A model of multitutor ontology-based learning environments

Antonija Mitrovic*

Intelligent Computer Tutoring Group,
Department of Computer Science,
University of Canterbury,
Private Bag 4800, Christchurch, New Zealand
E-mail: tanja@cosc.canterbury.ac.nz
*Corresponding author

Vladan Devedzic

Department of Information Systems and Technologies,
FON – School of Business Administration,
University of Belgrade, POB 52, Jove Ilica 154,
Belgrade, Serbia and Montenegro
E-mail: devedzic@galeb.etf.bg.ac.yu

Abstract: The paper proposes the M-OBLIGE model for building multitutor ontology-based learning environments. The model is based on local ontologies, describing the domain of each individual tutor in the environment, and external ontologies, describing more general concepts. The ontologies are used by ontology processors to decide which tutors might benefit a student who needs to learn new concepts. The model allows domain expertise to be shared, and can be used as a framework for integrating multiple tutors on the web. We show how the model can be applied to tutors in the database domain. We also illustrate the process of developing the ontologies for an existing system.

Keywords: ontologies; interoperability of web-enabled intelligent tutoring systems.

Reference to this paper should be made as follows: Mitrovic, A. and Devedzic, V. (xxxx) 'A model of multitutor ontology-based learning environments', *Int. J. Continuing Engineering Education and Lifelong Learning*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Antonija Mitrovic is a Senior Lecturer at the Computer Science Department, University of Canterbury, New Zealand. She received her BEng (1987), MSc (1991) and PhD (1994) degrees from the University of Nis, Yugoslavia. Her main research interest is the application of artificial intelligence techniques to education. She has authored and coauthored more than 90 research papers published in international and national journals and conferences. She is the Leader of the Intelligent Computer Tutoring Group, where a new methodology for developing intelligent tutoring systems has been created. Several ITSs based on this methodology have been evaluated in real classrooms since 1998, and the results show that these systems have a significant positive effect on students' learning.

Vladan Devedzic is an Associate Professor of Computer Science at the Department of Information Systems, FON – School of Business Administration, University of Belgrade, Serbia and Montenegro. He has received all his degrees from the School of Electrical Engineering, University of Belgrade, Serbia and Montenegro (BS, 1982; MS, 1988; PhD, 1993). His main research interests include software engineering, intelligent systems, knowledge representation, ontologies, intelligent reasoning, and applications of artificial intelligence techniques to education and medicine. So far, he has authored and coauthored about 200 research papers published in international and national journals and conferences. His major long-term professional goal is to bring close together ideas from the broad fields of intelligent systems and software engineering. He has developed several practical intelligent systems and tools, and actively participates as a consultant to several ongoing projects in industry.

1 Introduction

Once a team of developers builds a successful intelligent tutoring system (ITS) that helps students learn a part of some knowledge domain efficiently, chances are that they may want to extend and upgrade the system later on and/or port it to another technology. If the knowledge domain is complex enough, the developers may also want to build other tutoring systems to cover other parts of the knowledge domain. Several examples in the history of ITS design and development illustrate this point. For instance, the whole family of cognitive tutors started with the Lisp tutor. Later on, cognitive tutors for geometry and algebra were developed using the same methodology, including support for various types of problems and skills [1]. Likewise, ELM-ART, an intelligent interactive educational system to support learning programming in LISP, had started as an intelligent learning environment that supported example-based programming, intelligent analysis of problem solutions, and advanced testing and debugging facilities; later on, it was upgraded to use a combination of an overlay model and an episodic student model, adaptive navigation support, and course sequencing [2,3]. Also, ELM-ART was integrated with InterBook, an authoring tool for creating electronic textbooks with adaptive annotation of links, and NetCoach, another authoring tool that lets authors create fully adaptive and interactive web-based courses without being required to program or learn a programming language [3]. Similarly, the initial success of SQL-Tutor, an ITS that teaches SQL (the most widely used database query language), has inspired the development of other database tutors [4–6,7]. Another analogous example is the pedagogical agent Adele. As a result of its successful use in teaching family medicine and graduate level geriatric dentistry, a series of web-based courses were developed for continuing medical education at the University of Southern California and at the University of Oregon [8].

In such cases, the developers will want their tutoring systems to reuse parts of the knowledge built in other systems they have developed. Moreover, two different teams may develop two different tutoring systems that partially overlap in expertise. Therefore, the need for automatic knowledge sharing, reuse, and exchange among several different tutoring systems in the same complex domain is highly likely to arise. This is especially

the case in web-based education, where the systems may be distributed across a number of sites.

This paper analyses the use of ontologies in such settings. We propose extending the architecture of web-based ITSs with a module called the ontology processor that performs all ontology-related knowledge processing in the system, thus helping the traditional components (the student modeller and the pedagogical module) handle such knowledge and interoperability issues more effectively. We also discuss using web services in the context of web-based ITSs and building networks of educational ontologies.

We first briefly look at other relevant ontology-related work in the ITS domain in Section 2. We present the model of a *Multitutor Ontology-Based LearnInG Environment* (M-OBLIGE) in Section 3. In Section 4 we discuss how that model would be applied in the context of database tutors [4], and how the architectures of these systems should change in order to support the use of ontologies. We discuss the initial steps we took in making ontology-related enhancements to the database tutors in Section 5. The last section contains conclusions.

