Semantic Integration of Adaptive Educational Systems

Sergey Sosnovsky[1], Peter Brusilovsky[1], Michael Yudelson[1],
Antonija Mitrovic[2], Moffat Mathews[2], and Amruth Kumar[3]

[1] University of Pittsburgh, School of Information Sciences,
135 North Bellefield Ave. Pittsburgh, PA 15260, USA
[2] University of Canterbury,
Department of Computer Science and Software Engineering
Private Bag 4800, Christchurch 8140, New Zealand
[3] Ramapo College of New Jersey
505 Ramapo Valley Road, Mahwah, NJ 07430-1680, USA
sosnovsky@gmail.com, peterb@pitt.edu, myudelson@gmail.com,
Tanja.Mitrovic@canterbury.ac.nz, moffat@cosc.canterbury.ac.nz,
amruth@ramapo.edu

Abstract. With the growth of adaptive educational systems available to students, integration of these systems is evolving from an interesting research problem into an important practical task. One of the challenges that needs to be addressed is the development of mechanisms for student model integration. The architectural principles and representation technologies employed by adaptive educational systems define the applicability of a particular integration approach. This chapter reviews the existing mechanisms and details one of them: the evidence integration.

Keywords: Adaptive Educational System, Semantic Integration, User Model Interoperability, Ontology.

1   Introduction

Over the last 10 years, a number of adaptive systems have migrated from research labs to real life. Web recommender systems [1], mobile tourist guides [2] and adaptive educational systems (AES) [3] are now employed by thousands of real users. In some application areas, the "density" of practical adaptive systems is reaching the point where several adaptive systems are available. Yet, in most cases, these systems do not compete, but rather complement each other, while offering unique functionality or content. This puts the problem of using several adaptive systems in parallel on the agenda of the user modeling community. This problem has been explored over the last few years by several research teams and from several perspectives: architectures for integrating adaptive systems [4], cross-system personalization [5], [6], user model ontologies [7], [8], and user modeling servers [9], [10], [11].

The main challenge of using several adaptive systems in parallel (or a distributed adaptive system) is making the whole more than the sum of its parts. In this context, it means that each of the systems should have a chance to improve the quality of user

modeling and adaptation based on integrated evidence about the user collected by all participating systems. At this point, the most popular approach to solving problem is *translation* [12] (or *mediation* [13]) from one user model to another. This approach is very attractive if two adaptive systems are used in a sequence, one after another. However, when two adaptive systems have to be used in parallel (i.e., the user models on both sides are being constantly updated within the same session), a translation of the whole user model from one representation to another becomes a relatively costly approach. To account for the combined information about the user, the integrated systems will need to translate the each other's user models before any adaptive decisions can be made.

Good examples of such a scenario are distributed adaptive E-Learning frameworks such as Medea [14] or KnowledgeTree [4], where students can work with educational activities provided by several independent adaptive systems. Each of the involved systems receives evidences about student knowledge and attempts to build the student knowledge model. To make this model reliable, each of the involved systems should take into account evidences produced by the student during his/her work with the systems. Our previous experience with distributed E-Learning systems shows that a student can switch from one system to another many times even within a single session [15]. To avoid multiple translations from one user model to another within the same session, we explored an alternative approach to user modeling in distributed adaptive systems called *evidence integration*. With this approach, adaptive systems do not exchange entire user models, but instead exchange elementary evidences produced as results of the student's actions. In this case, the problem of student model integration is actually a problem of evidence integration. While evidence integration is a relatively simple task in some domains (i.e., user's ratings for a specific movie can be easily taken into account by multiple recommender systems), it is not the case in e-learning. In e-learning, each educational activity (i.e., problem, quiz, or example) is typically described in terms of a system's internal *domain model*. Using this knowledge and the outcomes of student's actions (e.g. correct or incorrect solutions to problems), the user modeling component updates student knowledge model. In a rare case, where the component systems share the same domain model, integrating evidences from two or more adaptive systems is a relatively simple problem [14], [16]. However, in reality, two adaptive systems developed for the same domain (such as Java programming or SQL) can rely on very different domain representations. In that case, evidence integration becomes a difficult task, which requires some kind of translation from one *domain model* to another.

