

ITS'04 WORKSHOP

Applications of Semantic Web Technologies for Web-based ITS

in conjunction with
ITS 2004: International Conference on Intelligent Tutoring Systems

August 30th, 2004

Maceió-Alagoas, Brazil

Preface

The Semantic Web offers new technologies to the developers of web-based applications aiming at providing more intelligent access and management of the Web information and semantically richer modelling of the applications and their users. An important target for web application developers nowadays is to provide means to unite, as much as possible, their efforts in creating information and knowledge components that are easily accessible and usable by third parties. Within the context of semantic web, there are several hot issues, which allow achieving this reusability, shareability and interoperability among web applications. Conceptualizations (formal taxonomies), ontologies, and the available web standards, such as XML, RDF, XTM, OWL, DAML-S, and RuleML, allow specification of components in a standard way. The notion of web services offers a way to make such components mobile and accessible within the wide sea of web information and applications.

The research on Web-based ITS traditionally combines research interests and efforts from various fields. Currently, the efforts in the field of Semantic Web and ontologies play an important role in the development of new ITS methods and types of courseware. Starting with the traditional ITS and going towards Web applications, the research on adaptive and intelligent courseware strengthens its positions within this context. Employing Semantic Web technologies seems to be a promising way to improve reasoning, adaptation and flexibility for in single and group users (e.g. instructors, courseware authors and learners). The goal of this workshop is to present the state-of-the-art of the application of Semantic Web technologies and standards for problems in current Web-based intelligent tutoring systems and adaptive courseware. We explore, among others, various architecture trends, reference models, user models and knowledge representation techniques inspired and empowered by the achievements in the field of semantic web. We approach the problems of Semantic Web applications from the perspective of ITS, starting with the traditional view on ITS and their movement towards the web and the implications of adopting the emerging semantic web technologies. The following topics are addressed in the attempt to achieve the Educational Semantic Web:

- Using ontologies and/or semantic Web technologies to decrease the complexity of ITS modelling and implementation (e.g. new types of modularized ITS architectures).
- Using semantic Web technologies to move from ITS to web-based ITS.
- Using ontologies and/or Semantic Web technologies to support authoring of ITS (e.g. modelling of authoring and engineering process, authoring feedback, authoring GUI).
- Semantic Web and ontologies for user modelling in ITS (e.g. shareability of UM, interoperability between different ITS on the basis of their UM).
- Using Web services for modularization of ITS and Web-based systems' components (e.g. UM service, domain modelling service, etc.).
- Web standards issues for ITS and intelligent Web-based systems.
- Real-world systems, case studies and empirical research for Semantic Web-based ITS.
- The topics can be approached from different perspectives: theoretical, systems engineering, application, case study and system evaluation, etc.

SWEL'04 follows the successful workshop on Concepts and Ontologies in Web-based Educational Systems, held in conjunctions with ICCE'2002 in Auckland, New Zealand. The workshop edition in 2004 is organized in three sessions held at three different conferences. The aim is to discuss the current problems in e-Learning from different perspectives, including those of Web-based ITS and educational adaptive hypermedia courseware, and the implications of applying Semantic Web and educational standards and technologies for solving them:

- **SW-EL'04 Session at AH, 23rd August, 2004**
Session co-chairs: Peter Dolog and Martin Wolpers
- **SW-EL'04 Session at ITS, 31st September, 2004**
Session co-chairs: Vladan Devedzic and Tanja Mitrovic
- **SW-EL'04 Session at ISWC, 8th November, 2004**
Session co-chairs: Riichiro Mizoguchi and Yukihiro Itoh

Researchers with interest in various aspects of Web-based educational systems get together at SW-EL'04 in order to discuss important issues in this field and present their latest results. We hope that the workshop will continue on the road map towards the realization of the vision of the Educational Semantic Web.

August, 2004

Lora Aroyo and Darina Dicheva

Workshop Committee

General co-chairs:

Lora Aroyo, Eindhoven University of Technology
Darina Dicheva, Winston-Salem State University

ITS 2004 session co-chairs:

Vladan Devedzic, University of Belgrade
Tanja Mitrovic, University of Canterbury

International Program Committee:

Peter Brusilovsky (University of Pittsburgh)
Paloma Diaz (Universidad Carlos III de Madrid)
Vanja Dimitrova (University of Leeds)
Erik Duval (Katholieke Universiteit Leuven)
Jim Greer (University of Saskatchewan)
Tsukasa Hirashima, (Hiroshima University)
Ulrich Hoppe (University of Duisburg)
Geert-Jan Houben (Technische Universiteit Eindhoven)
Mitsuru Ikeda(JAIST)
Judy Kay (University of Sydney)
Kinshuk (Massey University)
Erica Melis (Universität des Saarlandes, DFKI)
Ambjörn Naeve (Royal Institute of Technology)
Ossi Nykänen (Tampere University of Technology)
Gilbert Paquette (LICEF)
Simos Retalis (University of Cyprus)
Demetrios Sampson (Center for Research and Technology - Hellas (CERTH))
Katherine Sinitsa (Kiev University)
Amy Soller (ITC-IRST)
Steffen Staab (AIFB, University of Karlsruhe)
Julita Vassileva (University of Saskatchewan)
Felisa Verdejo (Ciudad Universitaria)
Gerd Wagner (Eindhoven University)

Table of Content

Semantic Web and Intelligent Learning Management Systems Goran Šimić, Dragan Gašević, Vladan Devedžić	1
A Multi-Agent and Service-Oriented Architecture for Developing Integrated and Intelligent Web-based Education Systems Fuhua Lin, Peter Holt, Steve Leung, Mike Hogeboom, Yang Cao	11
General Architecture Supporting Component-based EIS Interoperability Darina Dicheva, Lora Aroyo	21
Individualized Selection of Learning Object Jian Liu, Jim Greer	29
Domain Ontologies Integration into the Learning Objects Annotation Process Telmo Zarraonandía, Juan Manuel Doderó, Paloma Díaz, Sarasa	35
The Use of Ontologies in ITS Domain Knowledge Authoring Pramuditha Suraweera, Antonija Mitrovic, Brent Martin	41

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.

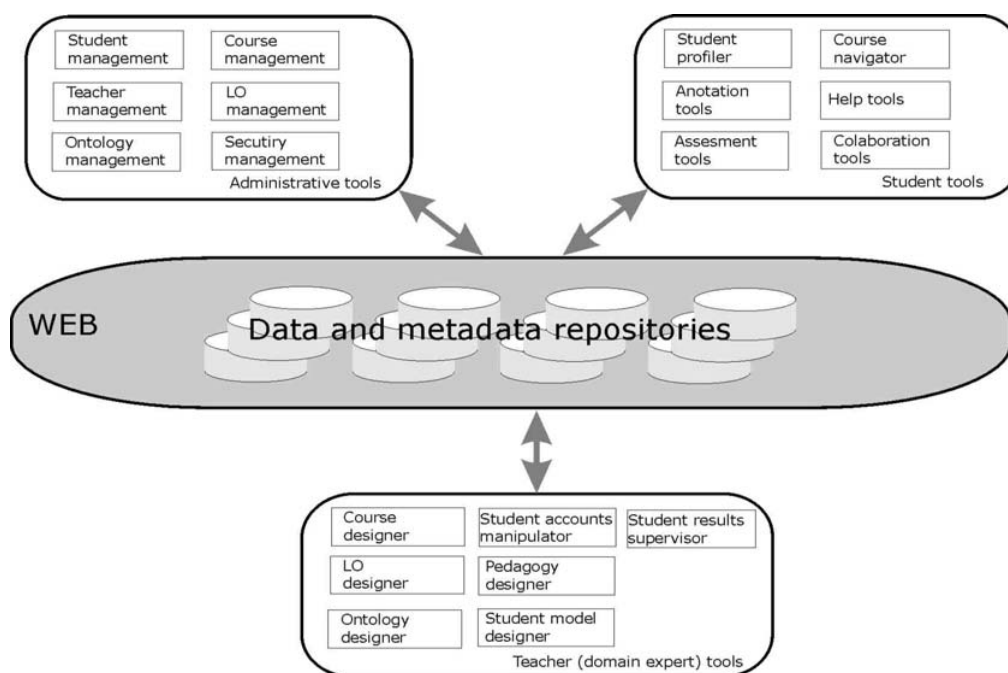


Fig. 1. The ILMS Architecture

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, *JavaTM 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.

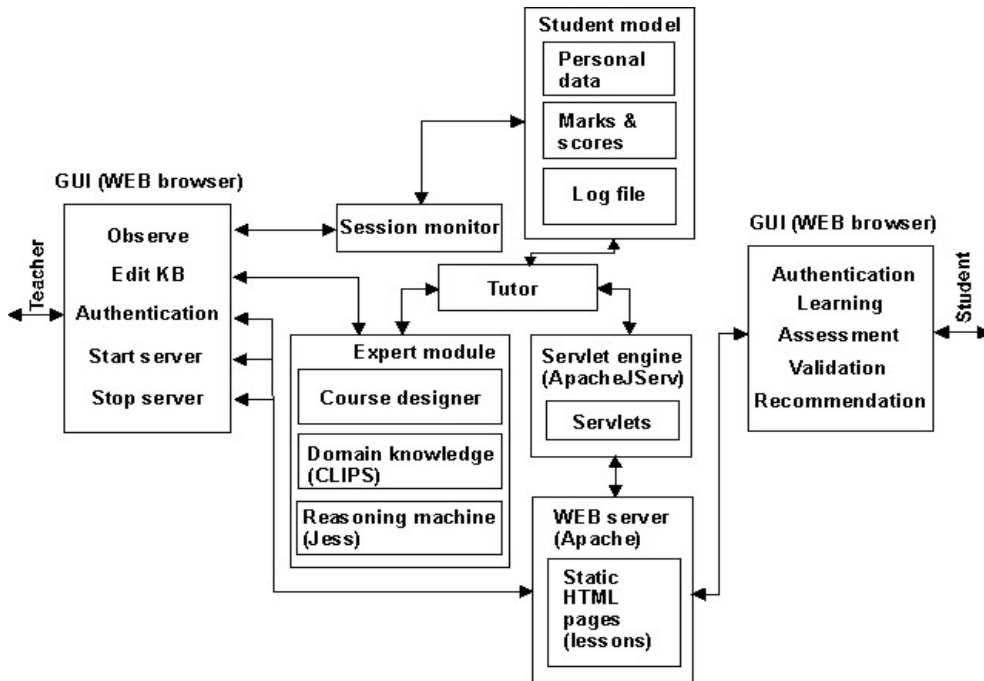


Fig. 2. The Multitutor architecture

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].

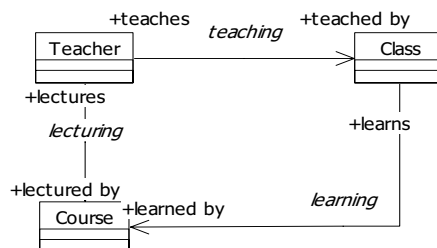


Fig. 3. The general concepts of the learning process

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.

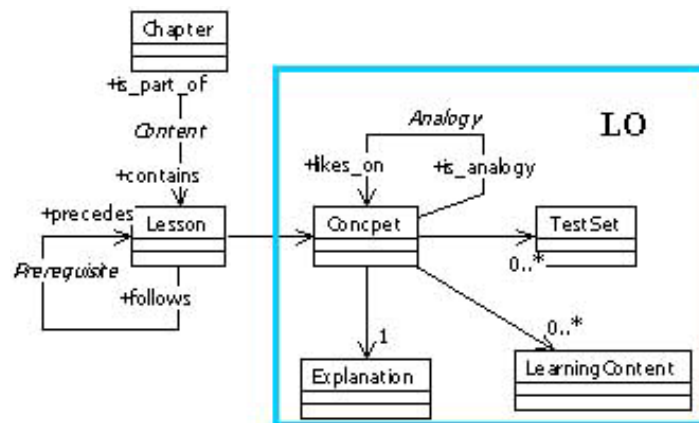


Fig. 4. The main concepts of the course ontology

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 *ProjectedSkill* 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.

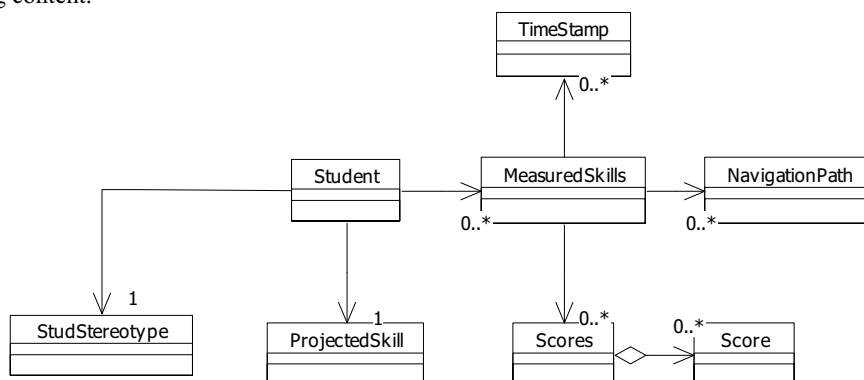


Fig. 6. The student ontology

4.4. Mutitutor Applications

In Mutitutor we have developed the Code Tutor educational systems for teaching radio-communications that we have already mentioned. In order to illustrate how Mutitutor 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 Mutitutor 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 Mutitutor system as well as examples of two learning applications developed in the Mutitutor. We especially stressed how the Mutitutor 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 Mutitutor 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 Mutitutor 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 Nejdil 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 learning objects and it would be a useful for Mutitutor. The navigation through learning materials as well as their findability 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 Mutitutor 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 Mutitutor courses can be described in the future. In fact, we can use an EML instead of the Mutitutor'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 Mutitutor 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 than 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 authors in the e-learning community defined ontologies of different kinds of knowledge in the last years. But, this raises many problems for developers as to 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. Weitzl, 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.

A Multi-Agent and Service-Oriented Architecture for Developing Integrated and Intelligent Web-based Education Systems

Fuhua Lin, Peter Holt, Steve Leung, Mike Hogeboom, Yang Cao

Center for Computing and Information Systems,
1 University Drive, Athabasca, Alberta, Canada, T9S 3A3
{oscarl, holt, stevel, yangc}@athabascau.ca

Abstract. This paper presents an architecture for developing integrated and intelligent Web-based educational systems (WBES) using multi-agent system technology and Web services technology. Intelligent agents are designed for support Web-based education, considering target population characteristics and topic areas, estimating relevant existing skills and knowledge of the learners, formulating objectives and learning outcomes, selecting appropriate learning objects, assembling courses and curricula, and assessment of learner performance. Web Services are designed for the modularization of WBES, and are excellent complimentary partners with intelligent agents, since they are characterized by their standardized communication protocol, interoperability, easy integration and development. To demonstrate the feasibility of the proposed approach, some agents and Web services for Web-based distance education are implemented.

1 Introduction

Web-based learning systems can potentially deliver personalized course material and services, and are, therefore, able to accommodate a larger variety of learners than what can currently be accommodated. To tap this potential, we propose a framework for an innovative WBES.

An innovative Web-based educational system should consider individual learners differences and the profiles of individual learners. It should be an integrated system, due to the dynamic and distributed nature of both resources and applications [1]. An innovative Web-based educational system should not merely respond to requests for information, but should intelligently adapt and actively seek ways to automate tasks to reduce the ever-increasing information workload.

We are working on designing, implementing, and evaluating, with real users, an agent-supported online educational environment able to support learners during the whole cycle of learning, and able to support educators in curriculum planning, course designing and delivering, tutoring, and learner performance evaluation.

The agent-based approach is suitable for supporting Web-based education since relationships among learners, courses, and instructors last for a considerable period of time [2]. Due to the inherent distributed nature of Web-based learning, a Web-based educational environment can be enhanced by a set of software agents [3, 4]. Much of the experimental research has shown that intelligent software agents have great potential for reducing information workload and for automatically performing many knowledge/labour-intensive tasks for both learners and educators [5]. However, people have faced many challenges in developing this technology for commercial applications, mainly because of the lack of an accepted industry-standard method for the development and implementation of agent-based systems [6] and environments where agents can live and run.