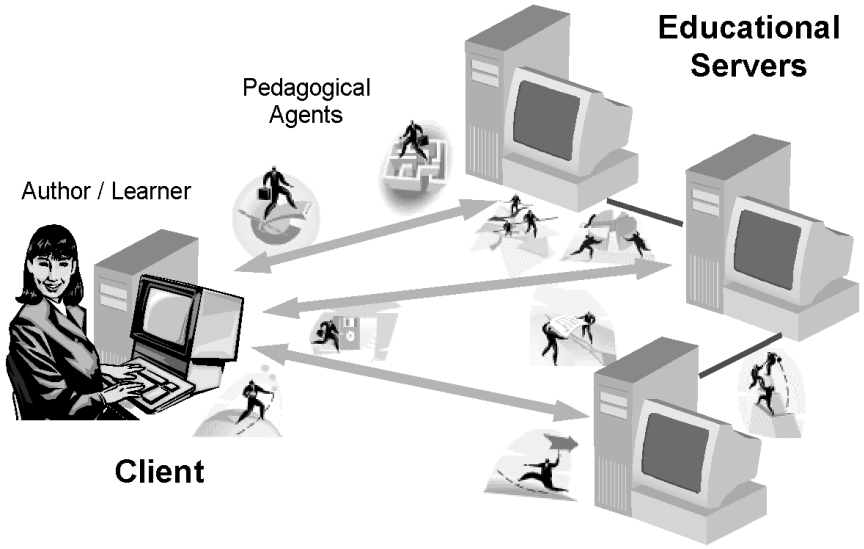
2 ITS and ontology-related work

The way we consider ontologies in our work is best described by Swartout and Tate's definition: "Ontologies provide the necessary armature around which knowledge bases should be built" [9]. In other words, ontologies compose the necessary reference knowledge for making two or more related intelligent systems *knowledge-consistent*. Our emphasis is on the infrastructure role of ontologies for knowledge sharing among related web-based educational systems.

We put our research in the context of the emerging *semantic web*, a huge network of machine-understandable and machine-processable human knowledge, not just ordinary information [10,11–13,14]. The semantic web is expected to provide explicit representation of the semantics of data in the form of various domain theories stored in many web-servers as a myriad of shareable ontologies, as well as advanced, automated, ontology-supported, and agent-ready reasoning services (http://www.semanticweb.org/). Whereas the learning technology and ITSs for the semantic web have just started to appear, Figure 1 illustrates a typical educational setting in the context of the semantic web. *Pedagogical agents* provide the necessary infrastructure for knowledge and information flow between the clients and the servers – they interact with students/learners and authors/teachers, and collaborate with other similar agents [8]. Ontology-aware pedagogical agents very useful in locating, browsing, selecting, arranging, integrating, and otherwise using educational material from different *educational servers*. They access *educational content* on a server by using the available *educational services*, such as learning services, assessment services, collaboration services, and services related to useful references. The server possesses enough intelligence to arrange for *personalisation* of the learning tasks it supports. In fact, from the learner's perspective, the server appears to act as an intelligent tutor with both *domain* and *pedagogical* knowledge to conduct a learning session. It uses a *presentation planner* to select, prepare, and adapt the domain material to show to the student. It also gradually builds the *student model* during the session, in order to keep track of his or her actions and learning progress, detect and

correct the student's errors and misconceptions, and possibly redirect the session accordingly.

Figure 1 Pedagogical agents in the context of web-based education



The most notable work in the ITS community related to the development of educational ontologies, comes from the Mizoguchi Lab at Osaka University, Japan [15,16], and from Tom Murray [17]. Mizoguchi et al. [15] indicate that the *ITS ontology* as a whole consists of *domain ontology*, which characterises the domain knowledge, and *task ontology*, which characterises the computational architecture of knowledge-based systems. They also make an important contribution to the hierarchy of ontologies in the domain of education, and study how the use of ontologies can contribute to the architecture of ITSs and ITS shells and authoring tools. Mizoguchi and Bourdeau [16] point out many limitations of current authoring tools in general, all of which pertain to tools for building web-based educational systems as well. Some of them are:

- authoring tools are not 'theory-aware', which means that they have neither explicit, built-in declarative representations of various domain theories, instructional design theories, and learning theories, nor links to such representations – it is the developer, not the tools, who knows the theories
- there is a deep conceptual gap between authoring systems and authors
- authoring tools, such as TopClass, WebCT, Authorware, LearningSpace, CourseInfo, Cyberprof, Mallard, CM Online, and the like, are neither intelligent nor particularly user-friendly
- authoring tools themselves provide poor support for sharing and reuse of knowledge and components developed for other educational systems.

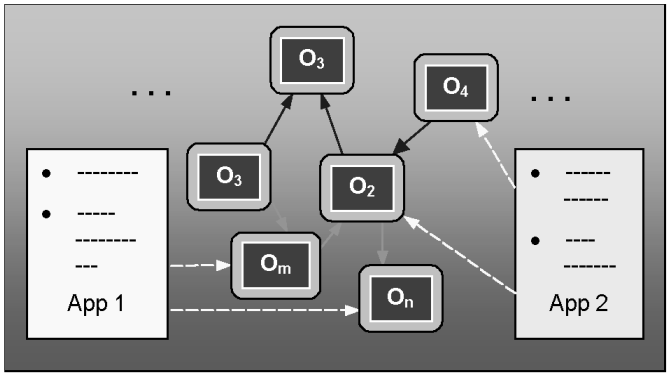
Murray [17] defines the important *topic ontology*, based on topic types (e.g., concept, fact, principle), topic link types (e.g., is-a, part-of, prerequisite, context-for), and topic properties (e.g., importance, difficulty). More recently, Devedzic studied educational services and servers in the context of ontologies on the semantic web [18]. Abraham and Yacef [19] experiment with their XML Tutor in delivering personalised instruction when domain ontology is represented in XML. Cimolino and Kay [20] present a system that supports students in creating concept mapping tasks intended to capture the student's understanding of the ontology of a small domain, as well as to infer his or her misconceptions in the learning process. SITS (Scrutable Intelligent Teaching System) deals with the problem of different (authors) understandings of what is most important and how things are related within a domain, i.e. with the existence of different ontologies underlying the sets of teaching documents created by different authors [21]. The approach used to handle this problem is the automatic extraction of the ontology from teaching documents metadata, which are kept separate from the documents. Apted and Kay [22] go one step further by building a system that automatically constructs an extensive ontology of computer science starting from FOLDOC, the Free On-Line Dictionary of Computing, and using it as a basis for making inferences about student models and other reasoning.

