig. 2. The LMS Architecture

The last two LMS layers are designed for composing, customizing and communicating services with end-users. This means that LMSs are high-distributed systems over the Web. One course presents an integrated structure of many learn-

ing 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 (Figure 3). The ILMS general architecture consists of three basic parts: *administration tools*, *teacher tools,* and *student tools*.

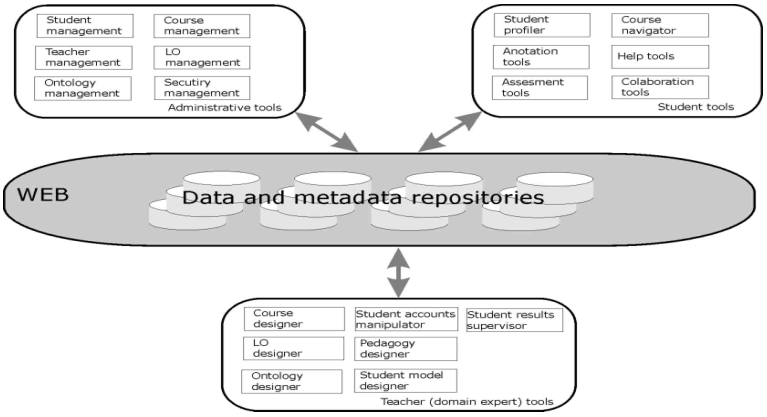

Fig. 3. The ILMS Architecture

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 (about educational domains, pedagogy and students). The teacher package provides the monitoring of student results that could be used by teachers to track the student sessions.

The student tools generally help students to master the knowledge. The system enables a student to declare his interests, favorites, predispositions 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 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:

the integration process and 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 couldn't produce easily the intelligent educational systems, because of the lack of ontological support [10]. They also have abssence of the learner modeling, reasoning and adaptible behavior. Although they provide presentation and management of leaming material and scenarios, as well as database management and administration.

7.3 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* [34]. 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.

7.3.1 Motivation

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 (http://www.ghg.net/clips/CLIPS.HTML), Java-based ES shell *Jess* was used to interpret these files (http://herzberg.ca.sandia.gov/jess/), *Java™ Servlet* technology (http://www.sun.com/products/servlet/) to implement the system's interactions with the students, *Apache HTTP* server (http://www.apache.org/) to store static HTML pages, *Apache JServ* (http://java.apache.org/dist/) to interpret the servlets, and *XML* technology [15] 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.

In the next development phase, we were focused on the authoring tool design. One of the main ITS disadvantages is their narrow domain specialization of the system. For example, the Cognitive tutor [33], which is recommended by NCTM[1], is focused on mathematics (algebra 1 and 2, geometry). The ELM ART [4] system is designed to teach students in LISP programming. The SQL Tutor [28] provides students the possibility to learn SQL. The ILESA [26] system is specialised to teach the solving of linear programming problems. The VALIENT [20] system provides the learning of the data base design.

[1] NCTM – U.S. *National Councel of Teaching Mathematic*

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.

The domain independence is possible only if it is supported with appropriate authoring tools. Above, enumerated ITSs don't provide the teachers the possibility to modify the learning content. On the other side there are a number of authoring tools for ITSs. These are divided in two general groups: *teachers oriented* and *domain oriented*. The first kind of tools provides an easy way for teachers to create courses. The latter type offers a rich interface for describing and structuring the domain. A good example of teacher oriented authoring tool is REEDEM [27]. This tool represents an author-friendly environment, in which, a teacher can define a student model, learning strategies and to describe the course materials. The domain-oriented tools have many possibilities for semantic description of the knowledge. EON [30] is an example of the authoring tool that provides semantic network design of domain concepts and facts (Figure 4). The graphical representation of the resulting domain ontology is very useful, but it demands the teacher to know how to use graphical designer environment.

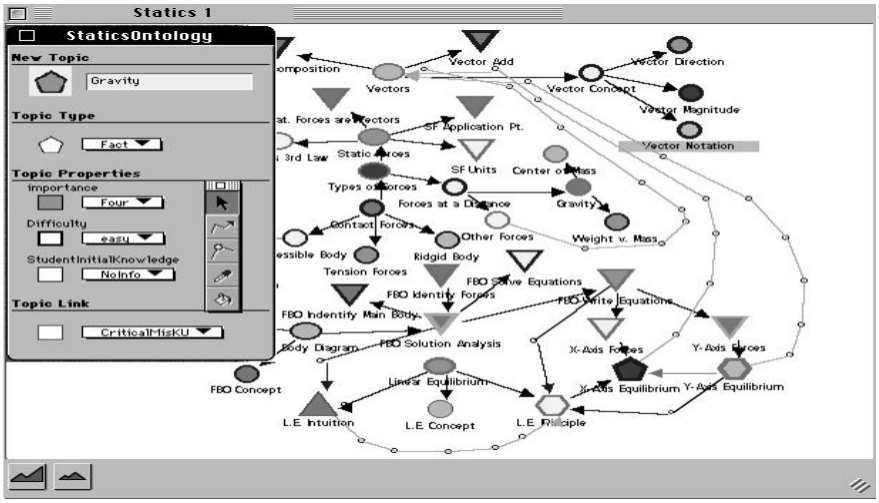

Fig. 4. Design of semantic network in the EON authoring tool

Unfortunately, the created courses can be used only in the specialized framework which is distributed with this tool. The interoperability with the outside applications is nearly impossible because this framework usually does not have interfaces for Dynamic data exchange (DDE) and does not support the standardized data structure formats. The data formats are specialized and strongly coupled with the system components. These frameworks have their own graphical interface and can be hosted only over the LAN environments.

7.3.2 Multitutor Architecture

We tried to design an authoring tool that is a part of the Mulitutor system. The component called *Course Designer* (Figure 7) 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 (Figure 5). The user is on the client side and the student can 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.