Since Web Services technology is characterised by standardized communication protocol, interoperability, easy integration and development, it provides an excellent architecture for developing service-based learning technology systems. For example, the learning services architecture and learning services stack have been proposed and developed by the Learning Systems Architecture Lab at Carnegie Mellon University [7]. However, when compared to agents, Web Services has some

limitations [8]. For instance, Web services are passive until invoked while agents are inherently communicative. Overcoming these limitations appears to require the integration of agents and Web services. In this paper, we propose an architecture in which Web services are used for modularization of WBES and are excellent complimentary partners with software agents in integrated and intelligent WBES.

2 The Proposed Architecture

In the proposed architecture, an integrated and intelligent WBES consists of users' personal agents, task agents running on distributed agent platforms, and a set of discoverable Web services. The Agent platforms are basic facilities for creating, deleting, and locating agents. They also involve inter-agent communication.

2.1 Personal Agents

A personal agent (PA) or user interface agent (UIA) is a GUI-driven interface between a user and an agent-based learning environment. Through a PA, a user can delegate rights to his/her agent, manage task agents within the environment and configure the options provided by the task agents. Users can meet their PA's either by running an application or opening a secure Web page in a desktop/laptop/pocket PC. There are two main kinds of PA's: instructor PA's and learner PA's.

An instructor PA is an assistant to the instructor, helping the instructor generate, deliver, and maintain online courses. These kind of agents interact with and mediate curriculum planning agents, course delivery agents, course update agents, learning object recommendation agents, and notification agents to fulfil the tasks delegated by the instructor or respond to requests from learners or learner PA's.

A learner PA is a simulated instructor that can provide adaptive course material and appropriate instruction according to the learning process of the individual learner. These kind of agents can be viewed as an authoritative representative of the course author, the instructor, or the tutor. Learner personal agents manage and configure program-advising agents, tutoring agents, performance-monitoring agents, and collaboration agents located in the agent platforms of the environment.

2.2 Task Agents

A task agent plays either the role of a client of a Web service or the role of a supporter of a Web service. As a client of Web services, an agent can perform searches of different entries stored in a UDDI, and can contain, reason about the semantics of Web services, and mediate and compose Web services. It then can make message- and RPC-style calls to a Web service. As a supporter of a Web service, a task agent facilitates and enables the service. The Web service benefits from the ability of the agents to perform the task autonomously and intelligently, dynamic creation of agents, and semantic level communication. Most Web services in Web-based learning environments can profit from the flexibility and robustness, autonomy, and intelligence of agents. For example, we need 'spiders' [9] which are Web agents to support course information Web services by monitoring and maintaining Web-course materials [10]. Another example is agents for learning object repositories. Vast educational resources available today and tomorrow simply could not function without being able to delegate to agents the multitude of tasks that would otherwise be left to armies of people to handle [11].

A task agent is required to perform certain specific tasks, such as providing services, knowledge, and information resources, and also providing intermediary functions such as coordinating and communicating with other task agents. Therefore, a task agent's memory contains specific task-related information in greater depth than the personal agent's memory can possess. A task agent usually has a monitoring and learning function, which allows it to update its own information through new updates when necessary. Because a task agent is deemed a "common resource" shared by many users, its processing capability comprises a spooling function, in which requests are queued in accordance to their priority. In performing multiple tasks, the resource allocation function determines how many resources should be provided to each uncompleted task.

2.3 Web Services

Web Services for Web-based education can include knowledge management and information resource management located in different places. Knowledge management Web services manage, locate, analyse, and retrieve knowledge for Web-based learning – for example, domain knowledge and curriculum planning knowledge. Information resource management includes ‘learner information management’, ‘staff information management’, ‘course information management’, and ‘educational resource information management’.

These knowledge and resource management Web services are also responsible for retrieving the knowledge and resource needed. For instance, a domain ontology Web service is designed for providing services about a taxonomy database. It is used for a target language for (1) terms in the prerequisite and post-conditions of learning objects, and (2) the terms in the learner profiles.

2.4 Agent Management and Deployment Service

An agent management and deployment service is implemented through the Web technology. A registered user can login to download his/her favourite personal agent(s). The agent management service assigns unique agent identification to agents and records agent information such as agent types.

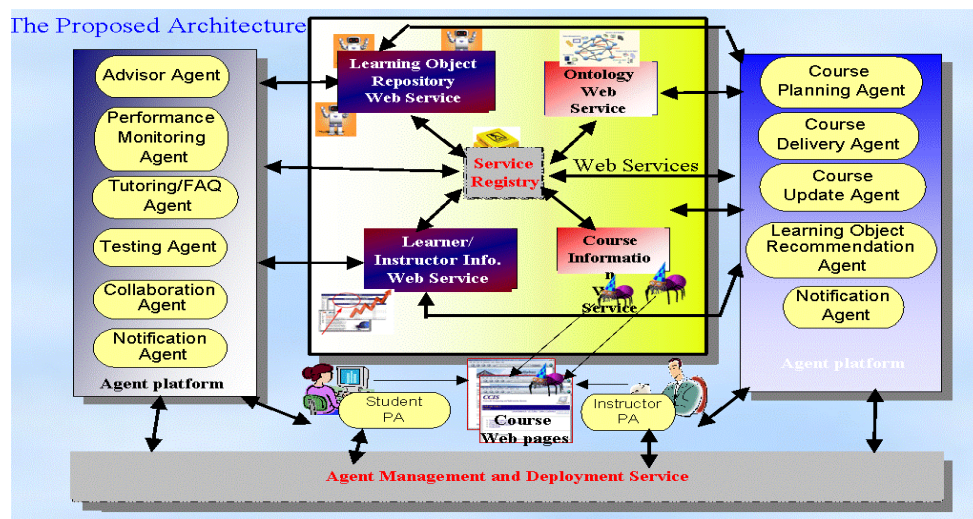


Fig. 1. The proposed architecture.

3 Implementation

3.1 The Agent Platform

We use Apache Axis as the SOAP engine and create a dispatcher that accepts SOAP remote procedure calls as input. We wrote the WSDL describing the method calls that our Web Services will accept by following a combination of the WSDL and schema specifications. We next published information about the Service Provider to a Service Registry, agent UDDI registry. There we entered in data about the Web site, such as the URL of the WSDL for the Web service.

A task agent on the agent platform performs a ‘find’ operation on a service registry. The agent finds the entry for our service and uses the listed URL to download a copy of the WSDL. Using the WSDL, the agent generates a program to serve as the Service Requester to access the service. When this is complete, the agent tests it by requesting that it perform a “bind” operation on the Web service. After the “bind” operation is successful, the agent passes the request and waits for a response.

3.2 Notification Agent

The Notification Agent makes use of JavaMail class to perform the actual sending. The agent gets the sender or recipient information from the Web services through sending XML request messages.

3.3 Learning Object Repository Web Services

The learning object repository Web services are provided by the internal eduSource infrastructure. The initial set of Web services available in the eduSource network are:

- Storing learning objects,
- Storing learning object metadata,
- Tagging tools for creating metadata records,
- Searching learning objects,
- Aggregating learning objects into lessons and
- Handling copyrighted materials.

The backbone of the interoperability is the eduSource Communication Language (ECL) [12] that implemented the core functions defined in the IMS digital repository interoperability reference model.

3.4 Ontology Web Services

We developed Ontology Web services that support Web-based learning in a language and platform-independent manner by using Protégé 2000 (<http://protege.stanford.edu>), a domain-modelling tool from Stanford University [13]. Protégé generates a default form when the domain is created, which can then be further customized to suit the project visual preferences and requirements. Once the domain model and data entry forms have been created, the instance tab, which is a knowledge acquisition tool, can be used to acquire instances of the classes defined in the ontology. Once the model has been populated with information, the Protégé library can be accessed using a Java API to retrieve that information for use in the Java Web Services. The domain models are being populated with IEEE/ACM Software Engineering Body of Knowledge 1.0 (SEBOK) (<http://www.swebok.org>) and ACM Computing Classification Systems (<http://www.acm.org/class/>). In the future, other publications can be added. The service can be invoked and a service side business command can be executed to retrieve the matching content based on the input ID [14].

For example, the “GetTopicsServices” is used to list all the topics (Subject areas) in the Protégé 2000 knowledge base. The service will be invoked and a server side business command can be executed to retrieve the topics. Just like any other WSDL document, the *GetTopics* WSDL is simply a set of definitions. Services are defined using six major elements:

- *Types* - Provides data type definitions used to describe the messages exchanged.
- *Message* - Represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system
- *PortType* - A set of abstract operations. Each operation refers to an input message and output messages
- *Binding* - Specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType
- *Port* - Specifies an address for a binding, thus defining a single communication endpoint
- *Service* - Used to aggregate a set of related ports

The following describes the WSDL file for the *GetTopics* Web Service. This Web service is used to return all the topics (Subject Areas) that are contained within the eLearning knowledgebase.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="GetTopicsService"
targetNamespace="urn:Foo">
```

The following *types* node provides the data type definitions used to describe the messages exchanged between the Web Service and the client. The *GetTopics* Service contains two complexType nodes. The first *complexType* describes the array of topics (*ArrayOfTopic*), while the second describes the actual

topic. The Topic *complexType* maps to the Topic Java object, while the *ArrayOfTopic* maps to both the root array of topics returned from the Java and the subtopics contained within each Java object. Both complexTypes in the following XML reference each other to create the same structure as the Java class as depicted in Figure 1 above.

```
<types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11-
    enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:Foo">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/">
    <complexType name="ArrayOfTopic">
      <complexContent>
        <restriction base="soap11-enc:Array">
          <attribute ref="soap11-enc:arrayType"
            wsdl:arrayType="tns:Topic[]"></restriction></complexContent></complexType>
        <complexType name="Topic">
          <sequence>
            <element name="subtopics" type="tns:ArrayOfTopic"/>
            <element name="title" type="string"/></sequence>
          </complexType>
        </schema></types>
```

The message nodes of the *GetTopics* WSDL file provide an abstract definition of the data transmitted. There are two message nodes defining the request and response. The part node of the response message describes the logical abstract content of the response message, being *ArrayOfTopic* in this case.

```
<message name="GetTopicsIF_getTopics"/>
<message name="GetTopicsIF_getTopicsResponse">
  <part name="result" type="tns:ArrayOfTopic"/>
</message>
```

PortType node describes the set of abstract operations and the abstract messages in the operation.

```
<portType name="GetTopicsIF">
  <operation name="getTopics" parameterOrder="">
    <input message="tns:GetTopicsIF_getTopics"/>
    <output message="tns:GetTopicsIF_getTopicsResponse"/>
  </operation>
</portType>
```

The binding node defines the message format and protocol of the communication.

```
<binding name="GetTopicsIFBinding" type="tns:GetTopicsIF">
  <operation name="getTopics">
    <input>
      <soap:body encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
        use="encoded" namespace="urn:Foo"/>
    </input>
    <output>
      <soap:body encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
        use="encoded" namespace="urn:Foo"/>
    </output>
    <soap:operation soapAction=""></operation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  </binding>
```

The service node simply groups a set of related ports together.

```
<service name="GetTopicsService">
  <port name="GetTopicsIFPort" binding="tns:GetTopicsIFBinding">
    <soap:address xmlns:wsdl=http://schemas.xmlsoap.org/wsdl/
      location="http://localhost:8080/elearning-jaxrpc/GetTopics"/></port></service>
</definitions>
```

3.5 ‘Spider’-like Course Information Monitoring Agent

Currently the Monitoring Agent is designed for monitoring online course material to (1) determine whether or not the links in those pages are broken and (2) determine whether or not the content in those pages have been significantly changed. The meaning of “significantly changed” is based on a couple of pre-defined criteria. For example, the number of hyperlinks or photos increased or decreased, or the content lengths of the Web page by examining its MIME header.

If the monitoring agent discovers such changes, it can trigger a Notification Agent to send out a message to those students who are interested to receive the message. Another way is to store the event of changes in the database. The instructor’s Notification Agent invokes the course information Web Services to notify the instructor of the Updated course information (e.g. newly-broken links in the course materials) or to recommend an alternative link from a learning object repository [10].

Most of the work is done by agentized and multi-threaded class Spider [9]. ‘Spiders’ are programs that can visit Web sites and follow hyperlinks. We have successfully implemented the agent system for Web-based course link maintenance using the architecture above. The agents and the agent platforms were written in Java and deployed at different locations. More than ten courses of CCIS of Athabasca University of Canada have been used for the testing. The agent then checks for broken links (by the spider) and writes the results into MySQL databases. The notification agent can send emails to the course instructors if some broken links have been found. The instructors can configure their agents via running their instructor Personal Agents.

3.6 MARC-IEEE LOM/CanCore Converter

Athabasca University has developed an IEEE LOM/CanCore compliant metadata repository application called ADLiB (<http://ADLiB.athabascau.ca/>), which is a Web application for creating and storing standards compliant metadata records, and for storing their corresponding learning objects in the repository. As part of eduSource Canada project, a MARC-IEEE LOM/CanCore converter (<http://emd.athabascau.ca/courses/crosstalk/converter.html>) has been developed by some of the authors. The tool enables XML-based digital repositories to interact with and harvest metadata from MARC (Machine Readable Cataloguing) records that are used in libraries. MARC records contain hundreds of fields and were developed to describe print materials [15].

3.7 Learning Object Recommendation Agent

Based on Learning object repository Web services, a Learning Object Recommendation Agent has been implemented, which is designed for supporting Learning Objects Repository users in identifying and accessing learning objects according to personalized specifications (preferences) that have been dynamically interpreted. We expect that the agent can work on behalf of the user, monitoring the new arrival of learning objects in the learning object repository and then notifying the user when relevant learning objects are deposited in the repository (<http://ADLiBx.athabascau.ca/lora/jsp/lora/index.jsp>).

3.8 Course Planning Advisor Agent

We have developed a prototype of Advisor Agent to help students and/or student advisors to plan their study. The goal of the agent is to provide a list of reasonable alternate program plans for a student. Although the final decision will be up to the student, his academic advisor and the program coordinator, the Advisor Agent will respond to the latest changes in the environment and promptly advise the involved parties. To facilitate the planning process, the agent will take into consideration course structure information, individualized students’ background, program constraints, and a list of extensible searching heuristics. We create an example to demonstrate the Advisor Agent. In this example we have a program that consists of 5 courses for Program: B.Sc. in Software Development (SD) (see Fig. 2).

We have two students planning to finish the program. One of the students does not have any computer background, while the other has finished a course in another institution that equivalent to SD303. Responding to this request the Advisor agent will advise possible plans (we will call them *program plans* hereafter) to the students and their academic advisor.

Course Structure information and Course Dependency: A program plan is a step-by-step list of courses in a certain sequence that should be taken by a student. The sequence of the courses in a program is built entirely on the idea of “course dependency”. For example, if a course, C1, is said to be dependent on another course, C2, a student will be required to complete C2 before starting C1. However, to establish such a relationship we need a common mapping of all relevant courses to a common course content repository. Though not without controversy, there exist some common course content repositories. For example in the domain of Software Engineering, SWEBOK is a well-known initiative to establish such a standard. The example used below is taken from SWEBOK.

Table 1. Knowledge Areas in SWEBOK
Knowledge Areas 1st level

KA's ID	KA1	KA2	KA3	KA4	KA5	KA6	KA7	KA8	KA9	KA10
KA's Name	Software Requirement	Software design	Software construction	Software testing	Software maintenance	Software configuration	Software engineering management	Software engineering process	Software engineering tools and methods	Software quality

We adopted the “Bloom’s Taxonomy” [16] to represent the intensity of understanding of course content. Combining the Bloom levels and knowledge areas, we can analyse the content of each course as:

ID	Name	Knowledge Areas
SD201	Fundamental of Programming	(KA1,1) (KA2,1) (KA3,1) (KA3,2) (KA3,3) (KA4,1)
SD303	Software Development	(KA2,2) (KA5,1) (KA7,1) (KA8,1) (KA8,2) (KA10,3)
SD324	Human Computer Interaction	(KA1,1) (KA2,2) (KA2,3) (KA9,1) (KA10,1) (KA10,2)
SD389	Project Management I	(KA1,3) (KA4,2) (KA6,1) (KA7,2) (KA7,3) (KA9,2)
SD423	Project Management II	(KA2,4) (KA5,2) (KA5,3) (KA6,2) (KA9,3)

Fig. 2. A course design pattern.

By assuming Knowledge Area dependency we derive the dependency relationship of the courses as shown in Fig. 3. For any two courses C1 and C2, if:

- Independent: Both C1 and C2 can be taken at the same time and at any sequence.
- Parallel: Both C1 and C2 can be taken at the same time and at any sequence.
- C2 is dependent on C1: C1 must be completed before taking C2, which means that C1 and C2 cannot be taken together.
- C2 is parallel dependent on C1: C1 & C2 can be taken together but C2 cannot be taken before C1.

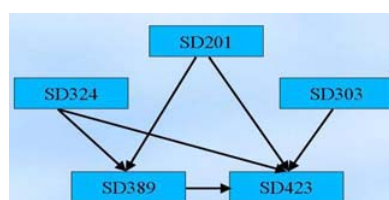


Fig. 3: An example of the dependency relationship of courses.

Assume the starting point of the program is SD201 (see Program Constraints below). Given the dependency relationship in our example we will have 4 alternate paths for students without any program background:

- SD201 → SD324 → SD303 → SD389 → SD423;
- SD201 → SD324 → SD303 → SD423 → SD389;
- SD201 → SD324 → SD389 → SD303 → SD423;
- SD201 → SD303 → SD324 → SD389 → SD423.

To achieve *individualized* course plans, the advisor agent considers the courses the students have taken before. However the presumption of this approach is that the courses can be mapped to the Knowledge Area repository. Since Student A does not have any computing background, his possible program paths will be all 4 of the above. For student B, since he already finished a SD303 equivalent course, he will have only one path: SD201 → SD324 → SD389 → SD423.

A *program constraint* is a factor that affects a program plan from the point of view of the program. In our implementation we have two constraints:

- Starting point: a student must start with a certain course.
- Semester, minimum and maximum courses: courses are grouped into a semester. Course taken in the same semester are considered as taking the courses at the same time. A student can take a minimum number of courses up to a maximum number within a semester. In our example, we prescribed that:

$$1 \leq \text{no. of courses/per semester} \leq 2$$

By applying these constraints to the above course path for student B we will have 4 alternate plans:

(S1: SD201, SD324) (S2: SD389, SD423)
 (S1: SD201, SD324) (S2: SD389) (S3: SD423)
 (S1: SD201) (S2: SD324) (S3: SDP389, SD423)
 (S1: SD201) (S2: SD324) (S3: SD389) (S4: SD423)

Program constraints are not limited to the above two examples. Other conditions, like course availability, number of credits per course and financial constraints, can be added to the consideration.

Heuristics are introduced to achieve more reasonable solutions. Considering all possible program plans and the 2 constraints, there are as many as 20 different plans for student A who does not have any computing background. Heuristics is not a new idea; it introduces rules that are obvious to human but not to computer programs. In our case, heuristics are used to eliminate excessive course paths. We introduce only one heuristic: there are not more than 3 semesters in each plan for the students. Therefore in the above example, the Advisor Agent suggests only plans #1, #2 and #3.

To take full advantage of Web services and agent architecture we did not implement everything into one single application and did not run all of the functions on the same platform. The Advisor Agent itself does not maintain/update any of the information it requires for the advising functionality. Instead, it will try to retrieve the information it requires. The Advisor Agent will keep a list of end-points of the Web services that holds the information. Namely, we have a Web service for student information, including the background of the student, a Web service for Course Repository and a Web service for Knowledge Area Repository. In the future, we would maintain a yellow pages service for the Web services in standard UDDI format so that we can eliminate the need of storing Web services end points in the Advisor Agent. Information, however, must be available on a format that can be interpreted by the Advisor Agent. The logical choice will be XML format. A sample student profile looks like:

```
<Student>
  <demographics>
    <id>1234567</id> <name>Steve Leung</name>
    ..... </demographics>
    <goal> <degree>BSc</degree>
  </program>Software Engineering </program> </goal>
    <experience> <course> <source_ID>CSC121</source_ID>
      <source_name>Introduction to Programming</source_name>
      <grade>85%</grade>
    <equivalent_ID>SD201</equivalent_ID>
  </course>... </experience>
```

A sample Course Repository looks like:

```
<Course_repository>
  <Course> <id>SD201</id> <description>Fundamental_of_Programming
</description>
<KA><label>KA1</label><bloom>1</bloom></KA>
```

```

<KA><label>KA2</label><bloom>1</bloom></KA>
<KA><label>KA3</label><bloom>1</bloom></KA>
<KA><label>KA3</label><bloom>2</bloom></KA>
<KA><label>KA3</label><bloom>3</bloom></KA>
<KA><label>KA4</label><bloom>1</bloom></KA>
</Course> ... </CourseRepository>

```

A sample Knowledge Area Repository looks like:

```

<KA_repository>
  <KA> <label>KA1</label>
    <description>Software_Requirement</description>
  </parent/>
</KA>
...
</KA_repository>

```

In the future we will be developing schema for universal validation of the data. The sample of the program plans in this example is shown in Fig. 5.

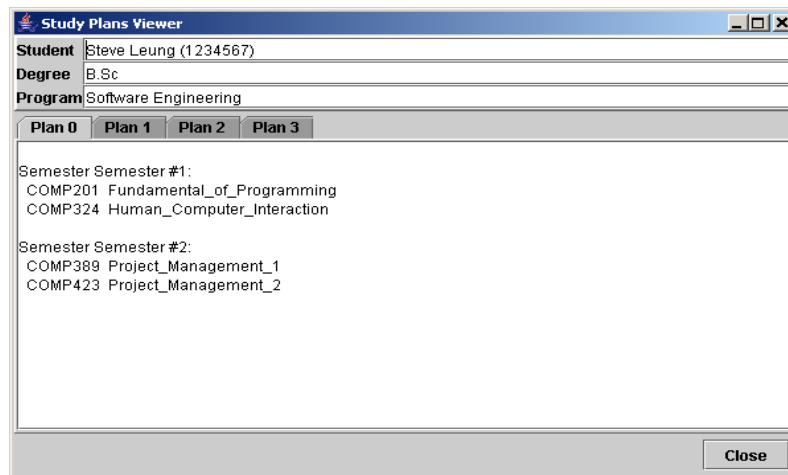


Fig. 5. A sample of the program plans.

Since the above is the client side program, it is easy to transfer the result on a Web server and make it available for enquiry using regular browsers. Moreover, one of the advantages of the agent-based approach over the Web services approach is the autonomous nature of an agent. Once the Advisor Agent starts, it will keep on running until it is instructed to stop. If there is any change to the student profile and course repository, the Advisor Agent will notice the impact on the program plans and inform the student and his academic advisor.

4 Conclusions and Future Work

Educational information standardization, educational resource development, Web technologies, and AI technologies have paved a way for Agent-Enhanced Web-based On-line Learning. To integrate agents into existing legacy learning environments or into heterogeneous learning environments, one may encounter many difficulties. Web Services technology provides a new way to integrate existing systems or applications, and the ability to access data in a heterogeneous environment and to provide interoperability of components and learning content.

We have proposed an approach to designing and developing adaptive Web-based learning environments by integrating agents and Web services. Some agents and Web services have been developed for both real applications and experiments.

We are working on the general issues related to performance monitoring and adaptation mechanism in the architecture. Also we will explore Adaptive E-Learning Based on Distributed Re-usable Learning Activities [17] and how to apply the current standardization efforts related to the Web Services Choreography [18] to the coordination of the agents in the architecture. Through our research into Web-based course generation and delivery, we are creating a distributed intelligent agent system that will allow instructors to generate timely, personalized and adaptive courses best suited for each student.

Acknowledgements

We would like to thank National Science & Engineering Research Council (NSERC) of Canada for sponsoring this research project.

References

1. Holt, P., F. Lin, H. Wang, G. Pu, A Learner-Centered Distributed Learning Environment, WSEAS Transactions on Communications, Issues 2-3, April 2003 and July 2003, ISSN: 1109-2742, pp271-276.
2. Chan, T-W, (1995), Artificial Agents in Distance Learning, International Journal of Educational Telecommunications, 1(2/3), 263-282.
3. Greer J., McCalla G., Vassileva J., Deters R., Bull S., Kettel L., (2001) Lessons Learned in Deploying a Multi-Agent Learning Support System: The I-Help Experience, *Proceedings of AIED'2001*, San Antonio, 410-421.
4. Baylor, A., (1999). Intelligent agents as cognitive tools for education. *Educational Technology*, Volume XXXIX (2), 36-41.
5. Thaiupathump, C., J. Bourne, J. Olin Campbell, Intelligent Agents for Online Learning, JALN Vol.3, Issue 2, November 1999.
6. Sturm, A., Dovi, D., Shehory, O., (2003), Single-Model Method for Specifying Multi-Agent Systems. *Proceedings of AAMAS, 2003, Melbourne*. 121 – 128.
7. Blackmon, W., & Rehak, D. (2003). Customized Learning: A Web Services Approach. *World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003*(1), 6-9.
8. Huhns, M. N., (2002), Agents as Web Services, *IEEE Internet Computing*, July/August, 93-95.
9. Heaton, Jeff, Programming Spiders, Bots, and Aggregators in Java, 2002, Sybex.
10. F. Lin and L. Poon, Integrating Web Services and Agent Technology for E-Learning Course Content Maintenance, Innovations in Applied Artificial Intelligence, B. Orchard, C. Yang, M. Ali (eds.), LNAI 3029, Springer, pp. 848-856.
11. F. Lin and L. Esmahi, Integrating Agent Technology and Web Services into Distributed Learning Environments, in book edited by F. Lin, Designing Distributed Learning Environments with Intelligent Software Agents, IGP, 2004.
12. Hatala, M., G. Richards, T. Eap, and J. Willms, The eduSource Communication Language: implementing open network for learning repositories and services, Symposium on Applied Computing Proceedings of the 2004 ACM symposium on Applied Computing, Pages: 957-962, 2004, ISBN:1-58113-812-1
13. Hogeboom, M., (Master Thesis), Domain Modelling for E-Learning Using PROTÉGÉ 2000 with ontology Web Services, Athabasca University, 2004.
14. Hogeboom, M. and F. Lin, Developing Domain Model Web Services for Agent-Supported distributed Learning Using PROTÉGÉ 2000, IEEE Learning Technology newsletter, vol. 6 Issue 2, April 2004.
15. Y. Cao, F. Lin, R. McGreal, S. Schafer, N. Friesen, T. Tin, T. Anderson, D. Kariel, B. Powell, and M. Anderson, Facilitating E-Learning with a MARC to IEEE LOM Metadata Crosswalk Application, Innovations in Applied Artificial Intelligence, B. Orchard, C. Yang, M. Ali (eds.), LNAI 3029, Springer, pp. 739-748.
16. Anderson, L.W., & Krathwohl (Eds.). (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York: Longman.
17. Brusilovsky, P. and Nijhawan, H. (2002) A Framework for Adaptive E-Learning Based on Distributed Re-usable Learning Activities. In: M. Driscoll and T. C. Reeves (eds.) Proceedings of World Conference on E-Learning, E-Learn 2002, Montreal, Canada, October 15-19, 2002, AACE, pp. 154-161.
18. Web Services choreography: <http://www.webservices.org/index.php/article/view/1178/>

General Architecture Supporting Component-based EIS Interoperability

Darina Dicheva¹ and Lora Aroyo²

¹ Winston-Salem State University, Computer Science Department,
Martin Luther King Jr. Drive, Winston Salem, NC 27110, USA
dichevad@wssu.edu

² Eindhoven University of Technology,
Department of Mathematics and Computer Science,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
l.m.aroyo@tue.nl

Abstract. Web-based Educational Information Systems (EIS) aim at providing the learner with immediate, on-line access to a broad range of structured information in order to support more efficient educational task performance. Currently, more and more efforts are concentrating on bringing all those systems to work together in order to provide better support within the context of web-based education. Our goal is to approach the problem from a rather practical and somewhat minimalist perspective: to better utilize resources and components of already existing EIS we show how through communication protocols, we can realize a general modular architecture comprising components that can be shared and interchanged.

1 Introduction

There is an increasing interest to online learning support systems that are aimed at providing resources and functional components for various educational goals and tasks. Such systems often need to interoperate, collaborate and exchange content or re-use functionality, in order to support a richer set of educational functions and increase their effectiveness.

Adaptive Web-based Educational systems (AWBES) as introduced by Brusilovsky [1] use techniques from Intelligent Tutoring Systems (ITS) and Adaptive Hypermedia (AH) to support courseware authors in combining different systems with different purposes to provide for different aspects of the instructional process and achieve an integrated solution. Another way of achieving such integration is to build the separate systems using component-based architectures, where all the functionality is modularized and well encapsulated.

Within the class of web-based educational systems, a major role in various instructional contexts play the Educational Information Systems (EIS) that are aimed at providing intelligent, task-centered information support for solving problems and performing learning tasks. Consequently, considerable effort is currently focused on defining frameworks and architectures to tackle issues of information support from multiple perspectives. On the one hand, we do have the example of monolith Learning Management Systems (LMS), such as Blackboard and WebCT, which on more or less superficial level cover various teaching, learning, and administrative activities and as a result provide web-enhanced courses. On the other hand, we see multiple examples of specialized and effective educational systems and content providers, which support only one task/function within the entire educational process. Representatives of such systems are adaptive textbooks constructed with AHA! [2], InterBook [3] and NetCoach [4], or adaptive courses within ELM-ART [5], PAT Online [6] and AIMS [7]. There are also more global but still highly specialized efforts, such as ARIADNE and EdNa courseware-reusability frameworks that provide repositories of re-usable educational objects.

Brusilovsky [1] claims that all those systems need to be integrated and that the “university has a clear need in a single integrated system that can support all critical functions in one package”. While we

basically agree, we believe that it is not feasible to expect reaching the one-integrated-system goal in a near future and our claim here is that instead of a complete integrated system, what is currently needed is a good standardized way to allow different specialized systems *to talk to each other*. Such an approach from one side proposes a solution for his concern that “modern AWBES are designed to be used as a whole, not component by component” and from the other fits nicely to his advocacy of “distributed component-based architectures for building adaptive systems”.

In the current efforts targeting integration of various educational systems and content providers, Devezic [8] proposes educational servers (INES), which are based on using standards, ontologies, and pedagogical agents to support interaction between clients (authors and students) and servers (hosting educational content and services). He claims that the interaction in the future educational systems will be between learners and services through educational service directories. We argue that WBES interactions in the near to medium future will be between educational systems’ components - before achieving the large scale service-based integration, we need to explore and exploit possible communication between components.

Another service-oriented perspective on the integration is given by the Elena project [9], which defines a smart learning space of educational service providers based on the Edutella [10] peer-to-peer framework for interoperability and resource exchange between heterogeneous educational applications and different types of learning resource repositories. In the same context, we also see specific efforts trying to fill the gap between adaptive educational systems and dynamic learning repository networks, by proposing service-based architectures for personalized e-learning. An example is the Personal Learning Assistant [11], which uses Semantic Web technologies for realizing personalized learning support in distributed learning environments.

In this paper we try to approach the integration problem of the systems from a rather practical perspective and propose a general framework for supporting communication between ontology-based EIS aimed at utilizing systems’ resources and components. It employs some features of web services and agent-based frameworks, but is intended to be much simpler.

The paper is organized as follows. After identifying the common characteristics of concept-based EIS (Section 2), we outline the current needs and requirements for them to interact and share knowledge and resources (Section 3). In Section 4 we depict a general service-oriented framework to support the interoperability of various concept-based EIS. We conclude with a short discussion.

2 Ontology-driven Educational IS

The main goal of web-based EIS is to provide the learner, on the one hand, with immediate, on-line access to a broad range of structured information and on the other, with domain-related help in the context of her work, thus supporting more efficient task performance. There are a number of concept-based EIS already developed [3,4,5,7,11,12], which typically include:

- Concept-based (ontology-driven) subject domain
- Repository of learning resources (digital library)
- Course (learning task) presentation
- Adaptation & personalization

The fundamental feature of such systems is the subject *domain conceptualization*. It supports not only efficient implementation of required functionality but also standardization: the concept structure can be built to represent a domain ontology providing an agreed vocabulary for domain knowledge representation. Thus the ontology specifies the concepts to be included and how they are interrelated.

The repository contains learning resources (objects) related to the subject domain concepts. We can think of the resources as being *attached* to the domain concepts they describe, clarify, or use. One of the most prominent themes in ontology research is the construction of reusable components. If the attached objects have also a standards-based representation as opposed to a proprietary representation, this will insure that the application’s content is reusable, interchangeable, and interoperable.

Course/learning tasks are typically described in terms of subject domain concepts and some *instructional* relationships (such as ‘prerequisite’, ‘uses’, etc) between the involved concepts. The domain

concepts are also used as a basis for implementing systems' adaptive behavior. The latter involves constructing learner models in terms of domain concepts, performed tasks, and user characteristics.

An EIS user is typically involved in exploring the subject domain ontology and searching the repository for information related to a specific task. Good examples of such systems are AIMS [7] and TM4L (Topic Maps for Learning) [12], which we use as a basis of our discussion. AIMS and TM4L both focus on providing contextual support that enables learners to identify information necessary for performing a specific task (e.g. course assignment). They can be used standalone (for example, as an extension to a traditional or on-line distance course) or integrated in a larger electronic learning environment that allows the users to perform open learning tasks in a specific subject domain. Since both focus on efficient information provision and support for task-oriented problem solving, these systems are quite similar but they can be also seen as complementary in the way they support learning tasks. While AIMS includes course representation and sequencing, TM4L is a kind of digital library, which does not include direct course representation.

3 Need for EIS Communication

Integration and interoperability are very important for EIS systems. If interoperable, two systems can benefit of additional functionality supplied by the other, and especially of sharing resources and common components, e.g. user models. In our example of AIMS and TM4L, TM4L can use AIMS course sequencing model, graphical viewer, and resource metadata, while in turn AIMS can use TM4L external and internal resources, domain and resources merging capability, text and external search.

We start our discussion on EIS system communication with presenting two use case scenarios illustrating the communication between the two considered systems.

Scenario I: A learner uses AIMS as a support tool in her coursework. When trying to solve a specific task, however, she is not satisfied by the informational support provided by AIMS since it is not enough for her to achieve her learning goal. She announces this and AIMS seeks external help (more relevant information). It sends a request to TM4L for more learning resources on the topic at hand. TM4L responds to AIMS providing information that it has. AIMS feeds this information back to the learner. Fig. 1 illustrates this scenario by showing the internal organization of the two systems and the interface which combines the task-based search from AIMS with the resources from TM4L.

As we mentioned already, ontology-driven EIS have concept-based representation of the specific subject domain and learning resources *indexed* by domain concepts. Consequently, such a system would *understand* requests for information expressed in terms of domain concepts and relationships between them. Concerning a specific concept, possible requests of AIMS to TM4L would be:

- Give direct parents (children) of concept X (i.e. the concepts in a direct 'is-a' relation with it).
- Give all parents (children) of X (i.e. all concepts on the same path as X).