Kassist [23] is a workbench for planning problem solving workflow. It takes into account an important difference between the models of problem solving processes and learning processes, and is based on an ontology for enhancing the learners' meta-cognition of their work. Sicilia et al. [24] introduce the concept of a *learning link*, as a context-independent, typed entity that can be used to represent (possibly imprecise) semantic relationships between learning resources on the web. Examples of good engineering design of ontological support for web courseware authoring include the recently ontology-enhanced AIMS architecture [25] and the Ontology editor [26] that enable collaborative ontological engineering involving both a domain expert and an instructional-design expert.

3 The M-OBLIGE model

Two or more web-based ITSs can refer to a common, shared part of their knowledge as in Figure 2. Note that both tutoring systems (App1 and App2) have their own private knowledge and reasoning mechanisms. The web pages corresponding to any single ITS in this scenario must contain pointers to ontologies the system uses as its meta-knowledge. Such *semantic markup* of the system's web pages enables an external agent or application to recognise the 'armature' part of the system's knowledge. The ontologies themselves are stored in a machine-readable form at possibly different locations on the web. They can point to each other as well, since the knowledge from one ontology can refer to that in another. For example, the ontology of, say, *fraction*, can refer to the ontology of *number*, which in turn may be stored at another location on the web.

Figure 2 Two web-based ITSs sharing ontologies



Every such ontology should provide a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic or service [27]. Interoperability and knowledge sharing between different educational applications can be achieved by using appropriate *languages* for representing ontologies and educational contents and services. Current trends in web technology suggest that appropriate representation languages include *XML*, *XML Schema*, *RDF*, and *RDF Schema* languages [10], all developed under the auspices of WWW consortium (http://www.w3.org/XML, http://www.w3.org/RDF). higher-level languages and development environments built on top of those four languages are a good choice for developing ontologies. Some of the most popular examples are OIL, DAML+OIL, and OWL [13], and Protégé-2000 [12]. Such highly interactive, graphical tools support the development of ontologies, and convert them automatically to a language from the XML/RDF family that ensures semantic interoperability and knowledge sharing on the web. M-OBLIGE model follows that design philosophy.

M-OBLIGE is unique among the other efforts related to ontologies and ITSs, discussed in Section 2, in that it stresses the use of semantic markup and intelligent web services [14] as techniques for representing, sharing, discovering, and reusing educational contents stored on educational servers. Generally, web services are web-accessible and computer-interpretable programs that can access databases, sensors, and a variety of other physical devices in order to realise specific services of interest to the users and applications (http://www.webservices.org/). Intelligent web services are featured by specific properties, built-in knowledge, reasoning capabilities, interfaces, and effects that are encoded in an unambiguous, machine-understandable, and agent-ready form. Educational servers shown in Figure 1 are supposed to provide a number of such intelligent educational services that can communicate with pedagogical agents, thus letting them locate, invoke, and compose educational contents and activities for their users automatically. Note that, just like educational contents, educational services need be semantically described by their own ontologies (e.g., ontology of assessment, or ontology of library access) and marked-up accordingly in order for pedagogical agents to locate them and invoke them when accessing web pages of educational servers.

We want to stress that this model of knowledge sharing significantly differs from web-based collaborative learning environments, shared knowledge bases, and shared student models. In collaborative learning environments (e.g. [28]), two or more learners

share the common problem-solving space while working together on completing a task. However, the system cannot share the domain knowledge with other systems. In the case of shared knowledge bases, two or more tutoring systems share the knowledge encoded in the knowledge base but are usually developed using the same technology and tools, and moreover, no other systems can share the same knowledge. Finally, in the case of shared student models, two systems may use the knowledge about the student if it is encoded in a way understandable by both systems [3]. We propose that meta-knowledge is shared between multiple systems, i.e. the static part of the systems' knowledge that makes each system aware of the knowledge models used by the other systems. This is similar to the approach taken in SITS [21].

4 Ontology-based database tutors

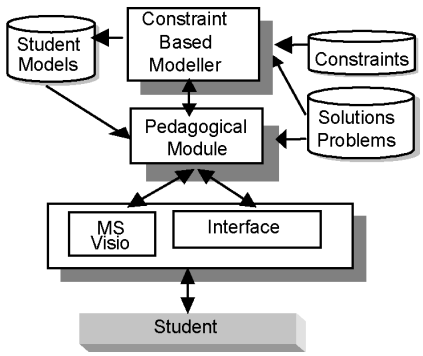
We now illustrate how M-OBLIGE would be applied to a suite of database tutors [4]. At the moment, three of these tutors are operational. KERMIT [7] is a tutor that teaches conceptual database design using the Entity-Relationship (ER) model. SQL-Tutor [5] teaches SQL, the most widely used database language. NORMIT [6] is a data normalisation tutor. Other planned tutors include a tutor for converting ER schemas to relational ones, and tutors for other query languages. The existing systems are not ontology-based.

We use KERMIT in the rest of the paper to illustrate the features of the M-OBLIGE model. We present the architecture of the system and its basic features in Section 4.1. Section 4.2 describes the ontology-based extension of KERMIT.

4.1 KERMIT: a knowledge-based ER modelling tutor

KERMIT is an ITS aimed at the university-level students' learning conceptual database design. The architecture of the system is illustrated in Figure 3. KERMIT is a problem-solving environment in which students practice database design using the ER data model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a student modeller, which analyses student answers and generates student models. At the moment, there is a stand-alone version of KERMIT running on MS Windows, and the system is being ported to the web.

Figure 3 The architecture of KERMIT



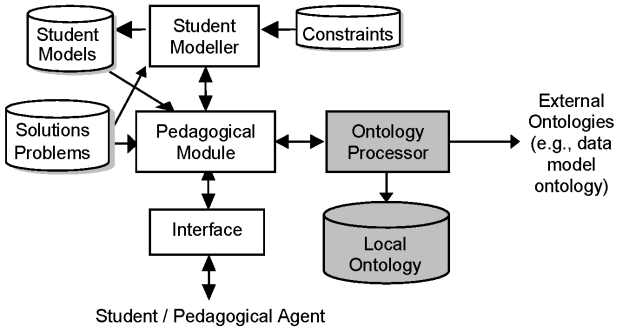
KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the correctness of the student's solution, KERMIT compares it to the correct solution, using domain knowledge represented in the form of more than 90 constraints. It uses Constraint-Based Modelling [29] to model the domain and student's knowledge. Students have several ways of selecting problems in KERMIT. They may work their way through a series of problems, arranged according to their complexity. The other option is a system-selected problem, when the pedagogical module selects a problem for the student on the basis of his/her student model. The interface is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting a feedback level. The middle window is the main working area, in which the student draws the ER diagram. The feedback from the system is displayed in the bottom part of the screen.