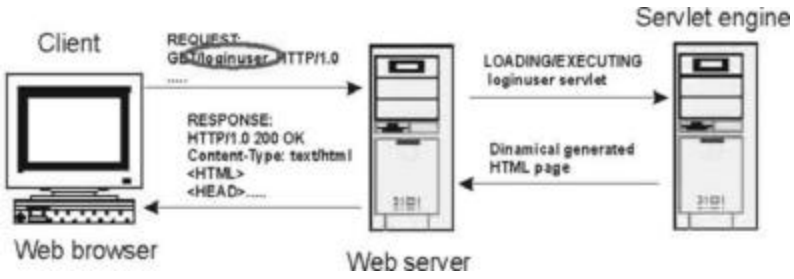

Fig. 5. Client-server paradigm of the Multitutor

The system architecture can be divided in more than three layers (Figure 6). The client's browser can open the HTML pages on any Web location in one session with the system. The Web server proceedes an HTTP request to the application server. Data that are used by the Web applications may be hosted on the different Web repositories. Different applications can use the same data on the Web. One can see in Figure 6 that all the three applications use the domain and pedagogical knowledge from the same network places. This utillity is provided by semantical linked data.

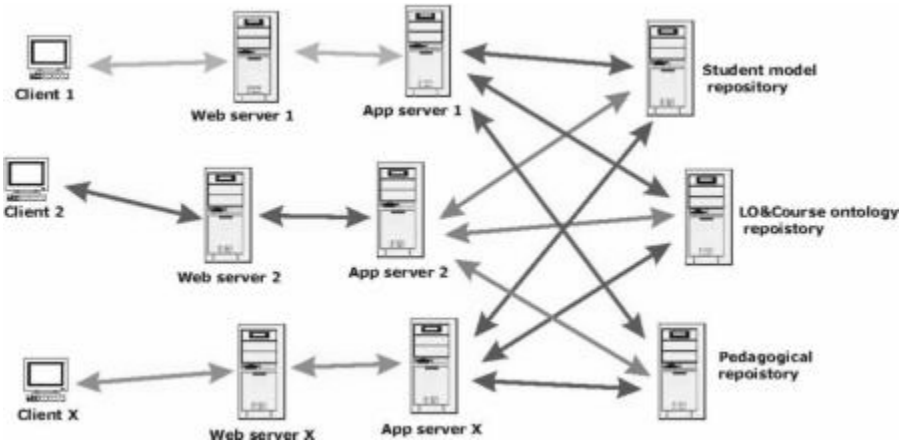

Fig. 6. Part of the real network

Web applications contain the reference maps of Web resources and standardized parsers that can recognize the structure and semantic of these resources. The Web application processes (puts) the user data in correlation with ontologies data from the repositories. This processing can be rule-based (i.e. pattern matching), or based on non-linear reasoning (i.e. fuzzy logic), or other. For example, the applications try to find an adequate stereotype of the current user. Then the application accesses the student model repository and compares the student profile with the accessible models. Based on the founded model, application consults the pedagogical repository to determine appropriate teaching strategy. Then the application can compose learning material for a specified student. The LO and course repository is used for this purpose.

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. He/she is responsible for:

- Adding and removing the teachers in the tutor system registry – only the registered teachers can use the system;
- Checking the data integrity – controls the teachers and students' registers and log files;
- Viewing the system log files and preserving the system from the malicious user operations – the student results are read-only data and only the course teacher can access and view their results;
- Maintaining the web server and the servlet engine – supervising the Web server repositories and the servlet properties; editing the configuration files and the zone files;
- Maintaining backup of the system files (teachers, students' results, ontology and knowledge base files) – the temporary updating of the copies of the tutor system files.

The teacher tasks are well known. The 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.

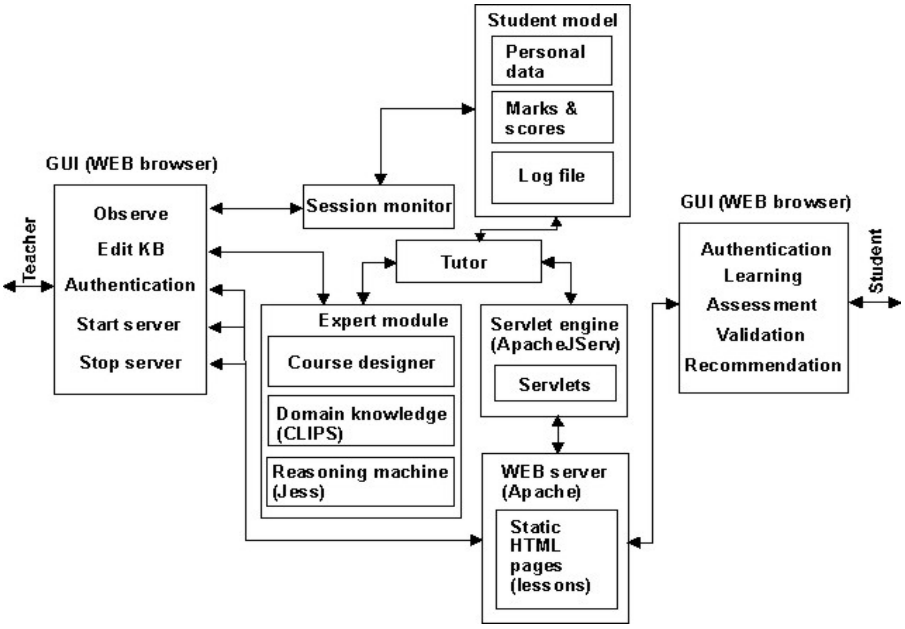

Fig. 7. The Multitutor architecture

The students are organized into a 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 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's 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. A servlet functional call is broadly propagated over the system. These results with many actions. This way the Multitutor is able to dynamically generate the learning content.

The architecture of the system is designed to support low coupling of particular components. Therefore the system components can be highly distributed over the Web. The administration tasks are performed on the teacher side of application. The administrator interface is not shown because we want to avoid confusion on the diagram. Note that the components are implemented in different technologies.

7.4 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.

7.4.1 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 8) abstracts these relations and it can be translated in the basic system ontology [8].

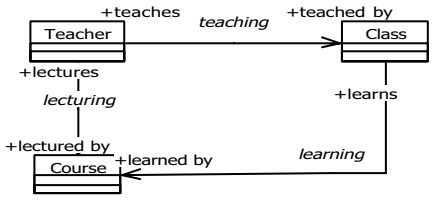

Fig. 8. The general concepts of the learning process

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. The students results is read only.