This paper details two practical examples of distributed student modeling using evidence integration. Each example involves two e-learning systems with considerably different domain models for the same subject (Java and SQL languages). One of these examples (Section 3) demonstrates fairly simple and straightforward evidence integration, while another (Section 4) presents a more sophisticated case based on the alignment between two large domain models relying on very different representation formalisms. Taken together, these cases stress the problems of distributed user modeling in the field of e-learning and demonstrate how the evidence integration approach can support conceptual and architectural integration in the context of a real college-level course. To make our example more useful, we preface

it with a discussion of existing integration approaches in the area of e-learning (Section 2) and present the implementation details of our approach (Section 5). We conclude with a summary of our results and a discussion of future work.

2   Existing Integration Approaches

This analysis focuses on a particular aspect of adaptive system integration. Due to the wide spectrum of existing adaptive technologies, there are many ways to integrate user modeling information collected and inferred by adaptive systems. In the field of recommender systems, this task can be transformed into aggregation of user ratings collected by several systems [17], or mediation between content-based and collaborative user models [18]. In the field of pervasive adaptation exploiting rich, multifaceted user profiles, integration of adaptive systems will require matching complex user modeling ontologies [19]. AESs focus on the modeling of student knowledge, which includes representation of the domain structure in terms of its elementary units and estimation of knowledge levels for these units. Hence, we will limit our discussion to the integration of AESs modeling student knowledge. Such integration will require target systems to achieve a certain level of mutual understanding of the domain semantics. Once the systems agree on the domain model, they can exchange student models for the equivalent or related parts of the domain and incorporate them into adaptive inference.

The general task of domain model alignment potentially involves resolution of multiple model discrepancies on two principle levels. The language–level mismatches, such as different syntax, expressiveness, or varying semantics of used primitives, need to be resolved first. However, the more critical are the model-level mismatches that occur due to the difference in structure and/or semantics of the domain models. Resolution of these kinds of discrepancies involves dealing with such problems as:

– Naming conflicts (the same concept is defined in two models by different terms or the same term defines different concepts);
– Different graph structure (the models choose to connect relevant sets of concepts in different ways);
– Different scope (two models cover parts of the domain that only partially intersect or the scope of one model includes that of another model);
– Different granularity (the size of concepts differ across the models; a single concept of one model represents a piece of domain knowledge covered by several concepts in another model);
– Different focus (the models examine different modeling paradigms or adhere to different modeling conventions).

This list does not include the mismatches specific to those formal models employing advanced modeling primitives, such as typed relations and axioms (e.g. the same entity can be modeled as a concept and as an attribute).

The next sections outline several approaches to semantic integration of adaptive educational systems described in the literature.

2.1 Single-Ontology Integration

One of the first steps toward interoperable adaptive systems would be implementation of domain models as ontologies. Ontologies express the shared view on domain semantics and come with a full package of technologies developed within the framework of the Semantic Web initiative. When user models of two systems rely on a common domain ontology, they can be exchanged and consistently interpreted when necessary. The OntoAIMS project provides a good example of such integration [20]. Two components of OntoAims: OWL-OLM [21] and AIMS [22] – were developed as separate systems, but with a mutual concern about interoperability. Both AIMS and OWL-OLM represent their domain models as OWL-ontologies and model user knowledge as ontology overlays. As a result, merging these two systems into an integrated adaptive environment providing a rich learning experience was a straightforward task. The long-term user model in OntoAIMS is shared by both its components. During a session with either AIMS or OWL-OLM, a short-term user model is populated and then used to update the long-term model.

Several research teams have generalized this approach to the level of integrated architectures based on central user modeling servers (e.g. Personis [23], ActiveMath [24], CUMULATE [25]). These servers perform centralized domain and user modeling, and supply this information to the individual adaptive systems. As a result, the adaptive systems themselves do not need to support domain and user modeling. They update the central user model and request the modeling information from the server.

2.2 Central-Ontology Integration

The single-ontology integration can work only if the participating systems fully agree on a single ontology for modeling the domain of discourse. Unfortunately, the practice of AES is still far from the use of common ontologies. Although the designers of AES more and more frequently choose to represent the domain models as ontologies, they tend to employ different ontologies for the same domain.