4.2 Extending KERMIT

Each tutoring system in the database suite will have its own local ontology, supporting the tasks being taught. Although the domains of these systems are not identical, there is an overlap between them: there may be several fairly general DB ontologies as a basis for knowledge reuse in all tutors. For example, a database schema that comes out of KERMIT is expressed as an ER schema, which may be converted into the corresponding relational schema. SQL-Tutor (and other query language tutors) may refer to this relational schema when the student builds queries. Such an extension of KERMIT would enable students to practice not only database design problems within the system, but also, when the need arises, to improve their knowledge in the related areas (like the relational data model and languages) by solving problems in the other database tutors. Local ontologies are specific to each tutoring system, as each system focuses on particular domain aspects. On the other hand, the external ontologies are general, describing the whole domain, and may be used by external tutoring systems and other applications (such as [28]).

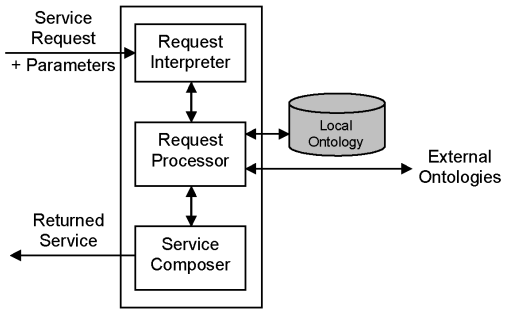
In order to be able to use ontologies, the architecture of the tutoring systems needs to be modified. For example, the architecture of KERMIT will be extended as in Figure 4. Its local ontology refers to the idiosyncrasies of the implemented ER model. On the other hand, the ontologies of data models and databases are general enough to be used by other DB tutors and other intelligent DB applications alike.

Figure 4 Ontology extension for KERMIT



The ontology processor will help the pedagogical module in making instructional decisions and enable external agents to find out about KERMIT's instructional focus. Figure 5 shows the internal structure of the ontology processor. The pedagogical module can request a service from the ontology processor either due to its internal processing, or in order to interpret a request for and provide a certain educational service. Note that a service provided by the ontology processor is *not* the equivalent of an educational service – the latter is accessible externally, through the KERMIT's web page. The *Request interpreter* of the ontology processor parses the request and its parameters (e.g., a request can carry an external URL) and possibly decomposes it into a sequence of simpler tasks to be carried out by the *Request processor*. Certain requests can refer to the *Local ontology* only, whereas others require an external ontology to be accessed. In either case, processing the ontology may result in a set of items, such as a sequence or a tree of database concepts, an indicator of a relevant part of the *Local ontology*, or the URL of an external ontology. The *Service composer* takes such items and rearranges them into an output form suitable for the pedagogical module. This activity may involve adaptivity, heuristics, and various constraint processing. For example, the *Request processor* may return a set of concepts from an external ontology, whereas the *Service composer* may need to relate them to the input parameters.

Figure 5 Ontology processor



The following two scenarios illustrate how the pedagogical module and ontology processor can interact.

Scenario 1. The learner is practising the use of the concept of *regular relationship type* by solving a specific ER modelling problem. Suppose the learner does not specify the cardinality constraint for a defined relationship type. The student modeller identifies the mistake in the current solution, and updates the student model accordingly. The pedagogical module learns from the student model that the student understands neither the cardinality constraint, nor the participation constraint (these two constraints are integrity rules defined in the ER model for the relationship types). Furthermore, the student does not understand the integrities of the entity types. The pedagogical module then decides using the local ontology that the student has a problem with the integrity component of the ER data model, and requires the ontology processor to find appropriate external ontologies and learning resources based on them that cover the integrity component in a general way. The ontology processor consults the external ontology, gathers information about related learning resources and requires services from them, so that the student can learn about the required concept.

Scenario 2. A distant learner's agent consults KERMIT's web page about the possibility for the learner to find out examples of binary relationships in conceptual database modelling. The 'Examples' service invokes the Ontology processor, which could perform an ordinary text-based search in the KERMIT's knowledge and databases. Since the returned information based on such a trivial search may not be fully relevant, the Ontology processor performs semantic search instead. It parses the remote ontology represented at the site identified by the supplied URL, and finds out the relevant hierarchy of concepts: binary relationship – regular relationship – relationship type – ER construct. By comparing them to KERMIT's local ontology of ER model, as well as to the (external) more general ontology of data models, it provides the learner's agent the exact pointers to the parts of KERMIT's knowledge focusing on examples of binary relationships.

As stated above, the web-enabled version of KERMIT is currently under development. We have also started with the implementation of the ontology processor, taking into account that it needs to be usable not only in the specific area of instruction, but also in other ITSs. After completing the extended version of KERMIT, we plan to extend the other database tutors and integrate them through ontologies.

5 Ontological engineering of database tutors

In the course of developing the initial ontological support for database tutors, we developed the ER modelling ontology for KERMIT, as well as the data model ontology that KERMIT can share with other database tutors. To develop these ontologies, we used Protégé-2000 ontology development tool [12]. Protégé-2000 can store ontologies in different formats, one of them being RDF Schema – an important language for building semantic web applications, providing semantic interoperability between applications, and storing ontologies in a machine-understandable form [27].