This model can be converted in an ontology schema that is readable for another part of the application logic. We used XML Schema [15] to create the ontology vocabulary. An excerpt of this XML Schema definition is shown in Figure 9. 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.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!-- ...-->
    <xsd:element name="ontotutor">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Teacher">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element ref="Name"/>
                            <xsd:element ref="Teaches" maxOccurs="unbounded"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <!-- ... -->
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Teaches">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Course"/>
                <xsd:element ref="Class" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Course">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Name"/>
                <xsd:element ref="Reference"/>
                <!-- ... -->
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Class">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Name"/>
                <xsd:element ref="Student" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Fig. 9. Schema of initial application data

Based on the defined metadata in the schema, the Multitutor reads the initial application data from the *ini* file. In the code fragment in Figure 10 the teacher is "Peter Fox" who teaches the course called "Physics" to two classes – "SIG22" and "EW43". The shown example demonstrates one of the possible ways to represent the relation between the ontological concepts. Many formal and standardized

markup languages suffer of the impossibility to represent the semantic of the system data. The same way all other relations from the conceptual model (i.e. from Figure 8) are described by metadata in the schema file. Note that the *Course* concept has an element called *Reference*. This means the data of the specified course are located in one Web destination. Furthermore, the course data are well structured and described by schema (metadata). The *Class* concept semantically represents the student group. The *Student* concept is defined in the student model ontology that is located on a separate repository.

```
<?xml version="1.0"?>
<Ontotutor>
    <Teacher>
        <Name>Peter Fox</Name>
        <Teaches>
            <Course>
                <Name>Physics</Name>
                <Reference>http://servername/mtutor/ontologies/phisics.xml</Reference>
            </Course>
            <Class>SIG22</Class>
            <Class>SIG43</Class>
            <!-- ... -->
        </Teaches>
    </Teacher>
    <!-- ... -->
</Ontotutor>
```

Fig. 10. The initial system data

7.4.2 The Basic Concepts of the Course Ontology

The course has ontology that is referenced in the application *ini* file. 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* (Figure 11).

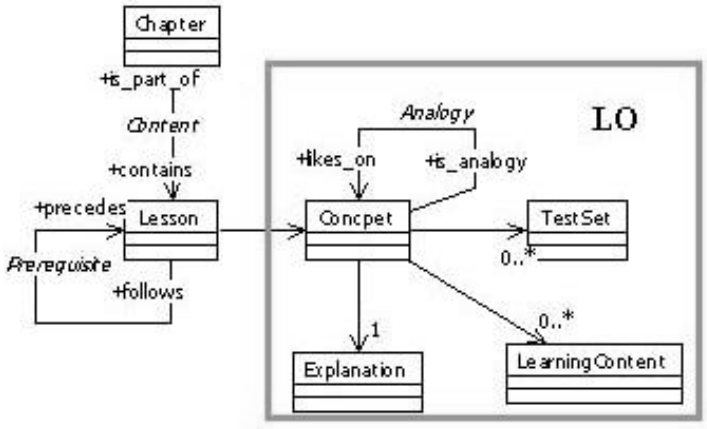

Fig. 11. The main concepts of the course ontology

By this, 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 system uses to assess the students' knowledge about one concept. The test set has attributes as ID, level and the test type. The Multitut?r 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.

Figure 12 shows a part of the course schema file that is derived from class diagram shown in Figure 11. In the schema we use *Analogies* elements to represent the concept's self-relation.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!-- ...-->
    <xsd:element name="ontocourse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Concept"/>
                <xsd:element name="Analogies">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element ref=" Concept " minOccurs="1" maxOccurs="1"/>
                            <!-- ...-->
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="LearnigContent">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element ref="Name" minOccurs="1" maxOccurs="1"/>
                            <!--... -->
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Concept">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Name" minOccurs="1" maxOccurs="1"/>
                <!-- ...-->
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Fig. 12. The fragment of the course schema file

The entities that are self-related can play different roles. In the next example (Figure 13), there are two lessons in the course *Physics* file (the chapters of the course are not shown). Before the student learns the lesson about the sound waves, he has to learn the lesson about the wave motions.

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. 13. The fragment of the course data

7.4.3 The Student Model

The student model has a separate ontology (Figure 14). 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).

