Database Exploratorium: A Semantically Integrated Adaptive Educational System

Peter Brusilovsky[1], Antonija Mitrovic[2]
Sergey Sosnovsky[1], Moffat Mathews[2],
Michael Yudelson[1], Danielle H. Lee[1], Vladimir Zadorozhny[1]

[1] University of Pittsburgh, School of Information Sciences,
135, North Bellefield ave. Pittsburgh, PA 15260, USA
[2] University of Canterbury, Dept. of Computer Science and Software Engineering
Private Bag 4800, Christchurch 8140, New Zealand
peterb@pitt.edu, Tanja.Mitrovic@canterbury.ac.nz,
sosnovsky@gmail.com, moffat@cosc.canterbury.ac.nz,
myudelson@gmail.com, suleehs@gmail.com, vladimir@sis.pitt.edu

Abstract. With the growth of adaptive educational systems available to students, integration of these systems is changing from an interesting research problem into an important practical task. One of the challenges that need to be accepted on the way is the development of mechanisms for student model integration. The architectural principles and representation technologies employed by the adaptive educational systems define the applicability of a particular integration approach. This paper overviews the existing mechanisms and detail one of them: the evidence integration.

1 Introduction

Adaptive Web-based Educational Systems (AWBES) emerged into an active research field over 10 years ago [1]. Since that time, a number of adaptive educational systems available on the Web has been constantly increasing. In some popular subject areas the "density" of AWBES is reaching the point where several adaptive systems are available. In most of the cases, these systems do not compete, but rather complement each other making it possible and even desirable to use these systems in parallel to teach a specific subject like physics, algebra, or programming. An integrated "mega-tutor", a vision shared by many AIED experts [2-5] in the early days of AWBES becomes a practical problem.

From our prospect, the main challenge of using several AWBES in parallel as a distributed system is to achieve "true integration" and make the whole more than the sum of its parts. On the student side, it means that a student should be able to use several systems in parallel transparently and with no additional overhead. A single login (required by almost all AWBES) should be sufficient to work with any numbers of systems involved into teaching the same subject. On the system side, it means that each of the participating systems should have a chance to increase the quality of student modeling and adaptation using integrated evidence about the student, which was collected by all participating AWBES. Achieving both kinds of integration is a reasonable technical challenge, which could be best supported by a dedicated integration framework such as Medea [6] or ADAPT^2 [7].

The major problem in the process of building a distributed AWBES for a specific subject is, however, not technical, but conceptual. To benefit from their complementary knowledge about the same student, two AWBES need to understand each other's approach to represent information about the student. This is very hard to achieve in practice since two different systems, even in the same subject area, are typically using very different student models. To achieve progress in conceptual integration of multiple AWBES we need to learn how to translate information collected by one system into format, which could be understood by another system.

The project presented in this paper attempted to explore a problem of AWBES integration and distributed student modeling using a practical, but challenging case of two essentially different models – a concept-based overlay and a constraint-based model. Our project was motivated by a practical goal – building a distributed AWBES for an important domain of SQL programming. Several components of this AWBES were already integrated into a "Database Exploratorium" using ADAPT^2 framework [8]. Among these components were WebEx, a system for interactive example exploration and a SQL-KnoT, a system for generating and evaluating database questions. These systems shared the same domain model, so their integration was relatively straightforward. The paper is focused on the most challenging step of the integration process: the inclusion of a well-known Intelligent Tutoring system SQL-Tutor [9], with its substantially different mechanisms of domain and user modeling.