5.1 The strategy of ontology development

In developing the ontologies, we used the following guidelines suggested by the semantic web community and discussed in detail in [18]:

- *Planning for expected uses of the ontology*. In our case, the expected applications to use the ontologies include (but are not limited to) the suite of database tutors mentioned in Section 4.
- *Start by building small, largely incomplete ontologies, and let them grow incrementally and iteratively over time, as opposed to working on an elaborated, complex conceptual design of ontologies for a long period of time before actually deploying them.* This way we wanted to avoid the 'analysis paralysis', i.e. the danger of just thinking about something forever, without putting it to life [30].
- *Test the initial ontology by using it to build a knowledge base.* We used the knowledge already built in KERMIT, NORMIT, and SQL-Tutor, in order to instantiate our ontologies.
- *Iterate over the above steps in order to revise and upgrade the initial ontology incrementally.*

5.2 KERMIT's local ontology

KERMIT teaches the ER model, hence its local ontology includes *ER modelling ontology*. That ontology is useful for many other applications, including the suite of database tutors. Figure 6 illustrates a part of the hierarchy of the concepts in the ER model that we have derived starting from the database design domain theory (e.g., see [31]). Starting from such a concept hierarchy, we developed the ER modelling ontology to the level of detail that allowed for suitable redesign of KERMIT's knowledge base to include ontological support (KERMIT teaches the original version of the ER model, known as the Chen model). Figure 7 shows a screenshot of our Protégé-2000 implementation of the ER modelling ontology. Not all parts of the ontology can be seen in the screenshot, but two of the most important domain concepts – *ER construct* and *ER attribute* – are shown explicitly.

Figure 6 Partial hierarchy of concepts in the ER model

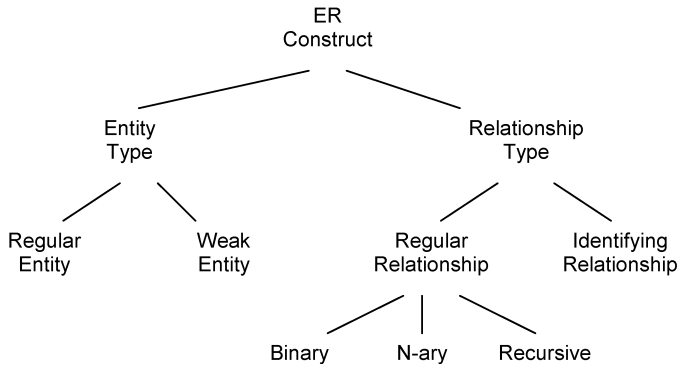
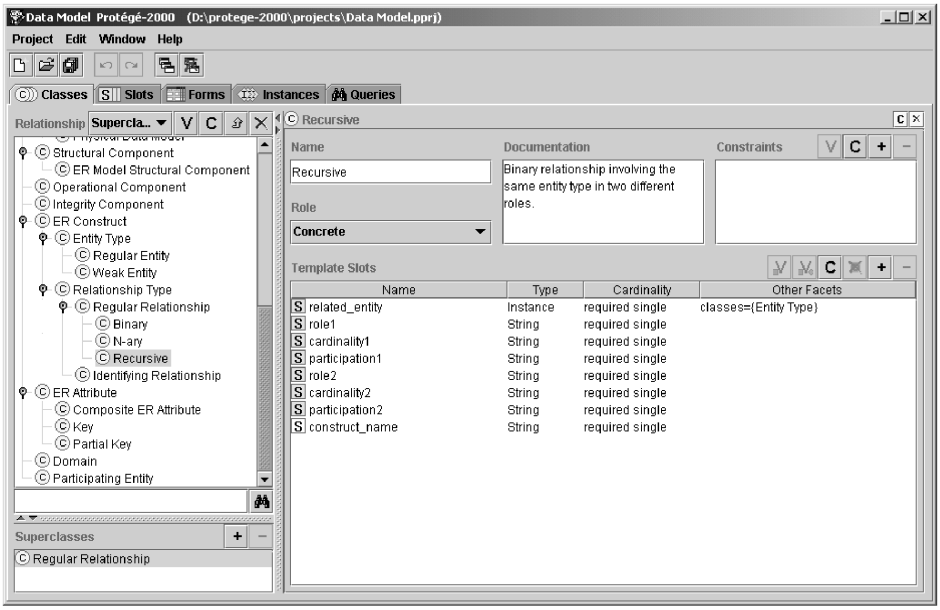
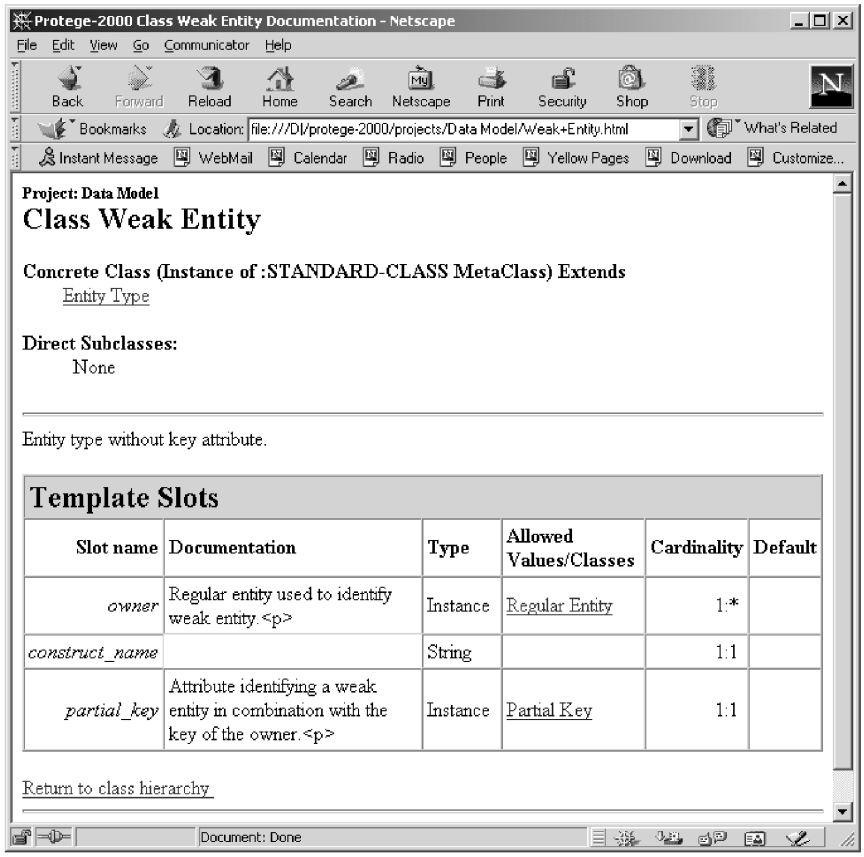