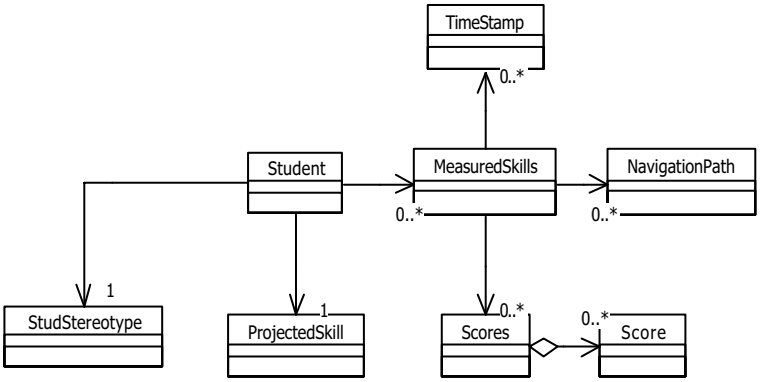

Fig. 14. The student ontology

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 *ProjectedSkill* concept of the model (Figure 15). 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 correlation 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.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="ontostudent">
        <!-- ... -->
        <xsd:complexType name="Skill" abstract="true">
            <!-- ... -->
        </xsd:complexType>
        <xsd:complexType name="ProjectedSkill">
            <!-- ... -->
            <xsd:extension base="Skill">
        </xsd:complexType>
        <xsd:complexType name="MeasuredSkill">
            <!-- ... -->
            <xsd:extension base="Skill">
        </xsd:complexType>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Student">
                    <!-- ... -->
                </xsd:element>
                <xsd:element name="PreferedLearningStyle" type=" ProjectedSkill ">
                    <!-- ... -->
                </xsd:element>
                <xsd:element name="Test" type=" MeasuredSkill ">
                    <!-- ... -->
                </xsd:element>
                <!-- ... -->
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <!-- ... -->
</xsd:schema>
```

Fig. 15. The student model metadata

7.4.4 The Learning Content

The learning materials are dynamically composed (Figure 16). The Multitutor correlates the current and historical student data and it makes decision about the learning content. The basic explanation of the domain (lesson) concept is in the text form (The *Explanation* entity in Figure 11).

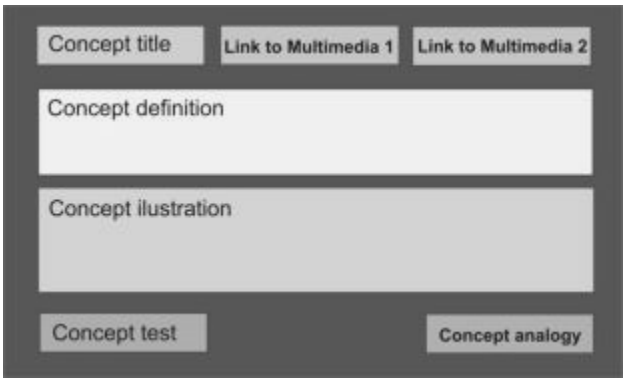

Fig. 16. The Web page structure

The text is yet the most precisely way to define the concept and avoid the ambiguity. In the Multitutor, the other contents (figures, sounds, and video) are used to support the better understanding and the faster learning of the concepts. The learning material is represented by Web page that has the table structure.

7.5 Mutitutor Applications

This chapter presents the description of the three courses which are designed by using the Multitutor.

7.5.1 Code Tutor

The Multitutor is used to compose three different courses. The first of them is Code tutor that is designed for learning the protocols and codes that are used in the radio-communications. The student starts the session by log in the system (Figure 17). He selects the class and types his name and password.

Fig. 17. The student login

The student adapts the page lookup (Figure 18). He can change the background texture, the font style and size. The first version of Multitutor has the yellow font and the blue texture in the background. We are founded by the inquiries, the half of the students do not prefer such interface lookup. The previous versions show the test (possible answers) by the list boxes. The problem is when the text is greater than the list box width. In the last version the student can select the radio-button style or the list boxes (Figure 19). The system remembers the students' selections from the previous sessions. When the student log in, the system presets the interface based on these data. This property reduces the communication between the students and system.

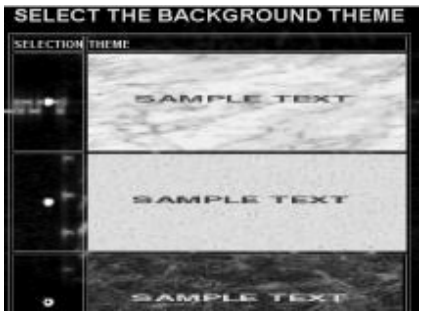

Fig. 18. The background and text setup

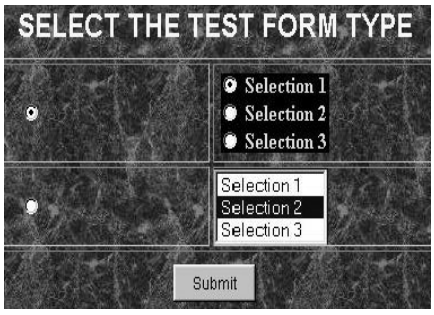

Fig. 19. The test form styles

The Multitutor offers the courses that are designed for the students' class. In the next step, the student selects the course. In the one session the student can learn one course. If the student wants to learn another course, he has to finish the session and starts the new one.

After the student selects the course, he has to choose the chapter. In the one session the student can learn more chapters of the one course. He can also learn the same chapter more times.

The next stage is the learning (Figure 20). The chapter is divided on to the lessons. The sequence of the lessons is defined from the student model data. The student has to learn the lessons that are not negotiated. Based on the current selections and the data from the previous sessions, the Multitutor composes and delivers the learning content to the student.

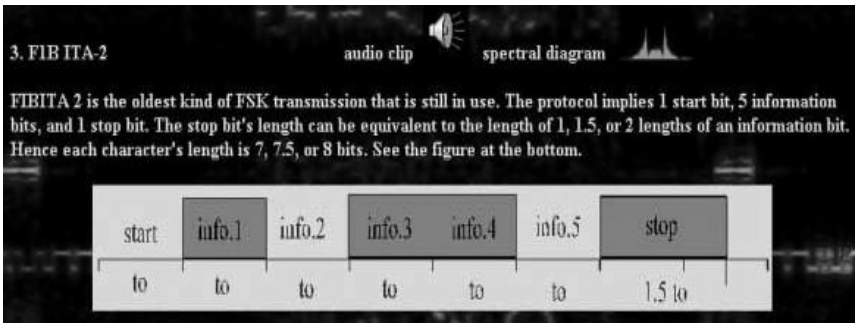

Fig. 20. The learning content

The learning material consists of five basic parts: The title of the concept, the basic explanation, the main illustration that describes the concept, the other multimedia content (e.g. audio clips and figures), and the links. There are two links: the analogy and the test link.

The student can learn the same lesson arbitrary times. The next phase is the testing (Figure 21). The Multitutor delivers the test questions to the user. The test content is related to the one-chapter lessons. If the student learns the same chapter more than one, every time he gets the new set of questions. There are tree types of the test sets in the system: the multiple selections, the single selection and the answers scaled by marks. In the case of the multiple/single selection types the system calculates the percentage of efficiency. In the third case, the system calculates the average of the particular marks. In the next stage, the system represents the results to the student. If the student is failed he has to repeat the bad marked lessons again. If the student passes the test, the system recommends him what to learn. In this case the student can accept this or select another chapter.

Fig. 21. The test page

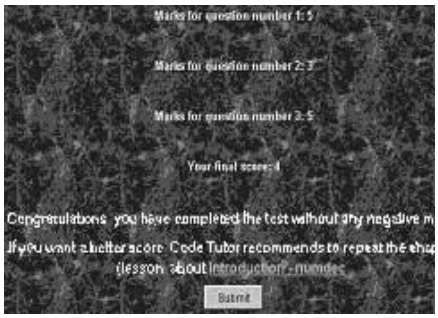

Fig. 22. The results and recommendation

Before the system represents the results to the student (Figure 22), it saves these data in the XML-formatted result file (Figure 23). In the previous versions of Multitutor, the results are grouped by the course. To support the better student model, the new version creates the results file for each student. The student can

see the test details by clicked on the question result. Than the system opens the new page that contains the question, student answer, correct answer and mark.