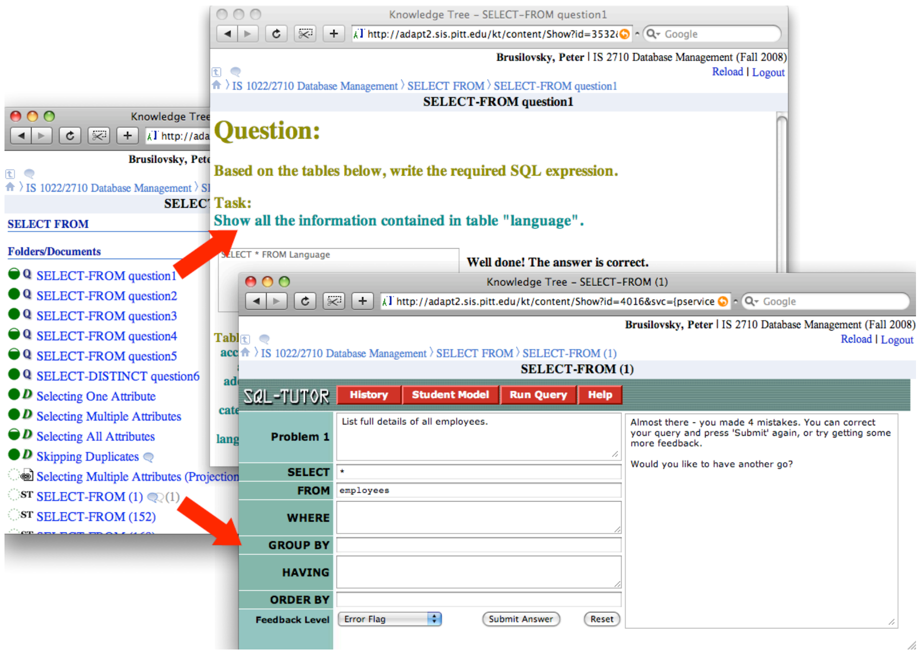
The paper is organized as follows. Sections 2 and 3 introduce the systems being integrated – Database Exploratorium and SQL Tutor including the employed mechanisms of student modeling. Section 4 presents in details our approach to AWBES integration, which is based on semantic-level mapping between the domain models. Section 5 summarizes the results of the classroom study of the integrated system. It concludes with a summary of the work done and a discussion of future plans.

2 Database Exploratorium

The Integrated Exploratorium for Database Courses [8] has been developed in the University of Pittsburgh to investigate the technical problems and the pedagogical benefits of using several kinds of interactive tools in a single learning environment. By the time we started the SQL Tutor integration project, the Exploratorium had already provided personalized access to three types of interactive learning activities: annotated examples, self-assessment questions and the SQL labs. Technical integration of these components was supported by the Knowledge Tree course portal providing a single sign-on access to all three systems. Conceptual integration was maintained by the user modeling server CUMULATE [10], which stored the integrated model of student knowledge of SQL programming language. The student models were built as overlays of SQL Ontology that has been developed as a collaborative effort of University of Pittsburgh and University of Canterbury [11].

2.1 The Adaptive Knowledge Tree Portal

Knowledge Tree portal offers students a single sign-on and personalized access to all kinds of learning resources available in the Exploratorium. The content and the structure of the information available through a portal to students taking a specific course is determined by a teacher of the course who could arrange the learning content according to the needs of the course. Knowledge Tree is implemented using a common folder-document paradigm. Each course is structured as a sequence of nested folders (for example, lecture folders, if the teacher chooses to structure material by lecture). Lecture folders contain individual resources relevant to this lecture such as SQL-KnoT and SQL Tutor problems, WebEx examples, and other course materials, which the teacher chooses to provide for this lecture. The Knowledge Tree interface is shown in fig. 1. This figure represent the state of the portal after the completion of the integration process presented in the section 4, when SQL Tutor problems became available through the portal. Here, the left window presents a list of items in a lecture folder. Windows on the right show an SQL-KnoT problem (top) and SQL Tutor problem (bottom) from that lecture.


Knowledge Tree – SELECT-FROM question1
http://adapt2.sis.pitt.edu/kt/content/Show?id=3532

Brusilovsky, Peter | IS 2710 Database Management (Fall 2008)
Reload | Logout
IS 1022/2710 Database Management 〉 SELECT FROM 〉 SELECT-FROM question1

SELECT-FROM question1

Question:

Based on the tables below, write the required SQL expression.

Task:
Show all the information contained in table "language".

SELECT * FROM Language

Well done! The answer is correct.

SELECT FROM

Folders/Documents

- ◒ Q SELECT-FROM question1
- ● Q SELECT-FROM question2
- ● Q SELECT-FROM question3
- ◒ Q SELECT-FROM question4
- ◒ Q SELECT-FROM question5
- ● Q SELECT-DISTINCT question6
- ● D Selecting One Attribute
- ● D Selecting Multiple Attributes
- ◒ D Selecting All Attributes
- ● D Skipping Duplicates
- ◌ Selecting Multiple Attributes (Projection
- ◌ ST SELECT-FROM (1) (1)
- ◌ ST SELECT-FROM (152)

Knowledge Tree – SELECT-FROM (1)
http://adapt2.sis.pitt.edu/kt/content/Show?id=4016&svc={pservice

Brusilovsky, Peter | IS 2710 Database Management (Fall 2008)
Reload | Logout
IS 1022/2710 Database Management 〉 SELECT FROM 〉 SELECT-FROM (1)

SELECT-FROM (1)

SQL-TUTOR | History | Student Model | Run Query | Help

Problem 1	List full details of all employees.
SELECT	*
FROM	employees
WHERE	
GROUP BY	
HAVING	
ORDER BY	
Feedback Level	Error Flag

Almost there - you made 4 mistakes. You can correct your query and press 'Submit' again, or try getting some more feedback.