- Give all relations in which concept X plays a role (i.e. concept X is involved).
- Give all concepts related (in a relation of any kind) to a given concept.
- Give all available resources connected with this concept.
- Give learning resources of a kind (e.g. using the LOM standard) connected to concept X.
- Give all available information related to concept X (everything listed above).

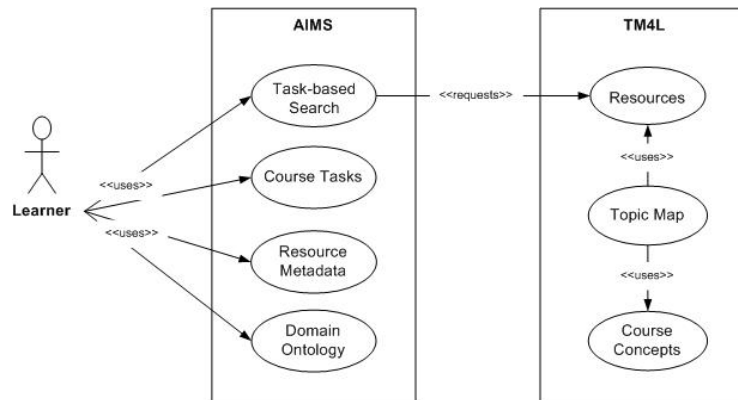


Fig. 1. AIMS – TM4L communication.

Scenario II: An author uses AIMS to construct a course. When preparing the course, she would like in addition to the learning resources she defines to reuse some other resources relevant to the course topics. She announces this to AIMS. It sends a request for information on specific topic(s) to TM4L. TM4L responds to AIMS providing information that it has. AIMS feeds this information back to the author to be approved for storing in the AIMS DM or disregarded.

Possible *requests* in this scenario include:

- Give all available resources connected with a concept X.
- Give learning resources of a kind (e.g. using the LOM standard) connected to concept X.
- Give all available information related to concept X.
- Import the entire domain ontology.

The communication between the two systems can be realized either at a system level or at a component level (in the second case between components with either identical or different functionality, e.g. domain models, user models, etc). The latter imposes additional requirements to the architecture of the involved systems – they should have a clear component-based architecture.

The main research questions related to implementing communication between the systems include:

1. *Level of granularity of information exchange:* what should be the information contained in one communication transaction? For example, in the second scenario, should the author be allowed to ask about importing the entire TM4L domain model in one transaction?
2. *Request semantics:* What kinds of questions the requesting system should be able to ask?
3. *Request syntax:* In what a form the questions should be expressed?
4. *Domain or user model awareness:* Should the requesting system send any indication about what it “knows” or its user already “knows”, so that the responding system doesn’t send information already known? If so, what kind of information and in what a form?

We attempt to answer these questions at two levels – a general one and a minimalist one - providing guidelines to the design of two corresponding frameworks for ontology-based EIS communication support. While the general one provides a powerful service-oriented framework to support efficient communication between component-based EIS, the minimalist one is intended to provide an efficient *current* solution for supporting shareability and exchangeability of systems resources. We started with defining the general framework with the intention to further constrain it to the desired minimalist one.

4 General Architecture for Component-based EIS Interoperability

If we consider the problem of efficient construction of EIS systems that complement (serve as advisors to) each other by sharing resources and components, the obvious answer is modularized building of such systems. This implies a component-based architecture that allows sharing knowledge (e.g. domain ontologies, learning resources, course models, and user models) and components (e.g. user modeling,

course sequencing, ontology visualization, keyword search). The systems will typically have different domain (and other) models and the interoperability between them should include support for:

- translating between systems' domain representations
- sharing learning sequencing components (educational context, results, etc.)
- sharing learning resources
- sharing user models.

4.1. The Architecture

In order to define such a general architecture we address the main research questions formulated in Section 3. With regard to the grain size of exchanged information (Question 1), we consider two issues as important for the communication quality and efficiency between the systems:

- Conciseness of the request and the reply.
- Completeness of the request and reply.

These factors suggest that a system request has to ask for precise information related to a particular user query (topic) and the reply is to be as specific and detailed as possible. This prompts us to propose that a finer level of granularity of information exchange is more appropriate. In some cases, however, as in Scenario II, we might require that the framework supports importing or merging of the complete (domain) models (e.g. TM4L DM merged with AIMS DM). We propose that this is realized through using appropriate communication support services and not through the *ordinary* request/response type of communication between the systems.

Concerning Question 2 from Section 3, in order to “understand” each other both systems must “know” the basic terms for structuring and using ontology-based learning resource repositories, such as ‘concept’, ‘relation/association’, ‘role in a relation’, ‘resource’, etc., which make the common ground for the semantic understanding. In other words, the different systems must know how to *map* their internal knowledge to the basic concepts of this common ground. This lightweight mapping process, as opposed to very expensive reasoning processes, is a key aspect of the proposed approach. To specify precisely the basic terms of this common ground we propose using a *communication ontology*, which we discuss in Section 4.2. It describes what exactly communication input/output can be, i.e. what requests will be allowed and what information will be returned.

Considering Question 3, we propose XML-based protocols, so that any application can “understand” them. In relation to Question 4, we propose that the systems share a common user model, possibly through using a user modeling service. In that case the responding system will not need to ask the requesting system what the user already knows. A common (shared) user model is only feasible since all EIS systems within the framework have concept-based representation of their subject domain.

Thus the proposed general architecture includes (see Figure 2):

- Stand-alone, component-based independent EIS using their private subject domain ontologies.
- Information brokerage bureau.
- Services to support systems communication.
- Information channels.

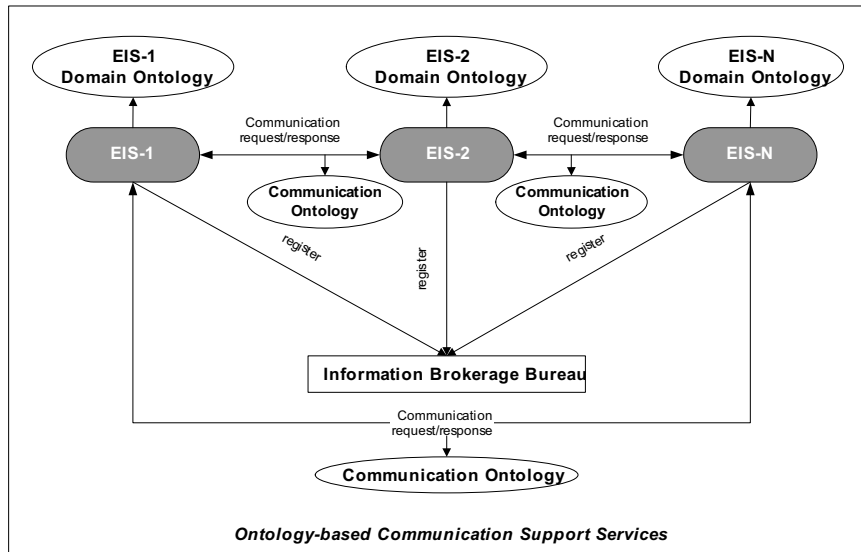


Figure 2. General architecture for component-based EIS interoperability.

The main purpose of the proposed architecture is to support sharing and exchanging information between EIS. This is achieved through communication between the systems (or their components) and services included in the framework to facilitate systems' communication. The services are intended to support different specific aspects of the communication. A significant class of services is ontology-related services since ontology alignment and translation are very important when different applications deal with different ontologies. This is exactly the case here as the different EIS in the framework have different domain ontologies.

A communication is an interaction between two software systems (agents) guided by an interaction protocol. The communication between the systems requires not only standardized transport mechanisms and communication languages, but also common *content languages* (see 4.2) and *semantics*.

Communication between the applications is supported by using *communication ontology* that defines the vocabulary of terms used in the messages at both layers: the message layer and the content layer (see 4.3). To interpret the requests and answers standardized domain ontologies, UM ontologies, as well as upper-level ontologies such as, WordNet, etc. can be used.

The information channels are "bridges" between the systems realizing the actual communication. They support standardized transport mechanisms and a common interaction protocol.

In order to "collaborate" the applications should register within "Information brokerage bureau". So, when a system needs help, it sends a request to this agency and it distributes the request to the other registered systems. In this regard, the applications can be seen as software agents and their communication can be supported by using, for example, the Knowledge Query and Communication Language (KQML) [13].

4.2. Content Language

The popular languages used to represent the content, embedded in messages in ACL (Agent Communication Languages), such as KIF (Knowledge Interchange Format) [14], SL (Semantic Language proposed by FIPA) [15], and Prolog, are 'logic' content languages, which aim at representing knowledge as logic expressions. For our purposes, we consider more attractive 'information' content languages, i.e., languages that set rules to describe a particular type of information elements, since we don't need to represent information as logic expressions. For this reason, we consider XML very appropriate to represent the content embedded in messages in our proposed architecture. The specified XML Schema can then correspond to the ontology of the messages, i.e., content ontology (see 4.3.1).

Since the different EIS (software agents) have to communicate, XML can be embedded within the messages of an ACL, such as KQML, which will be used for agent management.

4.3. Communication Ontology

To define the communication semantics, we propose a communication ontology, which consists of two parts corresponding to the two layers of an interaction, the message layer and the content layer:

- Content ontology - describes the content (knowledge) that can be exchanged by the systems (corresponds to the *content layer*).
- Interaction protocol ontology - specifies interaction communicative act types (corresponds to the *message layer*).

4.3.1 Content Ontology

Content ontology defines the concepts to exchange messages, i.e. gives the meaning of the symbols in the content expression. In our framework, the content ontology consists of two parts: the EIS domain ontologies and a domain-independent ontology describing the concept-based information model of EIS. The latter includes terms such as concept, concept name, relationship type, relationship role, etc.

4.3.2 Interaction Protocol Ontology

Interaction protocol (IP) ontologies describe the input and output data that are processed during protocol's execution together with the actions and the decisions that the agent (application) must perform [16]. A software application that has defined mapping between its internal code and actions and decisions in an ontology would then be able to interpret any IP that is defined with reference to that ontology. The IP ontology defines message types, reasons, and preconditions. While the communication content ontology is generally independent of the framework's functionality, the IP-ontology has to reflect its functionality (e.g. whether it supports agent communication).

Messages represent communicative acts denoting the actions related to communication. In general, communicative acts (performatives) include queries, responses, informational, capability definition, generative, and networking (see KQML [13]).

5 Conclusions

We approach the problems related to systems integration and communication by proposing a service-oriented framework to support efficient communication between component-based EIS. The communication semantics is defined by a communication ontology consisting of content ontology and interaction protocol ontology.

We believe that the proposed framework for supporting communication between applications will eliminate in many cases the need for exporting the entire DM or other application model to another application. Thus, this could be an alternative to interchanging and merging domain models. The advantage of this would be eliminating duplication of stored information, which is unlikely to be often used. In addition, if an application has a specific concept-based application model with no corresponding model in the other system (as is the case with the course model present in AIMS but not in TM4L), import would not work and this would be the only way that the second system (TM4L) could use information from the first one (AIMS). In this way the first system could complement the second one. We believe this would also solve problems with shareability and reusability for already developed applications that don't use standards-based information but rather their own internal representations.

The proposed general architecture can be constrained by considering only two communicating systems that "know" and "trust" each other. We consider this as a common case and will look to find the minimal configuration that will support communication and sharing of knowledge and resources between such systems. We believe that such a *minimalist* architecture will fill the gap between the current technology and realistic situation in the field of web-based educational information systems and the desired future educational semantic web.

Acknowledgements

We would like to thank Geert-Jan Houben for many valuable comments and suggestions on how to improve the paper.

References

1. Brusilovsky, P. (2004). KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In: Proc. of The Thirteenth International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), New York, NY, ACM Press 104-113.
2. De Bra, P., Aerts, A., Berden, B., de Lange, B., Rousseau, B., Santic, T., Smits, D., & Stash, N. (2003). AHA! The Adaptive Hypermedia Architecture. In L. Carr, L. Hardman (Eds) Proceedings of the fourteenth ACM Conference on Hypertext and Hypermedia (81-85). New York: ACM Press.
3. Brusilovsky, P., Eklund, J., and Schwarz, E. (1998). Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems. In Proceedings of 7th International WWW Conference, 30 (1-7), 291-300.
4. Weber, G., Kuhl, H.-C., and Weibelzahl, S. (2001). Developing adaptive Internet-based courses with the authoring system NetCoach. In Proceedings of the Third Workshop on Adaptive Hypermedia, AH'01.
5. Brusilovsky, P., Schwartz, E., Weber, G. (1996). ELM-ART: An Intelligent Tutoring System on the World Wide Web, Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), pp. 261-269.
6. Ritter, S. (1997). PAT Online: A Model-Tracing Tutor on the World-Wide Web, Proceedings of the Workshop on Intelligent Educational Systems on the WWW, pp.11-17.
7. Aroyo, L., & Dicheva, D. (2001). AIMS: Learning and Teaching Support for WWW-based Education. International Journal for Continuing Engineering Education and Life-long Learning, 11(1/2), 152-164.
8. Devedzic, V. (2003). Next-generation Web-based Education. *International Journal for Continuing Engineering Education and Life-long Learning*, Vol. 11, No. 1/2, 232-247.
9. Simon, B., Nejd, W., Miklos, Z., Sintek, M., Salvachua, J. (2003). Elena: A Mediation Infrastructure for Educational Services. In Proceedings of 12th International WWW'03 Conference.
10. Nejd, W., Wolf, B., QuLearning, C. (2002). EDUTELLA: A P2P Networking Infrastructure Based on RDF. In Proceedings of 11th International WWW'02 Conference.
11. Dolog, P., Henze, N., Nejd, W., Sintek, M. (2004). Personalization in Distributed eLearning Environments. In Proceedings of 13th International WWW Conference
12. Dicheva D., Dichev C., Sun, Y., & Nao, S. (2004). A Topic Map Authoring Tool for E-Learning, Proceedings of ADMI' 2004, May 20-22, 2004, Orlando, Florida, 62-67.
13. Finin, T., Fritzson, R., McKay, D., McEntire, R. (1994). KQML as an agent communication language. In Proceedings of 3rd International Conference on Information and Knowledge Management, pp. 456-463
14. KIF (1998). Knowledge Interchange Format. Available at: <http://logic.stanford.edu/kif/dpans.html>
15. FIPA-SL (2004). FIPA SL Content Language Specification. Available at: <http://www.fipa.org/specs/fipa00008/SC00008I.html>
16. Cranefield S., Purvis M., Nowostawski M. and Hwang P. Ontologies for Interaction Protocols <http://www.fipa.org/docs/input/f-in-00076/f-in-00076.pdf>

Individualized Selection of Learning Object

Jian Liu and Jim Greer

Department of Computer Science, University of Saskatchewan
Saskatoon, Saskatchewan, Canada
{jil089, greer}@cs.usask.ca

Abstract: It is becoming more necessary and possible to provide individualized help on selecting learning materials to learners in an online educational system because they usually face more choices. A framework for individualized learning object selection, called Eliminating and Optimized Selection (EOS), is proposed. This framework contains a suggestion on extending learning object metadata specifications and presents an approach to the selecting a short list of suitable learning objects appropriate for the learner and the learning context. The key features of the EOS approach are to evaluate the suitability of a learning object in its situated context and to refine the evaluation by using available historical information about the learner, the content, and the learning context.

1 Introduction

Rapidly evolved internet and web technologies have unlocked tremendous possibilities in the world. The movement towards web-based education is significant one among them. Through the internet, digital educational materials can be delivered by online learning systems effectively and affordably to a learner almost anywhere and at any time. Because of their convenience and flexibility, online learning systems have been increasingly gaining attention from both education providers and consumers.