```
<?xml version="1.0"?>
<results>
    <student>
        <name>Aldo_Boca</name>
        <class>EW52</class>
        <testdate>Mon_Mar_22_12:28:01_CET_2004</testdate>
        <tcourse>
            <name>code-3</name>
            <tchapter>
                <name>decoding</name>
                <tlesson>
                    <name>artrac</name>
                    <setnum>1</setnum>
                    <level>1</level>
                    <tquestion>What time (ms) of artrac CRC is ?</tquestion>
                    <tanswer>1016</tanswer>
                    <tbestanswer>706</tbestanswer>
                    <tmark>2</tmark>
                </tlesson>
            </tchapter>
        </tcourse>
        <!-- ... -->
    </student>
    <!-- ... -->
</results>
```

Fig. 23. The student results

The Multitutor provides XSLT (Extensible Style Language Transformation) to convert results from XML format to HTML format and represents them through the Web page. A student can see only his/her results, but the teacher can see the results of the whole class (classes). These data are shown in the read only format both the teacher and the student.

7.5.2 Design Pattern Tutor

Design Pattern Tutor is a Multitutor course for learning Design Patterns. The course gradually introduces student with the concept of design patterns and describes most frequently used classes of patterns.

The issue imposed in course realization for learning the Design Patterns, is the way of organization of learning process. In the course of realization of this problem, the "Design Patterns" book [16] should support this course. The course for learning Design Patterns is divided into three main sections: creation patterns, structural patterns and behavioral patterns.

The ITS system for learning the Design Patterns, described in this section, makes tutorial model of learning the Design Patterns possible, as well as independent study of patterns for more advanced students. The system provides an intelligent representation of educational material. This material is adjusted to student performance, such as degree of backward knowledge, desirable detail level

and assessments of the system. Also this system assess the level of student's acquaintance with this domain.

During interaction of a user with the system, the Tutor monitors the performance of the student. It updates the Student model and checks if the plan is still appropriate. The Tutor uses tests and exercises to get feedback from the student in order to infer changes in the Student model. Assessments obtained through such interactions are compared with the other parameters and assessments in order to get the final judgment of a student.

If a student is having a session with Tutor for the first time, system gives him a set of exercises and tests before the tutoring process starts. In this way, the student model estimates the new student's background knowledge and determines initial values. After the initial assessment of the student, knowledge system runs in teaching mode. At the beginning of this mode, system develops an initial plan of teaching actions. The plan is based on the contents of the lesson being presented, the relevant pedagogical rules and the student model. The plan represents a detailed outline of the lesson presentation.

At the end of presented topic, the Tutor runs at examination mode. It generates exercises and tests for the student and assesses his knowledge. During the session, system observes and adapts the student progress with the generated course. If the student answers the test items correctly, he progresses along the course and no changes to the course are necessary. However, if the student fails to answer the test items correctly, the system must modify student performances. If the student's performance does not meet expectations, the course is dynamically re-planned. Through dynamic regeneration, each student is able to get a highly personalized course for his needs. [7].

The system must memorize each activity of a user and the system, as well as all assessments of a student, updating the student model. These data may be used to prepare instructional plan, as well as to give advice and recommendations for further work [32].

During the session, system observes and adapts the student progress with the generated course. Domain model of Design Pattern Tutor is made up of concepts, which correspond to one pattern. Each concept is divided in units – the elementary pieces of domain knowledge. There are a fixed number of units in particular concept, but the size of unit is not fixed. The system uses unit variants technique, which consists of keeping two or more alternative pages with adapted content, e.g. one for each knowledge level: beginner, intermediate and expert. Each unit has an arbitrary number of fragments – a chunk of information that should be presented to the user. The user model and the concept relationships of the domain model provide the information that allows the system to determine which chunk of information should be presented to the user. The chunk of information may also consists of fragment variants, i.e. fragments related by an "or" relationship.