Would you like to have another go?

Fig. 1. An SQL KnoT problem (top right) and SQL Tutor problem (bottom right) accessed through the Knowledge Tree portal (folder with navigation on the left).

Knowledge Tree uses adaptive navigation support approach to guide students to the most appropriate educational activities: it provides an adaptive icon next to the link to each learning resource or a folder. The type of the icon and the adaptation approach depends on the type of the resource and the context. For example, in the current version of the Exploratorium each link to SQL-KnoT problems is annotated with a wholly/partially filled bullet that denotes user knowledge of the material underlying the problem (fig. 1, left). Folder annotations denote the cumulative student progress with resources in that folder.

2.2 SQL KnoT: Knowledge Testing for SQL

The SQL-KnoT (Knowledge Tester) is an original component of the Exploratorium. It offers students an opportunity to test and practice their problem-solving skills. It generates questions that require a student to write an SQL query for a sample database, evaluates the correctness of student's answer, and provides a student with feedback (fig.1, top). SQL-KnoT uses a novel approach to question generation and answer evaluation. Every time a student accesses an SQL-KnoT question, the actual question text is generated by the corresponding template from the set of predefined databases. When SQL-KnoT evaluates a student's answer, it randomly generates several starting states of the question database. After that, SQL-KnoT compares the result produced by the student solution for each database state with the corresponding result produced by the pre-stored correct query (model solution). To be evaluated as correct, the student solution should always produce the same result as the model solution. For the needs of our courses, we have developed about 50 templates capable of generating over 400 actual questions.

2.3 SQL Ontology

SQL Ontology serves as a backbone of the Exploratorium. It was developed as a collaborative effort between the PAWS Lab of the University of Pittsburgh, and the ICT Group of the University of Canterbury [10]. It serves as a basis for the overlay student model and as a vocabulary for indexing learning content. The ontology can be accessed at http://www.sis.pitt.edu/~paws/ont/sql.owl. It is a light-weight OWL-Lite ontology, with more then 200 classes connected via three types of relations: standard rfs:subClassOf (hyponymy relation) and a transitive relation pair sql:isUsedIn – sql:uses, which models the connection between two concepts, where one concept utilizes another. Fig. 2 gives some examples of these relations. More details on the ontology can be found in [11, 12]

```
<sql:WhereClause>        <rdfs:subClassOf>     <sql:Clause>
<sql:SelectStatement>   <rdfs:subClassOf>     <sql:Statement>
<sql:WhereClause>        <sql:isUsedIn>        <sql:SelectStatement>
<sql:SelectStatement>   <sql:uses>            <sql:WhereClause>
```

Fig. 2. Extract from the SQL Ontology.

3. SQL-Tutor and Constrained-based User Modeling

SQL-Tutor is a constraint-based intelligent tutoring system [9] designed to help students learn SQL. It is a part of a family of tools created and maintained by the

Intelligent Computer Tutoring Group (ICTG[1]) [13]. SQL-Tutor has been evaluated in twelve studies since 1998 and has been shown to be effective in supporting students' learning. SQL-Tutor contains about 300 problems relating to a number of databases; the databases provide a context for each problem. The pedagogical module presents students with problems appropriate to their knowledge level. Students have the freedom to ignore the system's suggestion and choose any other problem. The SQL-Tutor interface is shown in fig. 1 (right) and contains the problem definition area, the solution workspace, the feedback message pane, controls, and the problem context area.