In some cases, this problem can be remedied without much effort. If domain models of adaptive systems have a common reference ontology, it can facilitate the exchange of modeling information through the "hub" concepts shared by the domain models of both systems. This becomes important in the situation when several small adaptive systems model student knowledge in tightly related domains (or parts of a single domain). A central ontology can act as a meta-translator for the shared concepts and "bootstrap" the user modeling through such concepts. Mitrovic and Devedzic describe such a scenario in [26] and introduce M-OBLIGE – an architecture for centralized exchange of user-modeling information among multiple intelligent tutoring systems acting in related parts of SQL and Relational Algebra.

This scenario still requires a certain level of ontological commitment from the participating systems – their models should rely on the same reference ontology, which is hard to ensure when the systems are designed by different research teams. In general, adaptive systems use completely different ontologies to model student knowledge. These models can still be integrated; however, it requires more effort on both the architectural and conceptual sides. One of the first steps in this direction has

been made in Medea [27]. Medea combines the functionality of an adaptive learning portal that help students navigate through available learning resources and one of a user modeling server that keeps track of student's actions and computes her/his knowledge of course topics. Medea does not host the learning content itself; instead, it provides access to the participating adaptive services. On the modeling side, Medea allows adaptive services to report their local user modeling information into the central user model. The important feature of Medea is the possibility to manually map the domain model of participating services into the central Medea ontology. As a result, the user model updates (received from adaptive services) can be translated into the concepts of Medea's ontology and fused into the central user modeling storage.

2.3 Integration Based on Automatic Ontology Mapping

Both Medea and M-OBLIGE provide practical solutions for semantic integration of multiple AESs into distributed platforms for coherent student modeling and adaptation. However, they both have limitations. The applicability of M-OBLIGE is reduced to those situations where the domain models of participating systems share the references to the central ontology. The approach implemented in Medea relies on manual ontology mapping, which is a time-consuming task that requires a high level of expertise both in knowledge engineering and the domain of discourse.

Using ontologies for domain modeling enables a more general solution for semantic integration of adaptive systems based on automatic ontology mapping [28]. Ontology mapping techniques help to automatically identify matching elements (concepts, relations, axioms) in different ontologies. They rely on a set of technologies from natural language processing, graph theory and information retrieval to discover similar lexical patterns, conceptual sub graphs and statistical regularities in texts accompanying the ontologies.

Once the mapping between the domain ontologies is established it can be used as a translation component for user model mediation. We are not aware of any fully-implemented components based on this approach; however the first step in this direction has been made. Authors of [29] investigate the applicability of automatic ontology mapping for translation between two overlay models of student knowledge based on two different domain ontologies. The practical evaluation shows that automatic ontology mapping results in user model translation, which is statistically close to the best possible translation done by human experts.

2.4 Evidence Integration

Several ontology-based techniques for semantic integration have been discussed; however, many successful adaptive e-learning systems do not employ ontologies for knowledge representation. They implement adaptation and user modeling technologies relying on formalisms that are different from the conceptual networks, which are the core components of ontologies.

Integration of such models is still possible, although is becomes subject to the two major limitations. First, numerous automatic ontology mapping techniques are not applicable for such models, nor can one expect these models to refer to some common upper ontology. Hence, the alignment of underlying domain models of such systems

can only be done manually. Even though, the participating models can be of any kind (as long as they support the general principle of composite domain modeling) we argue that ontologies could still be useful as a common denominator and facilitate future integration.

Second, the differences in modeling principles and inference mechanisms make the coherent merging of user modeling information harder to achieve. Even when the mapping between two domain representations has been established, the consequent translation of user models can result in noisy and inadequate modeling. This becomes critical when the integration of user modeling information is organized as a rare holistic model exchange (e.g. at the end of the learning session). To remedy this problem, the user model exchange should be triggered as soon as the modeling event is observed. In this case, the influence of internal model inference (e.g., a student has learned this) on the objective event (e.g., a student has answered a problem correctly) is reduced and is maximally close to the evidence exchange happening in central user modeling servers. We call such a mechanism evidence integration.

The next sections of this chapter describe two examples of evidence integration of real adaptive E-Learning systems. The first case implements simple, server-side evidence integration, where the integrated models are fairly close and the user model exchange is not intensive. The second case is an example of more complex evidence integration, where a lot of the work is done on the system side and the user model reports to the server are much bigger.