The system provides students with two kinds of navigation through the course material (Figure 24.):

- Direct guidance – The student sees only one options to continue with the browsing activity i.e. just one button to navigate to the "next" page is displayed. The destination of the "best" link is determined by the system.

- Link removing – advanced students could choose which topics to learn by selecting appropriate link from content menu, but links that the system considers inappropriate are removed, i.e. they are not longer available. Anchors of these links are replaced by text.

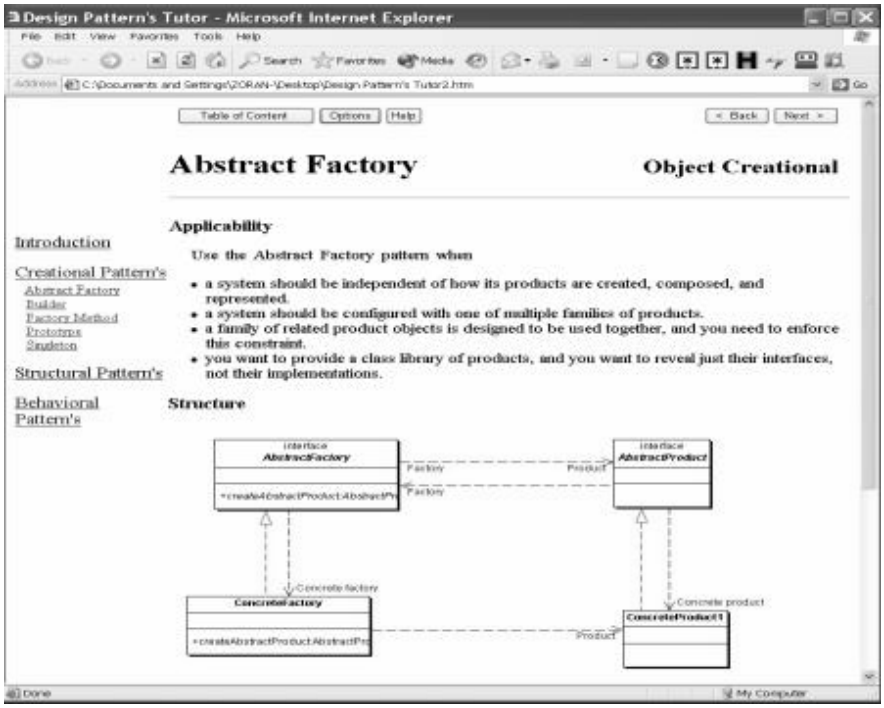

Fig. 24. A Web page shows a screenshot of Design Pattern Tutor - course created by Multitutor

The student has the option of letting the Tutor choose the next topic or choosing it himself. In both cases, the student must achieve sufficiently ready score for the topic. The topics are represented in a dependency graph, with links representing the relationship between topics, which include prerequisite and related topic. A student is ready to learn a topic only if he has performed sufficiently well on its prerequisites.

At the end of each topic student has to complete test. If the student fails to give correct answers, the Tutor must provide an alternative learning path to the student, such as a hint. If the learner tries to move on to a new topic before the Tutor founds that the student has explored the current topic sufficiently. The Tutor will generate a warning, suggesting better exploration of the current topic. These wamings also remind the student of the availability of hints. The student can choose either to follow the advice or to move on.

7.5.3 Semantic Web Empowered Learning: A support for Petri Nets

In order to illustrate how Multitutor can be used for Semantic Web learning applications we show a simple Petri net educational system. The Petri net system is intended to be used in a number of computer science courses that use Petri nets (e.g. distributed computer systems, computer architecture, operating systems, etc.). However, if we want to use Petri net model in Multitutor we should prepare suitable equipment. In our case we have the following elements:

- *The Petri net ontology* – This ontology we assume as a domain ontology of Petri net educational context. This ontology is developed using Protégé tool and UML. The ontology describes Petri net conceptualization using RDFS and OWL languages. Additionally, the Petri net ontology is in accordance with the Petri Net Markup Language (PNML) – an ongoing Petri net community effort for the standard XML-based sharing format. The Petri net ontology has a common part that contains concepts common for all Petri net dialects. Afterward, this common part should be specialized for concrete Petri net dialect [17].
- *The P3 tool* – A Petri net tool we have developed for teaching Petri nets [18]. The P3 tool supports the two Petri net dialects: P/T nets and Upgraded Petri nets. Also, it has the following Petri net analysis tools: reachability tree, equation matrix, firing graph, firing tree. The P3 tool has advanced model-sharing features based on PNML. Furthermore, it has a collection of the XSLTs that transform PNML to other Petri tool specific formats (i.e. DaNAMiCS, Renew, Petri Net Kernel, and PIPE). Also, P3 implements conversion of the PNML Petri net model description to Scalable Vector Graphics (SVG). Since this format can be viewed in standard Web browsers (e.g. Internet Explorer), it is suitable for creating Web-based Petri net teaching materials. Learning objects, created in this way, have their underlying semantics described in RDF form, and we are able to transform them (e.g. using XSLT) into PNML. That way, the learning object can be analyzed with standard Petri net tools.
- *Petri net Web Service* – A Web service that uses a PNML Petri net model as input, performs one simulation step and generates result, again, in PNML format [21].

The resulting Semantic Web infrastructure for Petri nets is shown below (Figure 25). This infrastructure summarizes all major features of the Petri net ontology, P3, and the Petri net Web Service. The central part of this infrastructure is PNML.

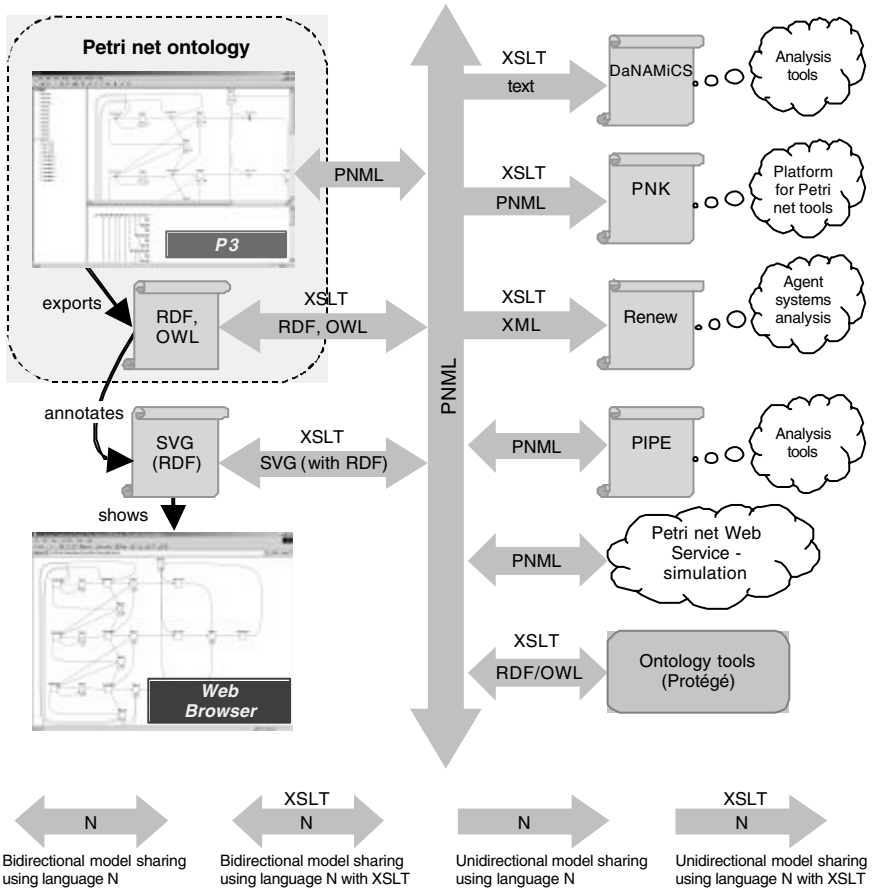

Fig. 25. Petri net infrastructure for the Semantic Web (that uses "PNML-based bus" for model sharing): the Petri net ontology, current Petri net tools, P3 tool, Web-based applications, Petri net Web Service, and ontology tools for validation of Petri net documents using the ontology

This Petri net infrastructure for Semantic Web can have a number of applications in practice. Here, we show how it can be used within Multitutor for developing Petri net courses. A teacher creates Petri net models in RDF-annotated SVG format using the P3 tool. Then, the teacher uses these models in Multitutor where he/she creates courses following the procedure that we have already explained. In fact, the teacher use Petri net models as figures, but these figures have an ontologically annotated content. After the teacher have finished a course students can use it for studying.