SQL-Tutor represents domain knowledge as constraints. Constraints are domain principles that must be satisfied in any correct solution. Each constraint contains two conditions: the relevance condition and the satisfaction condition. A constraint is relevant if the features within the student's solution match the same features described in the relevance condition. The satisfaction condition describes what must be true in order for the solution to be correct. If the student solution violates the satisfaction condition of any relevant constraint, the solution is incorrect. Feedback messages attached to each constraint allow the system to present detailed and specific feedback on violated constraints. The constraint set in SQL-Tutor contains about 700 constraints, which check for syntactic and semantic correctness of the solution. **Fig. 3** illustrates two constraints.

```
(16 "You have to specify the grouping in the GROUP BY clause before you can
specify how to restrict grouping in the HAVING clause!"
(not (null (having ss)))
(not (null (slot-value ss 'group-by)))
"GROUP BY")

(635 "Check the condition involving the nested SELECT!"
(and (not (null (where ss))) (not (null (where is)))
  (match '(?*d1 ?a1 "NOT" "IN" "(" "SELECT" ?d5 "FROM" ?t ?*d2)
          (where is) bindings)
  (not (member "IN" (where ss) :test 'equalp))
  (member "EXISTS" (where ss) :test 'equalp)
  (match '(?*d3 ??n "EXISTS" "(" "SELECT" ?a2 "FROM" ?t ?*d4)
          (where ss) bindings))
(equalp ?n "NOT")
"WHERE")
```

Fig. 3. Two example constraints

4. SQL-Tutor Integration

To integrate SQL-Tutor into the Exploratorium we have it enhanced with a new subsystem called SQL-Tutor Resource Component (STRC). The four modules of STRC are shown in **Fig. 4** and include SQL-Tutor, the mapping module, the authentication module, and the external communications module. Within the STRC, the core engine and modules of SQL-Tutor are treated as "black boxes". A simple internal API allows for basic control requests (for example, requesting a particular

[1] http://www.cosc.canterbury.ac.nz/tanja.mitrovic/ictg.html

problem from SQL-Tutor) while the SQL-Tutor solution evaluator reports student progress.

The fundamental differences in the domain models of SQL-Tutor and SQL-KnoT make reliable automatic alignment of these models rather impractical. A well-established set of ontology mapping techniques cannot be applied to this task due to the unique nature of SQL-Tutor's constraints. A constraint is not directly related to a single concept or a sub-tree of the ontology; instead it models the syntactic or semantic relations between various concepts. The purpose of the mapping module is to take any report from SQL-Tutor (i.e. a short or long-term student model based on constraints) and convert it to a report based on a pre-agreed common ontology used by a particular external server (CUMULATE).

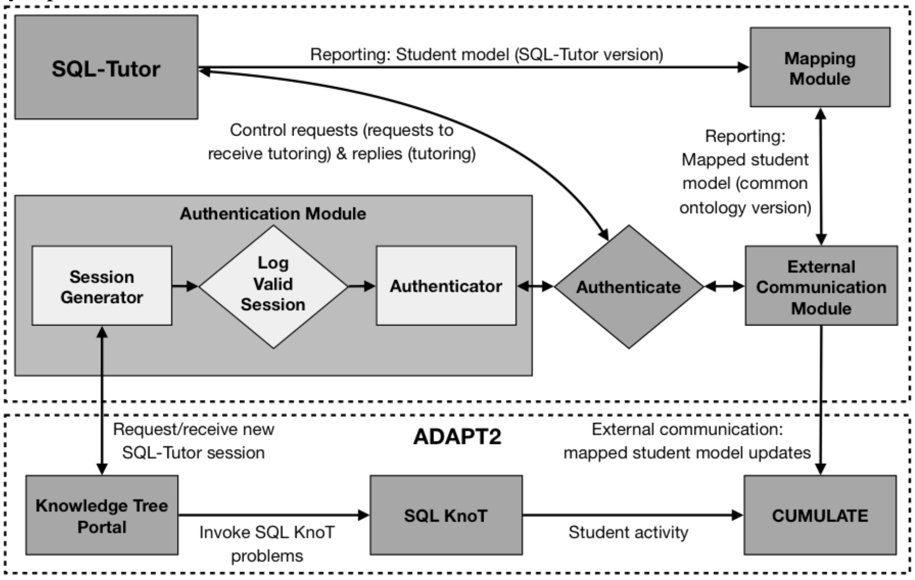

SQL-Tutor
Reporting: Student model (SQL-Tutor version)
Mapping Module
Control requests (requests to receive tutoring) & replies (tutoring)
Reporting: Mapped student model (common ontology version)
Authentication Module
Session Generator
Log Valid Session
Authenticator
Authenticate
External Communication Module
Request/receive new SQL-Tutor session
ADAPT2
External communication: mapped student model updates
Knowledge Tree Portal
Invoke SQL KnoT problems
SQL KnoT
Student activity
CUMULATE