Figure 7 Initial implementation of ER modelling ontology in Protégé-2000



After developing the ontology, we exported it into a set of HTML pages for easier reference. Figure 8 shows an example HTML page, representing the concept of *weak entity* from our ER modelling ontology (the same concept is also visible in Figure 7). Each concept in an ontology developed by Protégé-2000 can be represented by an equivalent HTML page, although in the case of a more complex concept, the page itself can be rather long.

Figure 8 HTML representation of the *Weak Entity* concept from ER modelling ontology



5.3 A shared ontology for a database tutors suite

As explained earlier, the local ontology in KERMIT describes the ER model as implemented in the system. There are many extensions and variants of the ER model [31]. Furthermore, the ER model is only a special case of the more general concept of *conceptual data model*, and there is a yet more general concept, the *data model* concept. We recognised these relations in developing our ontologies, and realised that other database tutors (as well as other applications) will need to rely on such more general ontologies in order to reuse and share common knowledge more efficiently. We developed an external data model ontology, again using Protégé-2000. Figure 9 shows the topmost HTML page for the external ontology, generated by Protégé-2000, while Figure 10 shows more details.

Figure 9 The data model ontology in HTML

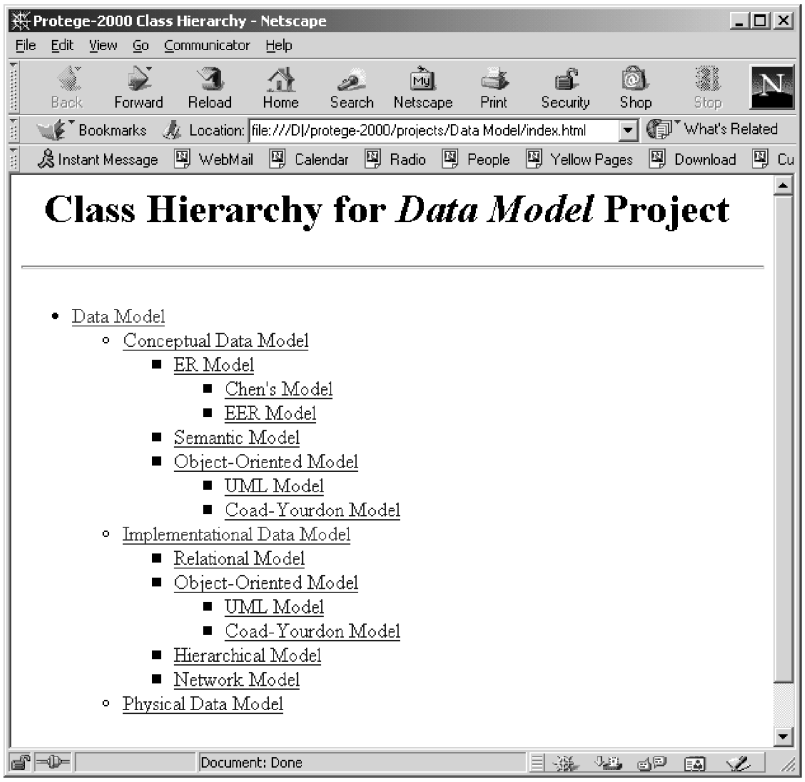
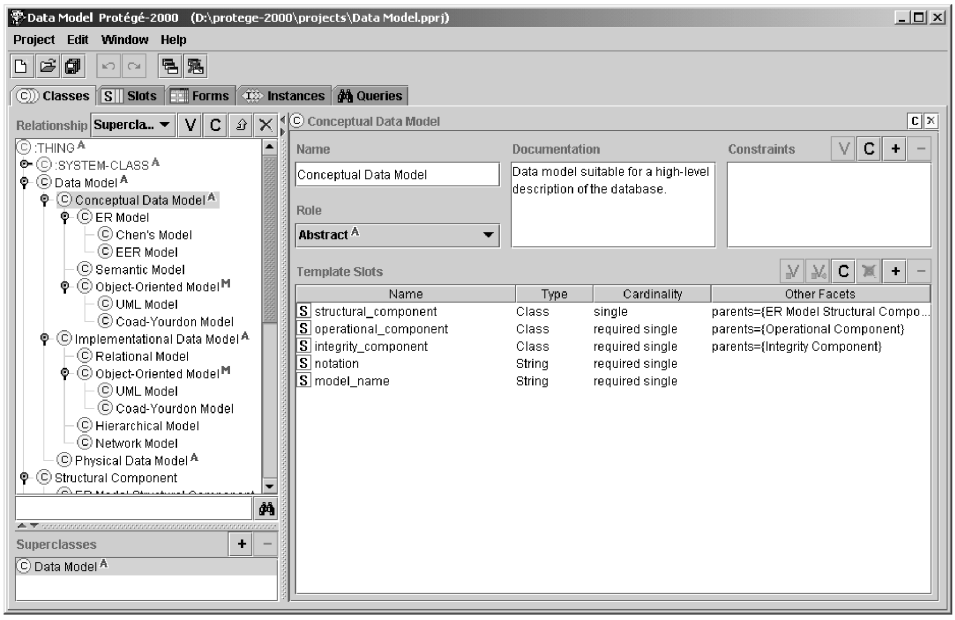
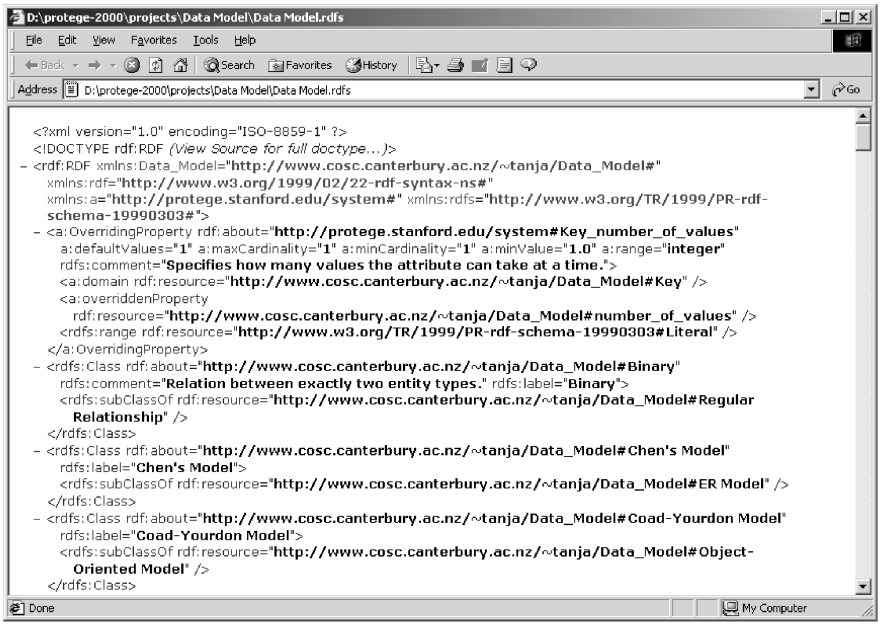