In order to empower Multitutor created courses with ability to perform interactive simulation of Petri net models, implementation of the logic of Petri net execution is needed. This can be achieved using Web service for Petri nets simulation

developed. From the created course we should forward a Petri net model to the Web service. This model is converted from RDF annotated SVG format into PNML format using an XSLT. Once the simulation is finished, another XSLT is used to transform the result from PNML to RDF annotated SVG format. Both XSLTs are part of proposed infrastructure. The suggested approach to educational systems development using proposed Petri net infrastructure for the Semantic Web is depicted below (Figure 26).

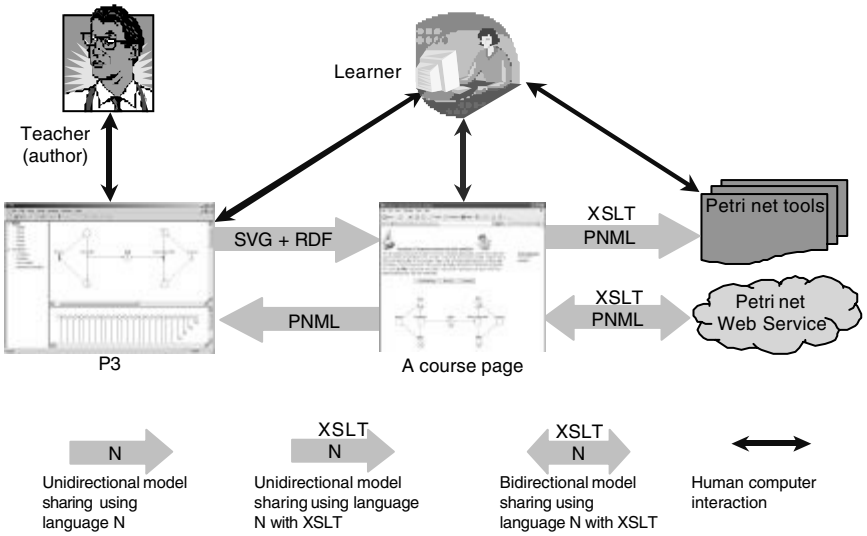

Fig. 26. An approach to using Petri net infrastructure for the Semantic Web in Web-based educational systems

Note that we can not implement the calls of Web service procedures from Multitutor. Actually, we should extend generated Multitutor's courses manually by adding a few method calls responsible for using Web service's methods. In the future Multitutor versions we are planning to implement suitable tools that will be able to access and use Web service.

The Web page from the lesson that helps students to understand and learn the well-known producer/consumer synchronization problem is shown below (Figure 27). This problem is a common part of many different courses in computer science. User begins his/her interaction with the Web page by pressing button **Initial Marking** in order to define initial marking of the Petri net model. Automatically, Petri net graph conforms to the specified data reflecting changes of the model. A press on the **Simulate** button is a sign for system to start simulation of the model. Simulation is performed in collaboration with the Web service according to the previously explained scenario. Simulation results are shown on the Petri net graph. User can save a Petri net he/she is working with in PNML format by choosing button **Save as**. Having studied this Web page the student is going to the next one according to the course plan.

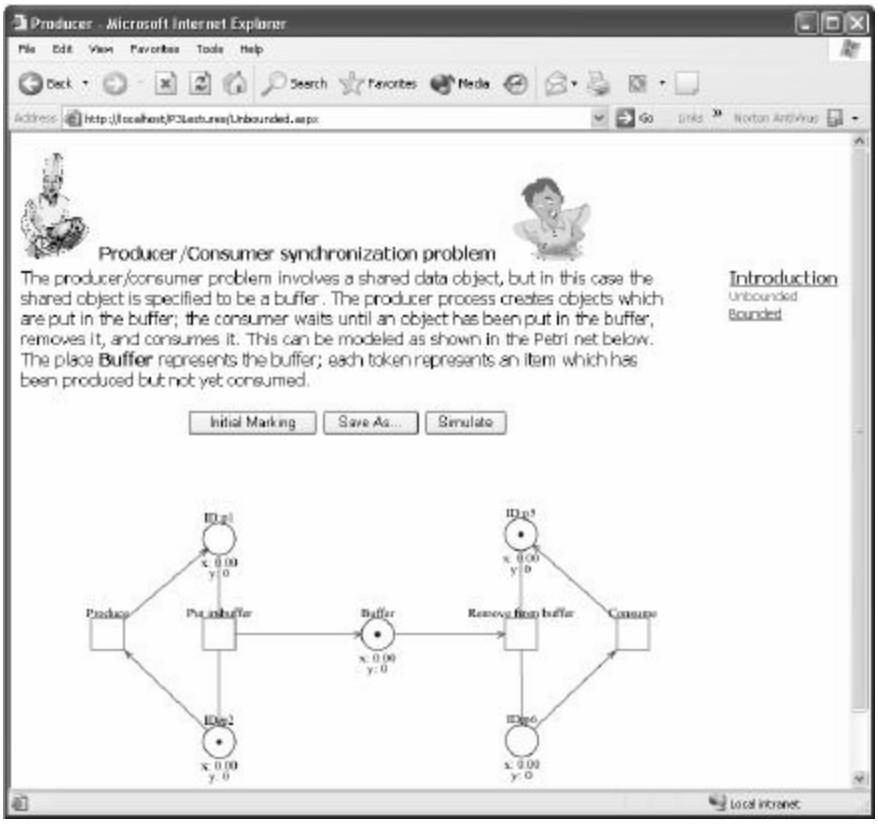

Fig. 27. A Web page shows how RDF-annotated SVG documents can be used in courses created by Multitutor

7.6 Intelligent Learning Management Systems and the Semantic Web: 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 (Figure 9), 2. Courses, 3. Student models. However, the XML Schema mechanism itself has several weaknesses regarding the ontology description [24], 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 [31]. Brase and Nejdl showed how ontologies could be exploited to enhance LO metadata in Edutella [3]. 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 [12]. 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 [13]. 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) [22]. 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 [25]. 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 [36]. 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.

7.7 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 – the student side and the teacher side. 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 course and student ontologies rather that current XML Schema definitions). 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 [29] in the last years. But, this makes the problems for developers like which solution is the most appropriate. Accordingly, the main challenge for the e-learning community is to adopt standard Semantic Web ontologies [11] that will be guidelines for the developers of LMSs/ILMSs.

References

[1] Beck J., Stern M. and Haugsjaa E.,(1996): Applications of AI in Education, ACM Crossroads, Vol. 3, No. 1, pp. 11-15

[2] Berners-Lee T., Hendler J., Lassila O.,(2001): The Semantic Web, Scientific American, Vol. 284, No. 5, pp 34-43.

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

[4] Brusilovsky P., Schwartz E. and Weber G.,(1996): ELM-ART: An Intelligent Tutoring System on the World Wide Web, In Proceedings of the 3rd International Conference on Intelligent Tutoring Systems, Montreal, Canada, pp. 261-269.

[5] Brusilovsky P.,(2001): Adaptive Hypermedia, User Modeling and User-Adapted Interaction, Vol. 11, No.1-2, pp. 87-110.

[6] Brusilovsky P.,(2003): A Distributed Architecture for Adaptive and Intelligent Learning Management Systems, In Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems, Sydney, pp. 5-13.

[7] Brusilovsky P., Vassileva J.,(2003): Course sequencing techniques for large-scale web-based education, Int. J. Cont Engineering Education and Lifelong Learning, Vol. 13, Nos. 1/2, pp. 75-94.

[8] Calvo R. A.,(2003): User Scenarios for the design and implementation of iLMS, In Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems, Sydney, pp. 14-22.