A world wide effort has been made in developing learning object metadata standards and specifications. The focus of learning object metadata standardization is to improve reusability and interoperability of learning objects. Learning objects that comply with these standards and specifications can be easily discovered, acquired, and reused. This enables the sharing and exchange of learning objects across different learning systems and also provides learners access to multiple learning resources.

As a result of such ubiquitous access, learners in an online virtual course may have more diverse backgrounds than those in a traditional course. The traditional one-for-all approach to content selection becomes inadequate in an online learning environment. Different learners have their distinctive characteristics and learning styles. The resources individuals may have (bandwidth, software, hardware) can also vary. The expected benefit of a learning object and the learning effect gained from it are usually different from learner to learner. Because of the limitation of time and capability, however, it is almost impossible for a learner (or a teacher) to go through all available learning materials to find the most suitable one. Selecting the most suitable learning object among all candidates for individual learners becomes imperative for a learning management system.

In this paper, a framework for individualized learning object selection is proposed. This framework contains a suggestion on extending learning object metadata specifications and presents the Eliminating and Optimized Selection (EOS) approach. In the EOS approach, irrelevant learning objects are eliminated at first. Then the importance of each feature of a learning object is identified by examining the current context, and the associated weight is assigned dynamically. A composite score of all features determines the suitability of the learning object. Finally, the result of the selection is refined by using available historical information.

2 Background

As a result of international efforts on standardization of learning object metadata, several standards and specifications have been released. IEEE Standard for Learning object Metadata (LOM) [2], IMS Metadata Specifications [3], and Canadian Core Learning Resource Metadata Application Profile (CanCore) [1] are well known examples. The core of existing metadata specifications is LOM. Elements in nine categories have been defined in LOM to describe features of a learning object, such as general information, technical requirements, intellectual property rights, and educational characteristics. An instance of the specification can facilitate search, acquisition, and use of learning objects sufficiently, but it cannot provide enough information for individualized learning object selection.

The maturity of the standardization of learning object metadata specification presents a new opportunity and challenge for researchers and developers in the area of intelligent educational systems. Various directions have been explored. For example, McCalla proposed the ecological approach for designing e-learning systems [4]. The key aspects of his approach involve gradually accumulating of information and focusing on end users. Mohan et al. investigated instructional planning processes in e-learning environments and recommended extensions to the current specifications [5]. Applying collaborative filtering and other techniques in web-based educational systems has also been explored [6, 7].

3 A Framework for Individualized Selection of Learning Object

The suitability of a learning object requires a comprehensive evaluation based upon its features. Whether a learning object is suitable depends on its own features and the context where it is used. The suitability of a learning object has various manifestations, such as its appropriateness with respect to the learning goals, its usefulness and helpfulness for learners, pedagogical value, popularity among learners, and endorsement by teachers.

A perfectly suitable learning object for a particular learner should possess the following features:

- It presents the knowledge that the learner wants to learn;
- It can be presented in the learner's environment, i.e. it is affordable in terms of cost and time to the learner, and it can be presented on the learner's platform;
- It is appropriate to the learner's knowledge level, which includes domain knowledge, reading capability, etc.;
- Its presentation style matches learner preferences as much as possible;
- It has high pedagogical value.

Unfortunately, such an ideal learning object can rarely be found in the real world. Usually, a learning object has only some of those desired features. Moreover, some features of a learning object contribute positively to its suitability, while the others contribute negatively. In a more complicated situation, a learning object whose features apparently match a learner's preferences might not be the best choice for the learner on the basis of other similar learners' negative evaluations and/or instructors' negative endorsements — information that can be retrieved from the usage history of learning objects.

3.1 Information Requirements for Individualized Selection

The existing learning object metadata specifications have a defined set of attributes that describe learning objects. The suitability of a learning object for a given learner and learning situation is, however, a contextual feature. It can be decided only when the learning object is situated in a certain context. To determine the suitability of a learning object, some information about the context is necessary in addition to the information about the learning object itself. Besides feature and requirement matching, the suitability of a learning object depends on some features that are more difficult to describe and measure. Historical usage of learning objects can provide valuable help in optimizing selection. As a first approximation, we have taken a pragmatic approach to identifying attributes that may be important in selecting suitable learning objects. We have identified attributes that are relatively easy to obtain, attributes that (based on the educational literature) seem to have maximal discriminatory power, and attributes that link content to context of learning.

3.1.1 Information about Learning Objects

The current existing specifications focus on promoting reusability and interoperability through defining text-based tags for categorizing and annotating learning objects, which facilitate learning object discovery and exchange across different learning objects repositories. To achieve individualized selection, however, extension and modification are required. The following is some examples.

Pedagogical Objective: describes the concept that the learning object presents and what is expected to be achieved. This is a critical attribute for determining the suitability of a learning object. In current existing specifications, pedagogical objectives of learning objects are not addressed, and they might be indirectly inferred from attributes such as *keyword* and *description*. *Description* is difficult to be used for automatic learning object comparison and selection. *Keyword* is not sufficient and sometimes could mislead to unexpected results. An ontology-based representation of pedagogical objectives may serve much better.

Expected Reading Level: indicates the reading capability that the learning object requires the learner to have. In the current existing specifications, the expected reading level is not defined. Instead attributes *context* (the level of education) and *typicalagerange* are used. Learners in the same level of education or in the same age, however, may have different reading ability. Their reading ability actually plays a more important role.

Prerequisite: specifies the knowledge needed by the learning object. The gap between the prerequisite of a learning object and a learner's knowledge level may cause frustration. This attribute is not defined in the existing specifications, but it is a very important factor for deciding the suitability of a learning object for a specific learner.

3.1.2 Information about Context

The suitability of a learning object may change when it is presented in a different context. An excellent learning object can become totally useless in the wrong context. For example, a well designed video clip is not profitable for a learner who doesn't have enough time to download it. A vivid animation of DNA replication won't do any good for a learner who has just seen three other vivid animations of DNA replication. The information about context determines the requirements for current learning.

Learning Objective: indicates what the current learner wants to intent to learn. Learning objects with irrelevant pedagogical objectives are useless.

Resources: define restrictions that may affect the learner's access to learning objects. For example, the *Financial Situation* attribute gives information about the learner's possible financial restrictions, i.e. how much money will the learner be able to pay for access to a particular proprietary resource. If the learner obtains learning materials via an organization, this will refer to how much the organization would spend for this purpose. The *Time* attribute provides information about the time the learner is willing to spend on a learning object. A lengthy learning object is probably not a good choice for a learner who can devote only very limited time.

Learner Characteristics: provides information about the learner. The learner is central to the context. Learner characteristics play a significant role in learning object selection. The information about the learner can be used to decide the degree of the match between learning object features and the learner's preferences. It determines the features of learning objects that have stronger effects on learning in various contexts. More important, sufficient learner information enables applying data clustering and collaborative filtering techniques to gain benefits from others' experience. Theoretically, the more that is known about a learner, the better the selection that can be made for him/her. However, many criteria and constraints may interfere with the selection, and sometimes situational variables add a great deal of complication to the decision. Exactly what attributes should be included is a question yet to be answered.

3.1.3 Information about Learning Object Usage History

Some features relating to quality and appropriateness of a learning object, which may impact its suitability in the given context, may not be readily describable by an author or evaluator. Much useful information can be indirectly gathered from prior experiences with the learning object by learners and instructors. This kind of information should be recorded in the learning object usage history.

Previous Learners: contains models and/or records of learners who have accessed the learning object in the past as well as their actions, evaluation, cognitive state, and achievement related to the learning object.

Previous Instructors: consists teachers who have accessed the learning object and their evaluation or endorsements of the learning object.

Statistics: records accumulated information about the access of the learning object. This can be helpful when more detailed information is not available.

Information about learning object usage history may provide very useful information for optimized selection, and in some situations such information is of the utmost importance. It is valuable to attach historical usage information to every learning object.

Table 1. Information about Context

Attribute Name	Explanation
Learning Objective	The subject or topic the current learner is going to learn
Learner Characteristics	Information about the learner. It is composed by sub-attributes.
Learner Type	Learner's category (e.g. high school student, university student, or non-credit)
Background	Information about related knowledge or experiences of the learner (e.g. major of a university student)
Knowledge in Related Area	Learner's level of domain related knowledge (e.g. experience with programming)
Details of Domain Knowledge	Model of learner's domain specific knowledge (e.g. knowledge about JavaScript and HTML)
Preferred Language	Languages that the learner prefers
Reading Level	Learner's capability of understanding written materials
Listening Level	Learner's capability of understanding vocal materials
Reading Speed	Learner's speed of reading
Preferred Presentation Style	Learner's preference about the way in which the content is presented
Learning Style	Learner's way of learning new concepts or knowledge
Study Attitude	Learner's attitude towards studies
Academic Achievement Goal	The academic goal the learner wants to achieve
General Academic Achievement	Information about the learner's academic performance
History of Using Learning Objects	Learning objects visited by the learner
Resources	Restrictions that may affect the learner's access to learning objects
Computer Environment	Hardware, software, and other related condition
Financial Situation	Financial restriction
Time	Time the learner wishes to spend

Table 2. Information about Learning Objects

Attribute Name	Explanation
Pedagogical Objective	The concept presented in the learning object and what is expected
Environment	The technical requirements needed for presenting the learning object
Cost	The price of the learning object
Language	The language in which the content is presented
Expected Reading Level	The reading capability required by the learning object
Prerequisite	The knowledge needed by the learning object
Typical Learning Time	Time needed for working with the learning object
Presentation Style	The way of presenting the content of the learning object

Table 3. Information about Learning Object Usage History

Attribute Name	Explanation
Previous Learners	Information about previous learners
Accessing Time	The time when the learning object is accessed by the learner
Learner Status	Snap shots of the learner's state before and after accessing the object
Interactions	Actions the learner makes while accessing the learning object
Evaluation	The learner's opinions about the learning object
Achievement	The assessment result of the learner after working with the object
Previous Instructors Statistics	Teachers who have accessed the learning object and their evaluation
Statistics	Accumulated information about the learning object
General Popularity	How often the learning object is selected for all types of learners
Categorized Popularity	How often the learning object is selected for certain type of learners

Table 1, 2, and 3 summarize attributes related to the three areas required by the individualized selection. It is not necessary to get explicit input for every attribute in order to perform the individualized selection. Some of them can be inferred from other attributes, and also sometimes the selection has to be done while some information is lacking. We believe that attributes important to learning object selection in adaptive systems should also become part of learning object metadata, and the results of this research will hopefully influence future work on metadata standards.

3.2 The Eliminating and Optimized Selection (EOS) Approach

According to their roles in learning object selection, attributes of a learning object are categorised into two groups, eliminating attributes and selecting attributes. The learning object becomes unsuitable and is eliminated out of hand if an eliminating attribute cannot match the corresponding requirement of current context; while a selecting attribute helps choose among candidate learning objects by a weighted analysis of the features surrounding the learning object or the context of its use.

The first step of the EOS approach is eliminating irrelevant objects. Eliminating attributes are usually constraints and therefore are binary variables (e.g. 1 or 0). If the feature of a learning object represented by an attribute satisfies the requirement of the current context, it has value 1 (true), and the learning object will be selected to perform further comparison; otherwise, its value is 0 (false), and the learning object is eliminated. Attributes in this category could be pedagogical objective, the language, environment condition (e.g. hardware and software), or the financial cost. Let $a_{eliminate\ i}$ be the value of an eliminating attribute, the evaluation result of eliminating step, $e_{eliminate}$, is

$$e_{eliminate} = \prod_i a_{eliminate\ i} \quad \text{where } a_{eliminate} \in \{0, 1\}$$

If any eliminating attribute does not fit in the current context, the learning object is omitted. The eliminating attributes should be chosen very carefully. When the quantity of available learning objects is limited, some constraints can be relaxed in this step to adjust the selection range.

After the range for learning object selection is reduced, the suitability of all relevant learning objects has to be decided. The contribution of an attribute to the suitability of the learning object depends on its importance in the context and the degree it matches the requirement. If the importance of an attribute i is represented by its weight (w_i) and the degree of the match is indicated by a value between 0 and 1 ($a_{select\ i}$). The result of this step for a learning object (e_{select}) can be reflected by the sum of evaluation of all attributes.

$$e_{select} = \sum_i w_i \times a_{select\ i} \quad \text{where } w_i, a_{select} \in [0, 1]$$