Figure 10 Initial implementation of data model ontology in Protégé-2000



Since M-OBLIGE prefers XML/RDF representation of ontologies in order to make them web-ready and agent-ready, we converted our ontologies into RDF format. Figure 11 shows a part the data model ontology in that format. Revisiting Figure 3, it is important to note that data model ontology in RDF format can be stored anywhere on the web. It suffices for each database tutor in the suite to know the location of the data model ontology and to include an RDF parser in its ontology processor in order to reuse the knowledge of data models encoded in that ontology. Moreover, marking up the web page

Figure 11 A part of RDF representation of data model ontology



corresponding to a database tutor with a pointer to the data model ontology and storing the markup with the web page enables external applications and intelligent agents to automatically machine-interpret the shared part of the tutor's knowledge. A learner eager to find out more about database concepts from the web can now send out his or her pedagogical agent to check out known database tutors. From the markup on the tutors' web pages, the agent can identify the underlying ontology and parse it in order to find whether the tutors' knowledge suits the learner's needs. All these actions of the agent are fully viable with current web technology.

6 Conclusions

There is a growing need for applying ontologies in the field of ITSs. Artificial tutors often need to exchange and reuse their knowledge, and ontologies provide infrastructure for that. The M-OBLIGE model proposed in this paper is our first step towards a unifying framework for enabling interoperability of multiple tutors on the web. We presented the model in general, and showed how it can be used to extend an existing system so that it can be used in a broader context of a suite of database tutors.

The major contributions of the M-OBLIGE model to the growing field of ontology-aware web-based ITSs are:

- the introduction of the ontology processor in the architecture of web-based ITSs
- stressing the benefits of deploying the technology of intelligent web services in web-based ITSs, along with using ontologies and pedagogical agents
- the notion of interoperability of a suite of intelligent tutors, enabled by a set of ontologies.

In spite of the fact that we presented M-OBLIGE with a clear specific goal in mind (supporting interoperability for a suite of database tutors), the model is general enough to support any number of interoperating tutors. Its major limitation is related to the fact that practical educational ontologies are still lacking, so the further development, refinement, and deployment of the proposed model in other educational domains require the development of ontologies first. Future standardisation of educational web services will certainly speed up further development of M-OBLIGE.

The work reported in this paper is still in early stages. After completing work on KERMIT, we plan to extend two other database tutors, and further explore the effectiveness of the proposed model.

Acknowledgements

The authors thank the Erskine Fund of the University of Canterbury, for funding the visit of the second author to New Zealand. This research could not be done without the help of all members of the Intelligent Computer Tutoring Group.

References

1 Anderson, J.R., Corbett, A.T., Koedinger, K.R. and Pelletier, R. (1995) 'Cognitive tutors: lessons learned', *Journal of the Learning Sciences*, Vol. 4, No. 2, pp.167–207.

2 Brusilovsky, P., Ritter, S. and Schwarz, E. (1997) 'Distributed intelligent tutoring on the web', in du Boulay, B. and Mizoguchi, R. (Eds.): *Proceedings of AI-ED'97, 8th World Conference on Artificial Intelligence in Education*, 18–22 August, pp.482–489.

3 Weber, G. and Brusilovsky, P. (2001) 'ELM-ART: an adaptive versatile system for web-based instruction', *International Journal on Artificial Intelligence in Education*, Vol. 12, pp.351–384.

4 Mitrovic, A., Mayo, M., Suraweera, P. and Martin, B. (2001) 'Constraint-based tutors: a success story', *Proceedings of the Industrial & Engineering Application of Artificial Intelligence & Expert Systems Conference IEA/AIE-2001*, pp.931–940.

5 Mitrovic, A., Martin, B. and Mayo, M. (2002) 'Using evaluation to shape ITS design: results and experiences with SQL-Tutor', *International Journal of User Modeling and User-Adapted Interaction*, Vol. 12, Nos. 2–3, pp.243–279.

6 Mitrovic, A. (2002) 'NORMIT, a web-enabled tutor for database normalization', in Kinshuk, R., Lewis, K., Akahori, R., Kemp, T., Okamoto, L. and Henderson, C-H. (Eds.): *Proceedings International Conference on Computers in Education ICCE 2002*, Auckland, pp.1276–1280.

7 Suraweera, P. and Mitrovic, A. (2002) 'KERMIT: a constraint-based tutor for database modeling', in Cerri, S., Gouarderes, G. and Paraguacu, F. (Eds.): *Proc. 6th Int. Conf. Intelligent Tutoring Systems ITS 2002*, Biarritz, France, LCNS 2363, pp.377–387.