[9] Devedžic V.,(2003): "Web Intelligence and AIED," In Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems, Sydney, pp. 23-33.

[10] Devedžic V.,(2003): Key Issues in Next-Generation Web-Based Education, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 33, No. 3, pp. 339-349.

[11] Devedžic V.,(2003): Think ahead: evaluation and standardization issues for e-learning applications, International Journal of Continuing Engineering Education and Lifelong Learning, Vol. 13, No. 5/6, pp. 556-566.
[12] Dichev Ch., Dicheva D. D. and Aroyo L.,(2004): Topic Maps for E-Learning, International Journal on Advanced technologies for Learning, ACTA Press, Vol. 1, No. 1, pp. 1-7.
[13] Dolog P., Henze N., Nejdl W. and Sintek M.,(2004): Personalization in Distributed eLearning Environments, In Proceedings of the 13th International World Wide Web Conference, NY, USA.
[14] Duval E., Hodgins W., Sutton S.A. and Weibel S.,(2002): Metadata principles and practicalities, D-Lib Magazine, Vol. 8, No. 4.
[15] Fallside D. C., ed. (2001): XML Schema Part 0: Primer, W3C Recommendation [Online]. Available: http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/
[16] Gamma E., Helm R., Johnson R. and Vlissides J., (1995): Patterns –Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Company, USA.
[17] Gaševic D. and Devedžic V.,(2004): Reusing Petri Nets Through the Semantic Web, In Proceedings of the 1st European Semantic Web Symposium, Heraklion, Greece, pp. 342-351.
[18] Gaševic D. and Devedžic V.,(2004): Teaching Petri Nets Using P3, Educational Technology & Society (Journal of IEEE Technical Committee on Learning Technologies), (forthcoming).
[19] Gruber T. R.,(1993): A translation approach to portable ontology specifications, Knowledge Acquisition, Vol. 5, No. 2, pp. 199-220.
[20] Hall L. and Gordon A.(1998): Synergy on the Net: Integrating the Web and Intelligent Learning Environments, In Proceedings of The Workshop on Web-Based ITS, San Antonio, TX, pp. 608.
[21] Havram M., Gaševic D. and Damjanovic V.,(2003): A Component-based Approach to Petri Net Web Service Realization with Usage Case Study, In Proceedings of the 10th Workshop Algorithms and Tools for Petri nets, Eichstätt, Germany, pp. 121-130.
[22] Heckmann D., Krueger A.(2003): A User Modeling Markup Language (UserML) for Ubiquitous Computing, In Proceedings of the 9th User Modeling Conference, Johnstown, Pennsylvania, USA, pp. 393-397.
[23] iCMG Learning Management System (LMS) Architecture (May 25, 2004) [Online]. Available: http://www.icmgworld.com/corp/ces/ces.lms.asp
[24] Klein M.(2001): XML, RDF, and Relatives, IEEE Intelligent Systems, Vol. 16, No. 2, March/April, pp 26-28.
[25] Koper R.(2002): Educational Modeling Language: adding instructional design to existing specifications, Workshop "Standardisierung im eLearning", Frankfurt, Germany.
[26] López J. M., Millán E., Pérez-de-la-Cruz J.L. and Triguero F.(1998): Design and Implementation of a Web-based Tutoring Tool for Linear Programming Problems, In Proceedings of the Workshop on Web-Based ITS, San Antonio, TX, [Online] Available: http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/ilesa.ps, pp. 1-8.
[27] Major, N.(1995): REDEEM: Creating Reusable Intelligent Courseware, In Proceedings of The International Conference on Artificial Intelligence in Education, Charlottesville, VA, USA, 1995, pp. 75-82.
[28] Mitrovic A. and Hausler K.(2000): Porting SQL-Tutor to the Web, In Proceedings of the International Workshop on Adaptive and Intelligent Web-based Educational Systems, Montreal, Canada, pp. 50-60.
[29] Mizoguchi R. and Bourdeau J.(2000): Using Ontological Engineering to Overcome Common AI-ED Problems, International Journal of Artificial Intelligence in Education, Vol. 11, pp. 1-12.

[30] Murray T.(1996): Having It All, Maybe: Design Tradeoffs in ITS Authoring Tools, In Proceedings of the 3rd International Conference on Intelligent Tutoring Systems, Montreal, Canada, pp. 93-101.

[31] Nilsson M., Palmér M. and Naeve A.,(2003): 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, pp.78-85.

[32] Prentzas J., Hatzilygeroudis I., Garofalakis J.(2002): A Web-Based Intelligent Tutoring System Using Hybrid Rules as Its Representational Basis, In Proceedings of the 6thIntern. Conference, ITS France and Spain, pp. 119-128.

[33] Ritter S.(1997): PAT Online: A Model-Tracing Tutor on the World-Wide Web, In Proceedings of the Workshop-a Intelligent Educational Systems on the World Wide Web, Kobe, Japan, pp. 11-17.

[34] Šimic G., Devedžic V.(2003): Building an intelligent system using modern Internet technologies, Expert Systems with Applications, Vol. 25, No. 2, pp. 231–246.

[35] Stojanovic Lj., S. Staab S. and Studer R.(2001): eLearning in the Semantic Web, In Proceedings of the World Conference on the WWW and the Internet (WebNet 2001), Orlando, Florida, USA, pp. 325-334.

[36] Weitl F., Süß C., Kammerl R.and Freitag B.(2002): 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, pp. 1018-1025.