In different contexts a learning object feature affects the suitability in various ways. A very important feature may become a nonentity when the target learner or the environment where the learner resides changes. It is not feasible to define a fixed weight for each feature that applies to all contexts. In the EOS approach, the important features are identified by examining the current context and applying pedagogical principles, and their associated weight is assigned dynamically.

The individualized learning object selection is not simply finding the best match between the features of a learning object and the requirements of the context because in some situations a learning object whose features apparently match a learner's preferences might not be the best choice for the learner. The selection of the most suitable learning object is optimized by using information about previous usage of learning objects, such as experts' evaluation, similar learners' experience, and popularities of learning objects. Influences from these aspects can be negative, and they may also be assigned with different weights to distinguish their importance.

Let $e_{optimize}$ be the result of total optimized adjustment, and e_{final} be the final evaluation result of the learning object, we have:

$$e_{final} = e_{eliminate} \times (e_{select} + e_{optimize})$$

The learning object that has the highest e_{final} value is the most suitable object.

4 Validating the EOS Approach

The implementation of an EOS selector based on the individualized learning object selection approach is under way. First an evaluation of the attributes will be performed in terms of the ease of operationalizing variables and gathering data and in term of the educational relevance. Then we will evaluate the overall system performance by comparing the selection made by the EOS selector with human experts' judgements. The EOS selector and invited experts will perform selection in parallel on the same simulated test bed, which includes a number of created instances of learning object metadata, a number of artificial learners, and simulated usage history of the learning objects. Finally a sensitivity analysis will be performed on the attributes to determine if some can be collapsed or need to be modified.

5 Conclusions

This research aims at exploiting and improving techniques that were developed in Intelligent Tutoring Systems, Recommender Systems, and Semantic Web technologies to enhance web-based educational systems with intelligent functionalities. While currently at a relatively early stage, this research is aimed at improving our ability to select dynamically an appropriate learning object for a given contest. We are investigating the use Multi-Attribute Utility Theory in our selection computations. We are also examining the educational literature to identify additional pedagogical principles that may also be applied to improve the systems' performance. As more and more learning materials come online, the EOS selector should be able to help learners by recommending the most suitable learning objects for their individual learning needs and we hope this type of enhancement may become a core component in the next generation of learning management systems.

Acknowledgements

The Natural Sciences and Engineering Research Council of Canada (NSERC) provided financial support for this research.

References

- 1 CanCore. Canadian Core Learning Resource Metadata Application Profile. Retrieved May 2, 2004, from the World Wide Web: <http://www.cancore.ca/documents.html>
- 2 IEEE 1484.12.1-2002, Draft Standard for Learning Object Metadata. Retrieved May 2, 2004, from the World Wide Web: http://grouper.ieee.org/p1484/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
- 3 IMS Global Learning Consortium, Inc. IMS Learning Resource Meta-data Specification. Retrieved May 2, 2004, from the World Wide Web: <http://www.imsglobal.org/metadata/index.cfm>
- 4 McCalla, G. The Ecological Approach to the Design of E-Learning Environments: Purpose-based Capture and Use of Information about Learners. Submission to Journal of Interactive Media in Education.
- 5 Mohan, P. and Greer, J. E-Learning Specification in the Context of Instructional Planning. In Proceeding of Artificial Intelligence in Education. 2003, 307-314.
- 6 Recker, M., Walker, A., and Lawless, K. What do you recommend? Implementation and analyses of collaborative information filtering of web resources for education. *Instructional Science*, 31(4-5), pp. 299-316.
- 7 Tang, T. Y. and McCalla, G. I. Smart Recommendation for an Evolving E-Learning System. In AIED 2003 Workshop on Technologies for Electronic Documents for Supporting Learning.

Domain Ontologies Integration into the Learning Objects Annotation Process

Zarraonandía, T.¹, Dodero J. M.¹, Díaz P.¹, Sarasa A.²

¹ Universidad Carlos III de Madrid
Laboratorio de Sistemas Interactivos
dei@inf.uc3m.es

² Universidad Complutense de Madrid
asarasa@sip.ucm.es

Abstract. This paper describes an extension of the LOM metadata elements used in the IMS learning object specifications, to incorporate new labels for the semantic classification of non-educational information to the set of LOM metadata. These labels have been obtained from specialized vocabularies represented by means of domain-specific ontologies described using RDF and transformed using a LOM to RDF binding. They are included as LOM standard metadata under the classification subcategory. In order to test its viability, the proposed extension has been implemented as an additional functionality of a learning object editing tool developed as part of the MD2 project.

1 Introduction

In the e-learning context we can distinguish two different areas where the ICTs could be applied: one of them is related with the educational process, while the main focus of the other one is the didactic material development. The convergence of the two areas has been lately revealed by the integration of concepts provided by educational model languages [13] with the packaging of e-learning [5,6] industry contents, which is reflected on IMS Learning Design Specification [7]. This work is focused on the second one of the topics: didactic material development.

Due to the domain-specific characteristics of the educational material, it is necessary to annotate them with domain-specific information, in order to improve their management and exploitation. This paper describes an authoring and annotation tool for learning objects, which has been developed to support the creation of extended didactic material in the framework provided by the MD2 project. Firstly, the context of the project is exposed. Secondly, the problematic of didactic material annotation with metadata extracted from domain-specific ontologies is drawn. Next, an edition and annotation tool and a practical example of LOM extension are presented. Finally, some conclusions are given and future works are described.

2 The Context: Didactic Material Creation

This work is an approach to solve one of the goals of the MD2 project. The aim of the project is to provide solutions to some of the problems related to the generation of learning material, which can be resumed in the following:

- The development of a method for the collaborative generation of learning contents, offering a framework for cooperative knowledge production with a view to improve efficiency and reduce conflicts and coordination issues.
- The extension of current learning object standards to incorporate concepts of instructional hypermedia such as learning links, and to achieve metadata cohesion using accepted and shared concepts (ontologies).

- To elaborate an evaluation framework for a priori testing of the usability and utility of educational products, which includes a method and a number of criteria, parameters and metrics, concerning the educative and interactive quality of applications.

These theoretical endeavors will be practically tested into a platform that will be developed with this purpose. The platform architecture is shown in Fig. 1. Next, a brief description of its modules and their functionalities is presented.

- *Edition + Annotation*: these modules provide basic functionality for edition and annotation of learning objects. Both of them are integrated on a unique authoring tool, similar to Reload¹ or Aloha², but enhanced with some capability extensions to consider traversal aspects served by other modules as ontology import and collaboration support.
- *D-Ontology Import*: this module allows the extension of the label set used for the annotation of learning objects by using ontologies described with RDF(S).
- *Assessment*: This module provides the means to perform a priori tests of the quality of the in-development learning objects [15].
- *Collaboration*: this module supports the collaboration mechanisms during the development of the learning objects, especially during the annotation and evaluation processes. It serves as the base mechanism for the collaborative generation method mentioned among the project goals.
- *Performance Analysis*: it carries out an analysis of the behavior of learning objects' users during the didactic process, in order to evaluate their performance in a given learning context. It takes into account the user model and the run-time engine provided by the LCMS (Learning Content Management System) where objects are executed. The results will revert into annotations to the learning objects regarding the performance of the users.

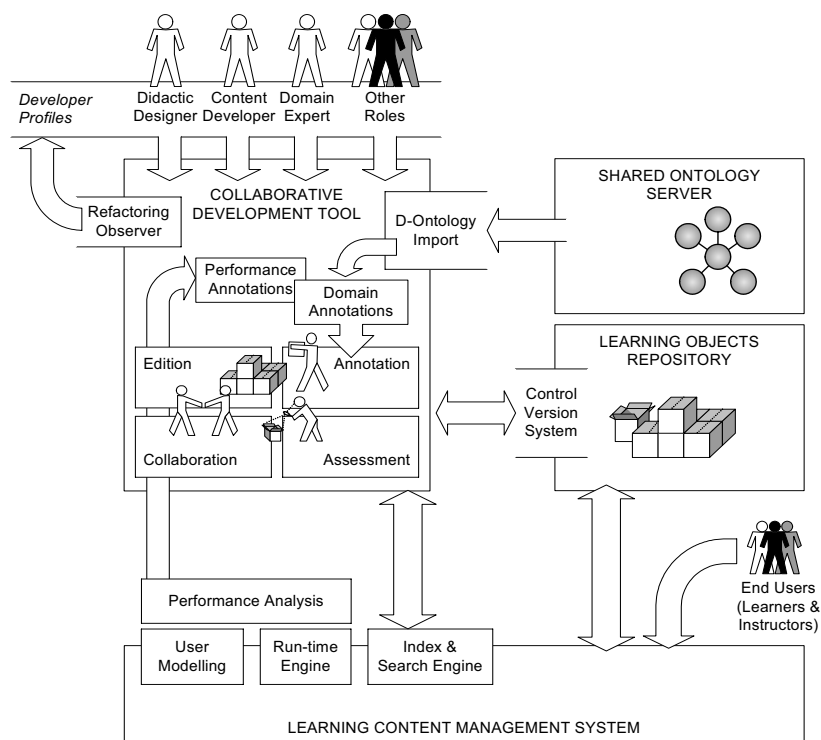


Fig. 1. General architecture of MD2 platform

- *Refactoring Observer*: this is an asynchronous system which, taking as input the values and annotations generated by the previous module, can generate proposals for re-designing the learning objects (i.e., refinement of their objectives and/or requisites, recommendations for

¹ <http://www.reload.ac.uk/>

² <http://aloha.netera.ca/>

new examples, splitting of merging contents, etc.) in order to obtain didactic improvements to adapt them to different learning contexts.

The rest of the components of the figure are external subsystems (LCMS, learning objects repositories and shared ontology servers) that should be adapted to the development platform in order to take advantage of their functionalities. To accomplish this, a Web Service architecture is being developed; a detailed description can be found in [10].

3 Learning Objects Annotation

A major task during the process of creation of a learning object is to generate annotations according to IEEE Learning Object Meta-Data (LOM) standards [6]. These specifications distinguish different metadata categories (i.e. general, technical, educative, etc.) to describe a learning object. Among them, the *classification* category is used to accommodate the annotations related to a particular classification scheme (e.g. the Dewey's decimal classification system [1], or the generalist taxonomies of the Open Directory Project [4]). In our work, the elements *taxon* and *taxonpath* from *classification* category are chosen for cataloguing resources with domain-specific information. It must be noted that this is a limited solution [3], since current LOM specification is designed for the use of simple taxonomies, but not for full-fledged ontologies that can be represented by description logics, as those proposed in OWL [8].

An ontology is as a formal, explicit specification of a shared conceptualization [16]. In this work we focus on the shared character of ontologies. Considering that the purpose of initiatives like the Knowledge Sharing Effort [11] is the development of conventions to support the sharing and reuse of knowledge among systems, it seems reasonable to think in ontologies as an appropriate basis to perform the annotation of learning objects. Despite the fact that currently we can only consider their taxonomic character, in order to not constraint the future evolution of annotations towards full-fledged ontologies, the selected languages have been the ones proposed by the W3C for the development of the Semantic Web: RDF(S) and OWL. On a first stage, it has been considered the importation of domain ontologies expressed in RDF(S), taking into account the strong taxonomic character of the annotations imposed by LOM specification. This has been achieved following the proposed recommendations from the LOM to RDF binding [9].

To increase the reusability and the quality description of a learning object, it is necessary to use more specialized metadata than those proposed by the LOM specifications. These are possibly of non-educational character, in order to adapt the objects to domain-specific, multidisciplinary learning contexts. We take advantage of the LOM infrastructure to introduce these metadata extensions. The development has been carried out on three different stages:

- Development of an authoring tool that provides basic edition functionalities.
- Development of a functionality to obtain metadata classification by using arbitrary hierarchies not directly linked with a specific domain.
- Development of a functionality to integrate hierarchical collections of labels based on domain-specific ontologies.

3.1. Learning Objects Edition & Annotation Tool

The learning objects edition & annotation tool is IMS content packaging [5] and metadata [6] standards compliant. The first one describes the structure of a learning object as a zip content package, composed by a manifest file *imsmanifest.xml* that is divided into four different sections (i.e. *organizations*, *metadata*, *resources* and *sub manifests*) and a set of referenced resource files.

A major objective of the tool was to provide the appropriate labels to classify the domain-specific information contained in the learning object, and for which the LOM specifications does not supply concrete metadata. Two different approaches can be undertaken to overcome this issue:

1. To create specific resources associated to the manifest file, by directly creating the taxonomy. In order to achieve this, an option has been implemented that allows the user to create generic taxonomies as deep and complex as needed. These taxonomies can be managed using visual

and interactive hierarchical components. Then the generated taxonomy can be stored in a different XML document, which will be referred to from the manifest file as another resource file.

2. To import specific domain ontologies: another approach to solve the problem is to extend the label set provided by LOM specification to generate new taxonomies. This is done taking *taxon* and *taxonpath* metadata elements as starting point. The objective of these metadata is to catalog the learning object information using specific classification systems. Based on them, LOM offers the possibility to introduce new labels relative to other namespaces, which share the same classification objectives. Our tool has taken advantage of this LOM extension facility to import specific domain ontologies by importing an RDFS description of the domain ontology and transforming it into LOM metadata [9]. The imported labels are internally stored in order to give a personalized label set for domain-specific classification, which can be reused in future annotations. The reverse process is also possible if the user opens a learning object for editing, which contains *taxon* metadata descriptions related to the specific domain ontology. In order to not constraint the semantic interoperability of the object in such situations, the namespaces used on the domain ontology have been maintained.

3.2. Practical Example of LOM Extension

A concrete example is provided, which consists in the annotation of a learning object about music specialties. D-Ontology Import module support will be used to obtain the required labels to annotate resources according to a domain-specific taxonomy, which is represented separately in RDF (underneath the xml namespace http://www.mysite.com/dmoz_music) and stored on the Shared Ontology Server. The reference taxonomy was built inspired from *dmoz.org* Open Directory project categories.

According to [9] an *rdfs:subPropertyOf lom-cls:classification* should be used to perform the adequate annotation in the manifest file, pointing always to *taxon* values in a taxonomy hierarchy which should be an instance of *lom-cls:Taxonomy*, as the example shown at **Fig. 2**. There are three different possibilities depending on the value of *Purpose* subcategory from the *Classification* category:

1. When the value is “Discipline” or “Idea”, use *dc:subject*, which is an *rdfs:subPropertyOf lom-cls:classification*.
2. If the value is part of the LOM restricted vocabulary for the *Purpose* subcategory, use one of the following, which are *rdfs:subPropertyOf lom-cls:classification* *lom-cls:prerequisite*, *lom-cls:educationalObjective*, *lom-cls:accessibilityRestrictions*, *lom-cls:educationalLevel*, *lom-cls:skillLevel*, *lom-cls:securityLevel* or *lom-cls:competency*.
3. Define our own *rdfs:subPropertyOf lom-cls:classification* to use for local purposes.

From these annotation approaches the third one is the less adequate as it could compromise the semantic interoperability of the object, since third party systems are obliged to know the defined set of labels in order to understand domain annotations. If we compare the first and second approaches, the former is more suitable as it is less specific and more easily adaptable to existing classification systems. Taking into account those issues, a common annotation will look like:

```
<dc:subject rdf:resource="http://www.mysite.com/dmoz_music#blues"/>
```

```

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:lom-cls="http://ltsc.ieee.org/2002/09/lom-classification#">
  <lom-cls:Taxonomy rdf:ID="DMOZ">
    <rdfs:label>Music Specialties</rdfs:label>
    <lom-cls:taxon>
      <dcterms:DMOZ rdf:ID="blues">
        <rdf:value>blues</rdf:value>
        <rdfs:label>Blues</rdfs:label>
      </dcterms:DMOZ>
      <dcterms:DMOZ rdf:ID="flam">
        <rdf:value>flam</rdf:value>
        <rdfs:label>Flamenco</rdfs:label>
      </dcterms:DMOZ>
      <dcterms:DMOZ rdf:ID="jazz">
        <rdf:value>jazz</rdf:value>
        <rdfs:label>Jazz</rdfs:label>
        <lom-cls:taxon>
          <dcterms:DMOZ rdf:ID="jazz-fus">
            <rdf:value>jazz-fus</rdf:value>
            <rdfs:label>Jazz Fusion</rdfs:label>
          </dcterms:DMOZ>
        </lom-cls:taxon>
      </dcterms:DMOZ>
      <dcterms:DMOZ rdf:ID="eth">
        <rdf:value>eth</rdf:value>
        <rdfs:label>Regional & Ethnic</rdfs:label>
        <lom-cls:taxon>
          <dcterms:DMOZ rdf:ID="eth-lat">
            <rdf:value>eth-lat</rdf:value>
            <rdfs:label>Latin</rdfs:label>
          </dcterms:DMOZ>
        </lom-cls:taxon>
      </dcterms:DMOZ>
      <dcterms:DMOZ rdf:ID="rnb">
        <rdf:value>rnb</rdf:value>
        <rdfs:label>Ryhtm & Blues, Funk, and Soul</rdfs:label>
      </dcterms:DMOZ>
    </lom-cls:taxon>
  </lom-cls:Taxonomy>
</rdf:RDF>

```

Fig. 2. A simple RDF taxonomy for learning resources about music specialties.

4 Conclusions and Future Work

We have described our experience during the development of a tool for edition and annotation of learning objects using domain-specific ontologies. We found that LOM metadata elements for the generation of specific domain taxonomies are rather limited. This inconvenient can be overcome by extending LOM. Three possible forms of annotations have been considered to perform this extension. After their evaluation, we suggest using a *dc:subject* tag as we found is the less constraining with the semantic interoperability of the learning objects. Other possibility of LOM extension, which partially solves the taxonomic classification problem, is to use ontologies to perform the annotation. But since ontologies offer a wider spectrum of knowledge representation possibilities, more complex issues like the integration of description logics into LOM-based annotations are still open.

Upcoming research work is in the context of MD2 project and will involve the study of the extension of the LOM specifications to provide a more complete support to OWL ontologies that can be defined in any of its three sublanguages (*Lite*, *DL* and *Full*) [8].

Apart from that, we found that little attention has been paid to the developers of educational material and the multidisciplinary nature of their tasks (e.g., teaching, didactic advice, usability, or any other

specialized field from the knowledge domain). From that point of view we consider the educational material development as a collaborative and multidisciplinary process [12]. Augmenting the edition tool capabilities to integrate support for the collaborative development of learning objects will conform another line of future research. A learning object quality model is being also developed and integrated within the tool, to allow the definition of appropriate metrics to evaluate and annotate the learning objects during their development process.

Acknowledgements

This work is part of the MD2 project (TIC2003-03654), funded by the Spanish Ministry of Science and Technology.

References

1. M. Dewey, *The Man and the Classification*, Gordon Stevenson & Judith Kramer-Greene (eds.), 1983.
2. J. M. Dodero, M. A. Sicilia and E. Garcia, "A Knowledge Production Protocol for Cooperative Development of Learning Objects", In *Proceedings of the 2nd Workshop on Agent-Supported Cooperative Work, International Conference on Autonomous Agents*, Montreal, 2001.
3. H. B. Enderton, *A Mathematical Introduction to Logic*, Academia Press, 1972.
4. A. Gerhart, "Open Directory Project Search Results and ODP Status", *Search Engine Optimization*, April 5, 2002.
5. IMS Global Learning Consortium, "IMS Content Packaging Information Model", Versión 1.1.3 Final Specification, 2003.
6. IMS Global Learning Consortium, "IMS Learning Resource Meta-data Information Model", Versión 1.2 Final Specification, 2001.
7. IMS Global Learning Consortium, "IMS Learning Design Best Practice and Implementation Guide", Versión 1.0 Final Specification, 2003.
8. D. L. McGuinness and F. Van Harmelen (eds.) "OWL Web Ontology Language Overview", W3C Recommendation, February 2004.
9. M. Nilsson, M. Palmer, and J. Brase, "The LOM RDF binding – principles and implementation", *3rd. Annual Ariadne Conference*, Katholieke Universiteit leuven, Belgium, November 2003.
10. C. L. Padrón, J. Torres, J. M. Dodero, P. Díaz, I. Aedo, "Learning Web Services Composition and Learner Communities Support for the Deployment of Complex Learning Processes", In *Proceedings of the 4th Int. Conf. on Advanced Learning Technologies (ICALT)*, Joensuu, Finland, August 2004.
11. R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. Mckay, T. Finin, T. Gruber, and R. Neches, "The DARPA Knowledge Sharing Effort: Progress Report", *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pp. 777-788, 1992.
12. P. Polsani, "Use and Abuse or Reusable Learning Objects", *Journal of Digital Information*, 3(4), Article No. 164, 2003.
13. A. Rawlings, P. van Rosmalen, E. J. R. Koper, M. Rodríguez-Artacho, and P. Lefrere, *Survey of Educational Modelling Languages*, CEN/ISSS WS/LT Learning Technologies Workshop, Brussels, 2002.
14. D. Rehak and R. Mason, "Keeping the Learning in Learning Objects", In A. Littlejohn (ed.), *Reusing Online Resources: A Sustainable Approach to eLearning*, Kogan Page, London, 2003.
15. A. Sarasa, and J.M. Dodero, "Towards a Model of Quality for Learning Objects", In *Proceedings of the 4th Int. Conf. on Advanced Learning Technologies (ICALT)*, Joensuu, Finland, August 2004.
16. R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and Methods", In *Data & Knowledge Engineering*, 25(1-2), pp. 161-197.

The Use of Ontologies in ITS Domain Knowledge Authoring

Pramuditha Suraweera, Antonija Mitrovic, Brent Martin

Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{psu16,tanja,brent}@cosc.canterbury.ac.nz

Abstract. Acquiring the domain knowledge is a task that requires a major portion of the time and effort when building an ITS. Researchers have been exploring ways of automating the knowledge acquisition process since the inception of ITSs with limited success. All past research attempts have focussed on acquiring knowledge for procedural domains. Our goal is to develop an authoring system that acquires knowledge for procedural as well as nonprocedural domains. We propose a four phase approach: composing an ontology of the domain, extracting syntax constraint from it, learning semantic constraints from the examples provided by the domain expert and finally verifying the generated constraints. This paper presents an overview of the knowledge acquisition system for acquiring knowledge for constraint-based tutors. It mainly focuses on composing the ontology and acquiring syntax constraints from it. Further work on this project will focus on learning from examples and validating the generated constraints.

1 Introduction

Acquiring domain knowledge is a major hurdle in building Intelligent Tutoring Systems (ITS) [1]. Although there have been several attempts to ease the burden on ITS developers by automating the process, they have met with limited success. All previous attempts have focussed on acquiring knowledge required for teaching procedural tasks. Our goal is to drastically reduce the time and effort required for acquiring domain knowledge by automating knowledge acquisition for intelligent tutors for both procedural and nonprocedural domains.

Constraint based modelling (CBM) [2] is a student modelling approach that somewhat eases the knowledge acquisition bottleneck by using a more abstract representation of the domain compared to other commonly used approaches [3]. CBM is based on Ohlsson's theory of learning from performance errors [4]. It focuses on correct knowledge rather than describing the student exactly as with model tracing. However, building a complete constraint base still remains a major challenge. Mitrovic reported that she took just over an hour to produce a constraint for SQL-Tutor [5, 13], which currently contains more than 650 constraints. Therefore, the task of composing the knowledge base of SQL-Tutor would have taken over 4 months to complete. Our goal is to dramatically reduce the time and effort required for composing the knowledge base required for constraint-based tutors by automating the knowledge acquisition process.

We envisage ontologies to play a central role in the whole knowledge acquisition process. A preliminary study conducted to evaluate the role of ontologies in manually composing a constraint base showed that constructing a domain ontology indeed assisted the composition of constraints [6]. The study showed that ontologies can be used to organise the constraint base into meaningful categories. This enabled the author to visualise the constraint set and to reflect on the domain assisting them to create more complete constraint bases.

The remainder of the paper is organised into five sections. The next section presents a brief description of automatic knowledge acquisition systems. Section 3 gives an overview of our project. Details on developing the ontology are given in Section 4. Section 5 discusses the process of acquiring syntax constraints from the ontology. Conclusions and future work are presented in the final section.

2 Related Work

Past research on acquiring knowledge for ITSs have solely focused on acquiring knowledge for teaching procedural tasks such as tasks in simulated environments and solving mathematical algebraic problems. The knowledge acquisition systems that acquire domain knowledge as a runnable model for evaluating student solutions include KnoMic [7], Disciple [8, 9] and Demonstr8 [10]. All these systems acquire knowledge by observing the domain expert performing a task and generalising it to be applicable for other problems.

KnoMic is a learning-by-observation system for acquiring procedural knowledge in a simulated environment. The system observes and records the procedure taken by the domain expert in performing a task within the simulated environment. While performing the task the expert has to annotate the points where he/she had changed goals because it was either achieved or abandoned. The resulting set of observation traces are generalised by the system to learn the conditions of actions, goals and operators. During an evaluation to test the accuracy of the procedural knowledge learnt in an air combat simulator, KnoMic acquired 140 productions. Out of the total 140 created, 101 were fully correct and 29 of the remainder were functionally correct [7]. Although the results are encouraging KnoMic's applicability is limited to only simulated environments.

Disciple is a shell for developing personal agents. It relies on a semantic network of the domain that describes the domain, which can be either composed by the author or imported from a repository. Initially the shell has to be customised to the domain by building a domain-specific interface, which gives a natural way of solving problems for the domain expert. Disciple also requires a problem solver for the domain. The domain expert has to initiate the knowledge elicitation process by providing problem-solving examples. The agent generalises the provided example using a generalisation algorithm with the assistance of the domain expert. The generalised example is refined by requesting the expert to validate the examples generated by the system. As Disciple depends on problem solving instances provided by the domain expert, they should be carefully selected to reflect significant problem states. The task of selecting significant problem states requires expertise in knowledge engineering which is scarce. Furthermore, building a problem solver for some domains is extremely difficult, if not impossible.

Demonstr8 is an authoring tool for building model-tracing tutors for arithmetic. It relies on the domain expert to specify all the algebraic functions that can be used and their outcomes in the form of a table. It uses programming by demonstration to reduce the authoring effort. The system provides a drawing tool like interface for building the student interface of the ITS. The system automatically defines each GUI element as a working memory element (WME), while WMEs involving more than a single GUI element must be defined manually. The system generates production rules by observing problems being solved by an expert. Demonstr8 performs an exhaustive search in order to determine the problem-solving procedure used to obtain the solution. If more than one such procedure exists, then the user would have to select the correct one.

3 Automatic Constraint Acquisition

Existing approaches to knowledge acquisition for ITSs acquire procedural knowledge by recording the domain expert's actions and generalising recorded traces using machine learning algorithms. Even though these systems are well suited to simulated environments where goals are achieved by performing a set of steps in a specific order, they fail to acquire knowledge for non-procedural domains. Our goal is to develop an authoring system that can acquire procedural as well as declarative knowledge.

The authoring system will be an extension of WETAS [11], a web-based tutoring shell that facilitates building constraint-based tutors. WETAS provides all the domain-independent components for a text-based ITS, including the user interface, pedagogical module and student modeller. The pedagogical module makes decisions based on the student model regarding problem/feedback generation and the student modeller evaluates student solutions by comparing them to the domain model and updates the student model. The main limitation of WETAS is its lack of support for authoring the domain model.

The domain model for CBM tutors [14, 15] consists of a set of constraints, which are used to identify errors in student solutions. As the space of false knowledge is much greater than correct knowledge, in CBM knowledge is modelled by a set of constraints that identify the set of correct solutions from the set of all possible student inputs. CBM represents knowledge as a set of ordered pairs of relevance and satisfaction conditions. The relevance condition identifies the states in which the constraint is relevant, while the satisfaction condition identifies the subset of the relevant states in which the constraint is satisfied.

As WETAS does not provide any assistance for developing the knowledge base, typically a knowledge base is composed using a text editor. Although the flexibility of a text editor may be adequate for knowledge engineers, novices tend to be overwhelmed by the task. Our goal is to reduce the time and effort required for building a constraint base by adding support for automatic constraint acquisition to WETAS. We propose a four-stage process initiated by modelling the domain as an ontology. The ontology would be composed by a domain expert using the ontology modelling tool. Once the ontology is completed, the system would analyse the ontology and extract syntax constraints directly from the completed ontology. During the third phase, the system would acquire constraints by analysing sample solutions provided by the expert. Finally the constraint set is validated with the assistance of the domain expert, where the expert would label the system generated examples as correct or incorrect.

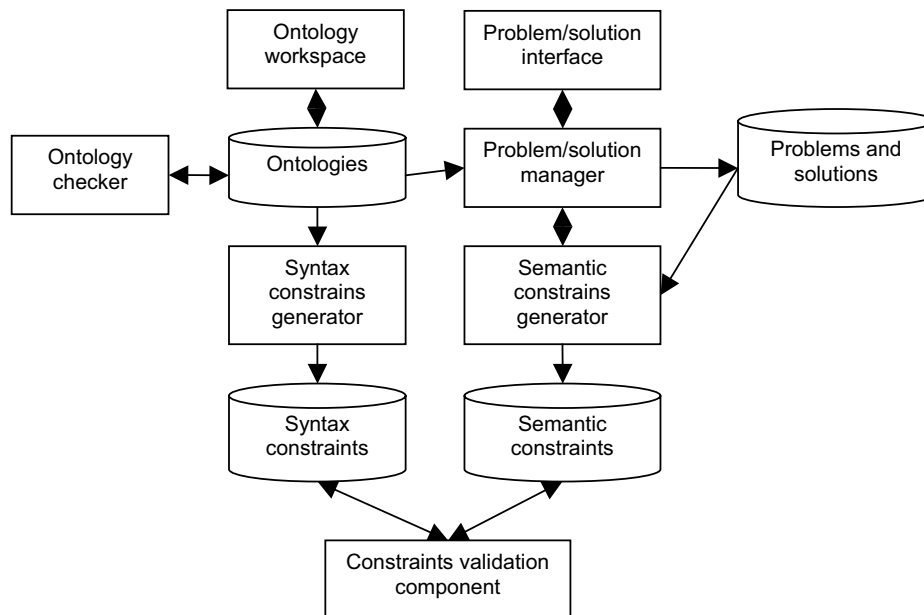


Fig. 1. Architecture of the constraint-acquisition system

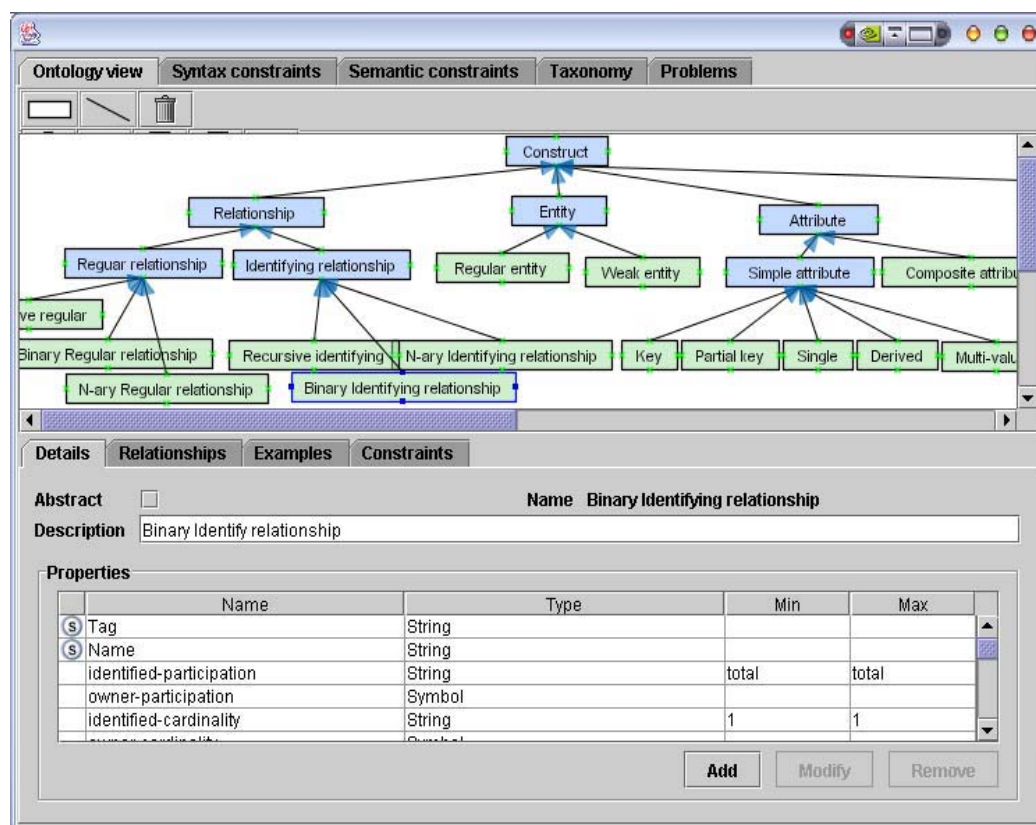
The architecture of the constraint acquisition system consists of an ontology workspace, ontology checker, problem/solution manager and syntax and semantic constraint generators, as depicted in Figure 1. During the initial phase, the domain expert models an ontology of the domain in the ontology workspace. The ontology checker validates the ontology during the ontology composition state. The completed ontology is stored in the ontology repository. The syntax constraints generator analyses the completed ontology and generates syntax constraints from it. The generated syntax constraints are stored in the syntax constraints repository. The generation of constraints from a domain ontology is discussed further in Section 5.