8 Johnson, W.L., Rickel, J. and Lester, J.C. (2000) 'Animated pedagogical agents: face-to-face interaction in interactive learning environments', *International Journal of Artificial Intelligence in Education*, Vol. 11, pp.47–78.

9 Swartout, W. and Tate, A. (1999) 'Ontologies, guest editors' introduction, IEEE', *Intelligent Systems*, Vol. 14, No. 1, pp.18–19.

10 Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M. and Horrocks, I. (2000) 'The semantic web: the roles of XML and RDF', *IEEE Internet Computing*, Vol. 4, No. 5, pp.63–74.

11 Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D.L. and Patel-Schneider, P.F. (2001) 'OIL: an ontology infrastructure for the semantic web', *IEEE Intelligent Systems*, Vol. 16, No. 2, pp.38–45.

12 Friedman-Noy, N., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W. and Musen, M.A. (2001) 'Creating semantic web contents with Protégé-2000, *IEEE Intelligent Systems*, Vol. 162, pp.60–71.

13 Hendler, J. (2001) 'Agents and the semantic web', *IEEE Intelligent Systems*, Vol. 16, No. 2, pp.30–37.

14 McIlraith, S.A., Son, T.C. and Zeng, H. (2001) 'Semantic web services', *IEEE Intelligent Systems*, Vol. 16, No. 2, pp.46–53.

15 Mizoguchi, R., Sinitsa, K. and Ikeda, M. (1996) 'Task ontology design for intelligent educational/training systems', *Proc. Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs*, Montreal, Canada, pp.1–20.

16 Mizoguchi, R. and Bourdeau, J. (2000) 'Using ontological engineering to overcome common AI-ED problems', *International Journal on Artificial Intelligence in Education*, Vol. 11, pp.1–12.

17 Murray, T. (1998) 'Authoring knowledge-based tutors: tools for content, instructional strategy, student model, and interface design', *Journal of Learning Sciences*, Vol. 7, No. 1, pp.5–64.

18 Devedzic, V. (2003) 'Next-generation web-based education', *International Journal on Engineering Education & Life-Long Learning*, Vol. 13, Nos. 3–4, pp.232–247.

19 Abraham, D. and Yacef, K. (2002) 'XMLTutor – an authoring tool for factual domains', in Aroyo, L. and Dicheva, D. (Eds.): *Proceedings of ICCE 2002 Workshop on Concepts and Ontologies in Web-based Educational Systems*, Auckland, New Zealand, pp.7–10.

20 Cimolino, L. and Kay, J. (2002) 'Verified concept mapping for eliciting conceptual understanding', in Aroyo, L. and Dicheva, D. (Eds.): *Proceedings of ICCE 2002 Workshop on Concepts and Ontologies in Web-based Educational Systems*, Auckland, New Zealand, pp.11-16.

21 Kay, J. and Holden, S. (2002) 'Automatic extraction of ontologies from teaching document metadata', in Aroyo, L. and Dicheva, D. (Eds.): *Proceedings of ICCE 2002 Workshop on Concepts and Ontologies in Web-based Educational Systems*, Auckland, New Zealand, pp.25–28.

22 Apted, T. and Kay, J. (2002) 'Automatic construction of learning ontologies', in Aroyo, L. and Dicheva, D. (Eds.): *Proceedings of ICCE 2002 Workshop on Concepts and Ontologies in Web-based Educational Systems*, Auckland, New Zealand, pp.57–64.

23 Seta, K. and Umano, M. (2002) 'A support system for planning problem solving workflow', in Aroyo, L. and Dicheva, D. (Eds.): *Proceedings of ICCE 2002 Workshop on Concepts and Ontologies in Web-based Educational Systems*, Auckland, New Zealand, pp.29–35.

24 Sicilia, M.A., García, E., Díaz, P. and Aedo, I. (2002) 'LEARNING LINKS: reusable assets with support for vagueness and ontology-based typing', in Aroyo, L. and Dicheva, D. (Eds.): *Proceedings of ICCE 2002 Workshop on Concepts and Ontologies in Web-based Educational Systems*, Auckland, New Zealand, pp.37–42.

25 Aroyo, L., Dicheva, D. and Cristea, A. (2002) 'Ontological support for web courseware authoring', in Cerri, S., Gouarderes, G. and Paraguacu, F. (Eds.): *Proceedings of the 6th International Conference on Intelligent Tutoring Systems ITS'2002*, Biarritz, France and San Sebastian, Spain, June 2–7, pp.270–280.

26 Bourdeau, J. and Mizoguchi, R. (2002) 'Collaborative ontological engineering of instructional design knowledge for an ITS authoring environment', in Cerri, S., Gouarderes, G. and Paraguacu, F. (Eds.): *Proceedings of the 6th International Conference on Intelligent Tutoring Systems ITS 2002*, Biarritz, France and San Sebastian, Spain, June 2–7, pp.399–409.

27 Gómez-Pérez, A. and Corcho, O. (2002) 'Ontology languages for the semantic web', *IEEE Intelligent Systems*, Vol. 17, No. 1, January–February, pp.54–60.

28 Constantino-Gonzalez, M.d.l.A. and Suthers, D.D. (2001) 'Designing and evaluating a collaboration coach: knowledge and reasoning', in Moore, J.D., Redfield, C.L. and Johnson, W.L. (Eds.): *Proceedings. of 10th International Conference on Artificial Intelligence in Education*, San Antonio, Texas, pp.176–187.

29 Ohlsson, S. (1994) 'Constraint-based student modelling', in Greer, J.E. and McCalla, G. (Eds.): *Student Modelling: The Key to Individualized Knowledge-based Instruction*, Springer-Verlag, Berlin, pp.167–189.

30 Devedzic, V. (2002) 'Understanding ontological engineering', *Communications of the ACM*, Vol. 45, No. 4, pp.136–144.

31 Elmasri, R. and Navathe, S.B. (2000) *Fundamentals of Database Systems*, 3rd ed., Addison Wesley.