Fig. 4. High-level view of the SQL-Tutor Resource Component (STRC)

Each constraint links to one or more concepts from the common SQL ontology. The degree to which each concept is associated with the constraint is modeled by the weight, such that a concept with higher weight has higher relevance in that constraint. Weights are small (1), medium (2), or large (3). On each attempt, the mapping module receives a report of the short-term student model consisting of two sets of constraints: satisfied and violated. A student knowledge score is then calculated for each concept using equation 1 below. The score for each concept ranges from -1 to 1. A score of -1 means that the student violated all the instances of all constraints relating to that particular concept and vice versa for a score of 1. The mapped student model is then sent to the external communications module, for converting it into the CUMULATE report format.

$$Student's\ knowledge\ score\ for\ each\ concept = \frac{sum\ of\ satisfied\ concept\ weights - sum\ of\ violated\ concept\ weights}{sum\ of\ relevant\ concept\ weights} \quad \textbf{(1)}$$

The authentication module contains the session generator and provides the authentication into the STRC. Server-level authentication operates on the belief that user authentication occurs at the external server. This means that anyone using STRC via an authenticated external server is pre-authorized and does not require further validation. This is different from the stand-alone SQL-Tutor version, which provides authentication at the user-level. Before communications with the STRC, an external server (e.g CUMULATE) identifies itself and requests a new session code from the session generator. Using this code and a secret key, the external server begins communications with the external communications module, which, after successful authentication, processes its request.

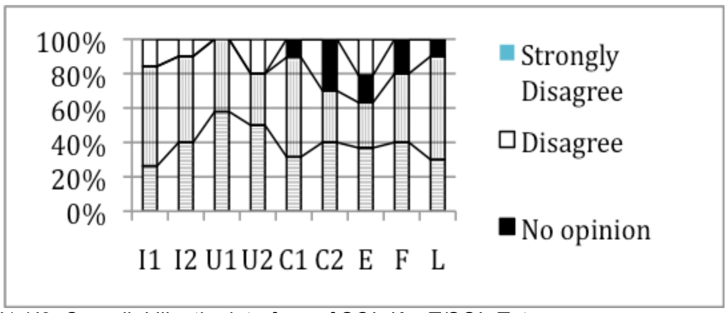
5. Classroom Evaluation of the Integrated Database Exploratorium

We have performed a half a semester study of the developed platform in the context of two introductory database courses at the University of Pittsburgh in the fall, 2008. The native Exploratorium tools (including Knowledge Tree and SQL-KnoT) were available to the student from the beginning of the semester. SQL-Tutor was introduced in the middle of the semester, when students were already studying more advance SQL topics.

Our main goal was do check whether the students actually need the integration of SQL-Tutor into Exploratorium. Although the previous study showed that students appreciated the integrated nature of Exploratorium tools [8], the original set of tools provided them with very different learning activities (problem, examples and labs). At the same time SQL-Tutor, while implementing advanced diagnostics of students' answers and rich problem solving support, did not introduce to them any sufficiently new learning activity comparing to existing and familiar SQL-KnoT.

The results of the study show, out of 42 students who worked with SQL-KnoT 18 tried SQL-Tutor problems. Several students after that switched to using SQL-Tutor, but most of them continued to use both tools. The session analysis show that out of 103 sessions, where students accessed SQL-KnoT system, in 66 they also worked with SQL-Tutor, which means they really used both systems simultaneously.

Besides objective usage parameters we also collected a short questionnaire evaluating students' subjective opinion about different aspects of the integrated systems. 21 students agreed to fill the questionnaire at the end of the semester (9 for graduate students and 12 for undergraduate students). We used the standard Likert scale with five values (from "*strongly agree*" to "*strongly disagree*"). The results are like the following. Fig. 5 demonstrates the result of this questionnaire.