The domain expert has to specify the representation for solutions prior to entering problems and sample solutions. The solution representation is a decomposition of the solution into components consisting of a list of instances of concepts. For example, a sentence in English consists of a list of words and a list of punctuation marks.

The domain expert has to enter sample problems and their solutions during the third phase of knowledge acquisition. The problems/solution interface assists the user by providing a dynamic form that consists of input boxes for populating each property of the concept instance. The expert is requested to provide different solutions that depict different ways of solving the same problem. While the expert enters in an alternative correct solution, the system attempts to match each component of the solution to components of the initial solution. These matches are later used to compose a set of semantic constraints that compare the student's solution against the system's ideal solution. The expert is also encouraged to supply solutions containing typical errors made by students. The system would use these erroneous solutions to provide more detailed assistance. The system also verifies the solutions provided by the expert using the generated syntax constraints. If a discrepancy is identified, the user is alerted and the solution may be modified to comply with the ontology or vice versa.

The final phase involves ensuring the validity of constraints. During this phase the system would generate examples for the domain to be validated by the author. In situations where the author's validation conflicts with the system's evaluation according to the domain model, the system would request the author to provide further examples to illustrate the rationale behind the conflict. The system would use the new examples to resolve the conflicts and may also generate new constraints. The author may also wish to examine the English description of the generated constraints and dispute them by providing counter examples.

We started developing the constraint acquisition system in 2003. The ontology workspace, ontology checker, problem/solution interface and the syntax constraint generator are completed. We are currently working on the semantic constraints generator.



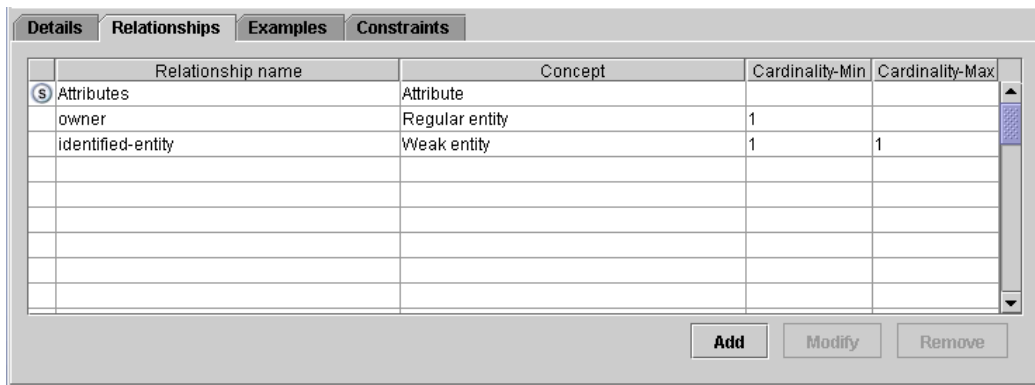


Fig. 2. Ontology workspace interface

4 Developing the ontology

As discussed earlier, the first phase in authoring a constraint base is developing the domain ontology. The ontology will be later used to generate constraints automatically. An ontology describes the domain, by identifying all the important domain concepts and various relationships between them. The ontology workspace provides an environment for composing the domain ontology in terms of concepts and their sub concepts as shown in Figure 2. All concepts are represented using rectangles and they are related to their sub concepts using arrows. The interface has no restrictions in placing concepts within the workspace. The user can position the concepts to display a hierarchical structure. The completed ontology is saved on a central server in the XML format.

The ontology displayed in Figure 2 represents the concepts of ER modelling, a popular database modelling technique. The ER model describes data as entities, attributes and relationships. An entity is the basic object represented in the ER model, which is a 'thing' in the real world with an independent existence. Each entity has particular properties, called *attributes*, that describe it. A relationship is an association between two or more entities.

The ER ontology depicted in Figure 2 contains *Construct* as the most general concept. *Relationship*, *Entity*, *Attribute* are sub-concepts of *Construct*. *Relationship* is specialised into *Regular* and *Identifying*, which are the two types of relationships and *Entity* is specialised, according to its types, as *Regular* and *Weak*. Subclasses of *Attribute* are *Simple* or *Composite* attributes and *Simple* attributes are further specialised into five categories: *Key*, *Partial key*, *Single*, *Derived* and *Multi-valued*.

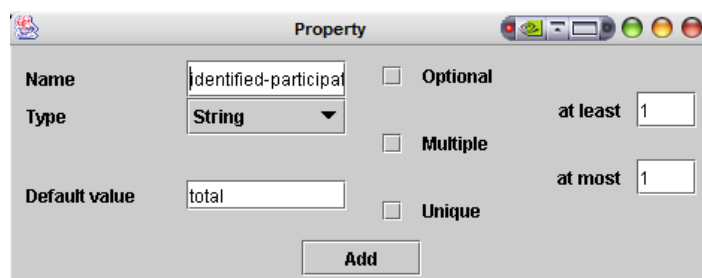


Fig. 3. Details of *identified-participation* property

Each concept has a set of properties that describes it. To define properties for a concept, the author uses the property addition interface shown in Figure 3. The range of values that the property may hold can be specified in terms of minimum and maximum values or as a set of distinct values. Other restrictions include specifying that the value of a property is unique, optional or can contain multiple values. Figure 3 depicts the *identified participation* property of the *Binary identifying relationship* concept. The property has a default value of 'total'. Furthermore as the 'at least' and 'at most' fields are both set to 1 the *identified participation* property has to have a single value.

The remainder of the properties of *Binary identifying relationship* concept, as shown in Figure 2, include *name*, *owner participation* and *identified cardinality*. Most properties are of type ‘string’ except *owner participation* and *owner cardinality* are of type ‘symbol’. Both *identified cardinality* property and *identified participation* property have default values: 1 and ‘total’ respectively.

The relationships that are involved with *Binary identifying relationship* concept are detailed in Figure 2. They include *attributes*, *owner* and *identified entity*. The *attributes* relationship is a relationship between *Binary identifying relationship* and *Attribute* with no restrictions on the cardinality. The *owner* relationship with *Regular entity* has a minimum cardinality of 1. The *identified entity* relationship, as detailed in Figure 4, between *Binary identifying relationship* and *Weak entity* has a minimum and maximum cardinality of 1.

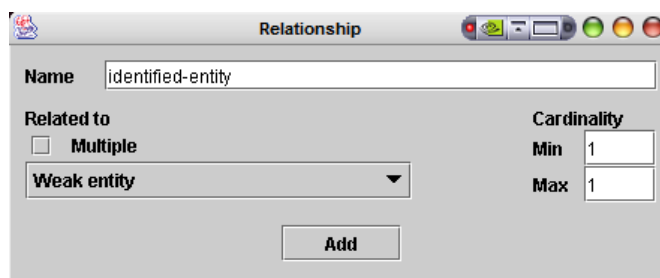
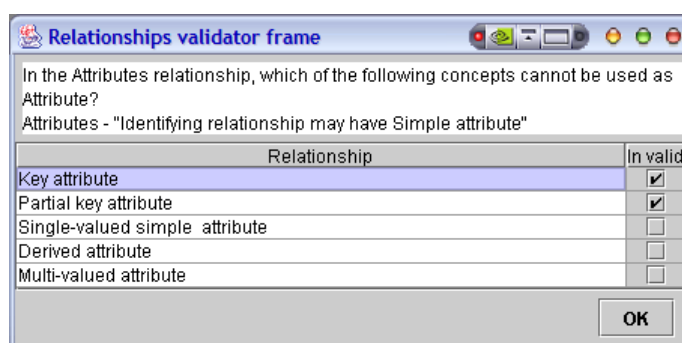


Fig. 4. Details of *identified-entity* relationship

During the task of composing an ontology, the domain expert may add relationships that are too general. Since constraints are composed directly from the relationships found in the ontology, it is imperative that the relationships are valid. In order to ensure that all added relationships are completely accurate, the system engages the author in a dialog. During this dialog the author is presented with lists of specialisations of concepts involved in the relationship and is asked to label the specialisations that violate the principles of the domain. As an example, consider the relationship between *Binary identifying relationship* and *Attribute*. As shown in Figure 5, the initial question posed asks whether each of the specialisations of *attribute* (*key*, *partial key*, *single-valued* etc) are applicable to the *attributes* relationship. The user would indicate that *key* or *partial key* attributes cannot be used in the *attributes* relationship. The system replaces the original relationship with a more specific one at the completion of the dialog.

Fig. 5. Relationship validate dialog for ‘Entity has attribute’ relationship



The ontology is represented in memory as a list of objects that keeps track of all the details about the concepts. The concept’s properties and relationships are contained as lists within each concept object. The concept object keeps a record of its super concepts and sub concepts. The generated constraints that belong to each concept are also stored in a list with the concept object.

The internal representation of the ontology gets converted to XML for storing in the central server. The XML representation uses a set of XML tags defined specifically for this project. Details of each concept along with a unique id that identifies the concept are enlisted using the appropriate XML tags.

The concept ids are used to specify the relationships between concepts. The position and sizes of the graphical representations for the concepts are also recorded in XML in order to recreate an identical ontology in the ontology workspace. During the restoration of the ontology new objects for representing concepts are created with the relevant information extracted from the XML representation.

Although the ontology is stored using proprietary XML tags, it can be easily be transformed to a standard ontology representation form such as DAML [16]. The system was developed to save ontologies using its own XML representation in order to speed up the progress of the research project, avoiding the need for extensive research into DAML implementation.

5 Acquiring Syntax Constraints from the Domain Ontology

An ontology contains a lot of information about the domain and is much easier to create than the final domain model. Our goal is to extract the useful syntactic information from an ontology to generate syntax constraints for the domain model. This process involves analysing the relationships between concepts and the properties of concepts that exist in the ontology.

Initially the constraint generator extracts all the relationships between concepts. Each relationship with restrictions on the cardinality such as a minimum or maximum yields a syntax constraint that restricts the instances participating in the particular relationship. As an example, consider the *identified-entity* relationship found in Figure 4. It generates a constraint which says *Binary identifying relationship* must have exactly 1 *Weak entity* as the *identified entity*. The relevance condition of the constraint focuses on identifying instances of *Binary identifying relationships*, whereas the satisfaction condition specifies that each of them has to have exactly one weak entity as the *identified-entity*.

The domain and range of each concept's properties are also analysed for generating constraints. The constraint generator creates a constraint for each restriction on the domain and range of a property. Such restrictions involve minimum and maximum values allowed, whether the property is required, multivalued or unique. The generated constraints are similar to the constraints generated from relationships, having a check for identifying each property as a relevance condition and a satisfaction condition that ensures that the specified condition is met.

For example, when the processing of the *Binary identifying relationship* concept illustrated in Figure 2, 4 the system generates six constraints:

- *Binary identifying relationship* must have at least 1 *Regular entity* as the owner
- *Binary identifying relationship* must have exactly 1 *Weak entity* as the identified entity
- The *identified participation* property of *Binary identifying relationship* must be total
- The *identified cardinality* property of *Binary identifying relationship* must be 1
- The *name* property of *Relationship* type has to be unique
- *Relationship* type must have exactly 1 name

The dialog sessions for validating relationships during the ontology composing phase also contribute towards generating syntax constraints. The specialisations of concepts involved in the relationship that violate the principles of the domain are used to generate constraints. They ensure that elements of a solution does not participate in such relationships that violate the domain principles. The specialisations marked as violations by the author in Figure 5 would be used to generate two constraints: *Binary identifying relationship* cannot have a *key* attribute and *Binary identifying relationship* cannot have a *partial key* attribute.

The syntax constraints generator produced a total of 48 syntax constraints from the ER ontology depicted in Figure 2. The generated set of constraints covered all syntax constraints that existed in KERMIT [12], a constraint-based tutor for ER modelling. Although the initial results are derived from only a single domain, we believe that the system would be able to successfully handle most non-procedural domains. The ontology workspace would be enhanced to handle procedural domains by adding further constructs.

6 Conclusions and Future Work

We provided a brief overview of our main research objective: automatically acquire domain knowledge required for constraint-based tutors. We propose a four phase process, initiated by modelling a domain ontology. The system then analyses the completed ontology and extracts syntax constraints from it. During the third phase, the author provides example problems and their solutions and the system generates semantic constraints by analysing the solutions. Finally, the induced constraint set is validated with the assistance of the author.

The paper included a detailed description of the first two phases: modelling the ontology and extracting constraints from it. The initial tests conducted on acquiring constraints from an ontology composed for ER modelling produced encouraging results. The system generated the complete set of syntax constraints found in KERMIT, a constraint based tutor developed for the same domain.

Currently we are working on acquiring semantic constraints from examples provided by the domain expert. We will be exploring machine learning algorithms such as learning from examples and learning from analogy for automatically acquiring semantic constraints. The ontology workspace will also be enhanced to handle procedural domains.

Finally the system will be thoroughly evaluated to test its effectiveness. Most importantly, the quality and the correctness of the knowledge base generated by the system have to be evaluated. Since this research aims to produce a system that is capable of acquiring knowledge for a vast range of tasks, it will be tested in different domains. The usability of the system will also be tested.

Acknowledgements

The work reported here has been supported by the University of Canterbury Grant U6532.

References

1. Murray, T.: Expanding the knowledge acquisition bottleneck for intelligent tutoring systems. *Int. J. Artificial Intelligence in Education* 8 (1997) 222–232
2. Ohlsson, S.: Constraint-based student modelling. In: *Student Modelling: the Key to Individualized Knowledge-based Instruction*, Berlin, Springer-Verlag (1994) 167– 189
3. Mitrovic, A., Koedinger, K., Martin, B.: A comparative analysis of cognitive tutoring and constraint-based modelling. In Brusilovsky, P., Corbett, A., Rosis, F.d., eds.: *UM2003*, Pittsburgh, USA, Springer-Verlag (2003) 313–322
4. Ohlsson, S.: Learning from performance errors. *Psychological Review* 103 (1996) 241–262
5. Mitrovic, A.: Experiences in implementing constraint-based modelling in SQL-tutor. In Goettl, B.P., Half, H.M., Redfield, C.L., Shute, V.J., eds.: *4th International Conference on Intelligent Tutoring Systems, ITS 98*, San Antonio (1998) 414–423
6. Suraweera, P., Mitrovic, A., Martin, B.: The role of domain ontology in knowledge acquisition for ITSs. In: *7th International Conference on Intelligent Tutoring Systems, ITS 2004*. (2004) to appear
7. van Lent, M., Laird, J.E.: Learning procedural knowledge through observation. In: *International conference on Knowledge capture*, Victoria, British Columbia, Canada, ACM Press (2001) 179–186
8. Tecuci, G.: *Building Intelligent Agents: An Apprenticeship Multi-strategy Learning Theory, Methodology, Tool and Case Studies*. Academic press (1998) 9. Tecuci, G., Keeling, H.: Developing an intelligent educational agent with disciple. *International Journal of Artificial Intelligence in Education* 10 (1999) 221–237
10. Blessing, S.B.: A programming by demonstration authoring tool for model-tracing tutors. *Int. J. Artificial Intelligence in Education* 8 (1997) 233–261
11. Martin, B., Mitrovic, A.: Domain modelling: Art or science? In U. Hoppe, F.V..J.K., ed.: *Artificial Intelligence in Education, AIED 2003*, IOS Press (2003) 183–190
12. Suraweera, P., Mitrovic, A.: Kermit: a constraint-based tutor for database modelling. In Cerri, S., Gouarderes, G., Paraguacu, F., eds.: *6th International Conference on Intelligent Tutoring Systems, ITS 2002*, Biarritz, France (2002) 377–387

13. Mitrovic, A. An Intelligent SQL Tutor on the Web. *Int. J. Artificial Intelligence in Education*, v13no2-4, 2003: 173-197.
14. Mitrovic, A., Mayo, M., Suraweera, P and Martin, B. Constraint-based tutors: a success story. *Proc. 14th Int. Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-2001*, Budapest, June 2001, L. Monostori, J. Vancza and M. Ali (eds), Springer-Verlag Berlin Heidelberg LNAI 2070, 2001: 931-940.
15. Mitrovic, A. Supporting Self-Explanation in a Data Normalization Tutor. In: V. Alevan, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo, K. Yacef (eds) *Supplementary proceedings, Artificial Intelligence in Education, AIED 2003*, pp. 565-577, 2003
16. DARPA Agent Markup Language, <http://www.daml.org/>