- *I1 / I2: Overall, I like the interface of SQL-KnoT/SQL-Tutor.*
- *U1 / U2: SQL-KnoT/SQL-Tutor is a useful learning tool.*
- *C1 / C2: SQL-KnoT/SQL-Tutor problems challenged me intellectually.*
- *E: SQL-KnoT is generating similar problems with different content and this feature is useful.*
- *F: When you answered wrongly, the feedback provided by SQL-Tutor was helpful to solve the questions correctly.*
- *L: Seeing various levels of feedbacks was important.*

Fig. 5. Results of the subjective evaluation

As we can see from the plot, students valued both systems and appreciated the opportunity to use them both. They also positively responded to the core features of the systems such as question generation by SQL-KnoT and corrective feedback of SQL-Tutor.

6. Summary and Future Work

This paper presents a rather unique project on integrating two Web-based adaptive educational systems. The two systems were originally developed by separate research teams and have been use in real-life educational settings by hundreds of students before we decided to integrate them. The implemented architecture provides one of the first working cases of real-life cross-system personalization in the context of e-Learning. From the student interface point of view we tried to achieve the feeling that the systems are the part of a single learning environment. The students were able to login in both systems using the central learning portal and accessed systems' resources in similar way from the corresponding course folders. From the integration point of view the biggest challenge resulted form the very different principles of domain representation and user modeling employed by the integrated systems. The classroom evolution of the developed platform has shown that students use the systems within single session, which support the need for integration and consistent inter-system user modeling and adaptation. The described project only a first step towards the true integration of AWBES. Our students could work with both SQL-Tutor and SQL-KnoT can be taken into account by our adaptive portal to provide adaptive navigation support for SQL-KnoT problems. However, SQL-Tutor so far does not take into account the modeling information available in CUMULATE. More careful study of the quality of the resulting user models is also required.

References

1. Brusilovsky, P., and Peylo, C. *Adaptive and intelligent Web-based educational systems*. International Journal of Artificial Intelligence in Education **13**(2-4), 159-172.
2. Murray, T., *A Model for Distributed Curriculum on the World Wide Web.* Journal of Interactive Media in Education, 1998.
3. Rowley, K. *The challenge of constructing a Mega-tutor over the Web*. in *Workshop "Intelligent Educational Systems on the World Wide Web" at AI-ED'97, 8th World Conference on Artificial Intelligence in Education*. 1997. Kobe, Japan: ISIR.
4. Koedinger, K.R., D.D. Suthers, and K.D. Forbus, *Component-based construction of a science learning space*, in *4th International Conference on Intelligent Tutoring Systems (ITS'98)*, B.P. Goettl, et al., Editors. 1998, Springer Verlag: Berlin. p. 166-175.
5. Roschelle, J., et al., *Scaleable Integration of Educational Software.* Journal of Interactive Media in Education, 1998. **98**(6).
6. Trella, M., C. Carmona, and R. Conejo. *MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web.* in *Workshop on Adaptive Systems for Web-based Education at 12th International Conference on Artificial Intelligence in Education, AIED'2005*. 2005. Amsterdam: IOS Press.
7. Brusilovsky, P. *KnowledgeTree: A distributed architecture for adaptive e-learning.* in *13th International World Wide Web Conference, WWW 2004 (Alternate track papers and posters)*. 2004. New York, NY: ACM Press.
8. Brusilovsky, P., et al., *An Open Integrated Exploratorium for Database Courses*, in *13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'2008)*. 2008, ACM Press: Madrid, Spain. p. 22-26.
9. Mitrovic, A. and S. Ohlsson, Evaluation of a Constraint-based Tutor for a Database *Language.* Int. J. on Artificial Intelligence in Education, 1999. **10**(3-4): p. 238-256.
10. Brusilovsky, P., Sosnovsky, S., and Shcherbinina, O. 2005. User Modeling in a Distributed E-Learning Architecture. In L. Ardissono, P. Brna & M. A. (eds.), Proceedings of 10th International Conference on User Modeling (UM'2001), Edinburgh, UK (pp. 387-391).
11. Sosnovsky, S., et al. *Towards integration of adaptive educational systems: mapping domain models to ontologies*. in *6th International Workshop on Ontologies and Semantic Web for E-Learning (SWEL'2008) in conjunction with ITS'2008*. 2008. Montreal, Canada.
12. Sosnovsky, S., et al., *Ontology-based integration of adaptive educational systems*, in *16th International Conference on Computers in Education (ICCE'2008)*. 2008: Taipei, Taiwan. p. 11-18.
13. Mitrovic, A., B. Martin, and P. Suraweera, *Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring.* IEEE Intelligent Systems, special issue on Intelligent Educational Systems, 2007. **22**(4): p. 38-45.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0633494.