3   Simple Evidence Integration

This section describes an example of simple evidence integration. Two e-learning systems helping students to practice Java, Problets and QuizJET, rely on different domain models. While QuizJET uses Java ontology, Problets model student knowledge in terms of pedagogically-oriented domain elements called learning objectives. There is not much difference between these two domain models, other than a shift in modeling focus, granularity, and scope. Each learning event observed and registered by Problets results in a small knowledge level update of corresponding learning objectives. The integration has been implemented within the framework of ADAPT[2] architecture on the CUMULATE user modeling server. The next three subsections detail the implementation of Problets and QuizJET as well as describe the integration procedure.

3.1   Ramapo College's Problets

Problets (www.problets.org) are problem-solving tutors on introductory programming concepts in C/C++/C#/Java. They present programming problems, grade the student's answer, and provide corrective feedback. Problets sequence problems adaptively [30], and generate feedback messages that include a step-by-step explanation of the correct solution [31]. Students can use Problets for knowledge assessment and self-assessment, as well as for improving their problem-solving skills. Fig. 1 presents the student interface of a Problet on *if/if-else* Statements in Java. The bottom-left panel contains a simple Java program. The students need to evaluate the program and

answer a question presented in the top-left panel. The system presents student's answers in the right-bottom panel, and indicates the correct and incorrect answers by marking them in green, and red correspondingly. The detailed help on how to use the system, submit the answers and read the system's feedback messages can be always opened in the right-top panel of the Problet interface.
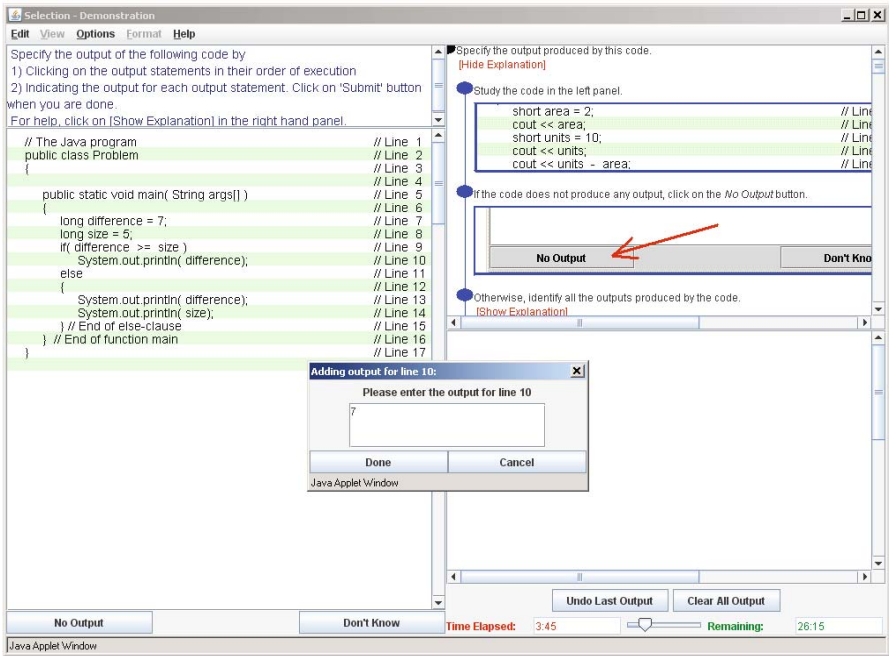

Fig. 1. A Problet on *if/if-else* statements in Java

Problets rely on the concept map of the domain, enhanced with pedagogical concepts called learning objectives, as the overlay student model [32]. Each learning objective is associated with the proficiency level calculated based on the student's answers. The student model provides the basis for adaptive decisions made by the tutor, through associating a proficiency model with each learning objective. The system propagates the proficiency values to the top levels of the concept hierarchy. At any point in the tutoring session, a student can observe the current state of her/his user model. Fig. 2 demonstrates an example of the user model snapshot for the *if/if-else* Statements in Java.

3.2   University of Pittsburgh's QuizJET

QuizJET (Java Evaluation Toolkit) is an online quiz system for Java programming language. It provides authoring and delivery of quiz questions and automatic evaluation of students' answers. A typical question in QuizJET is implemented as a simple Java program. The students need to evaluate the program code and answer a
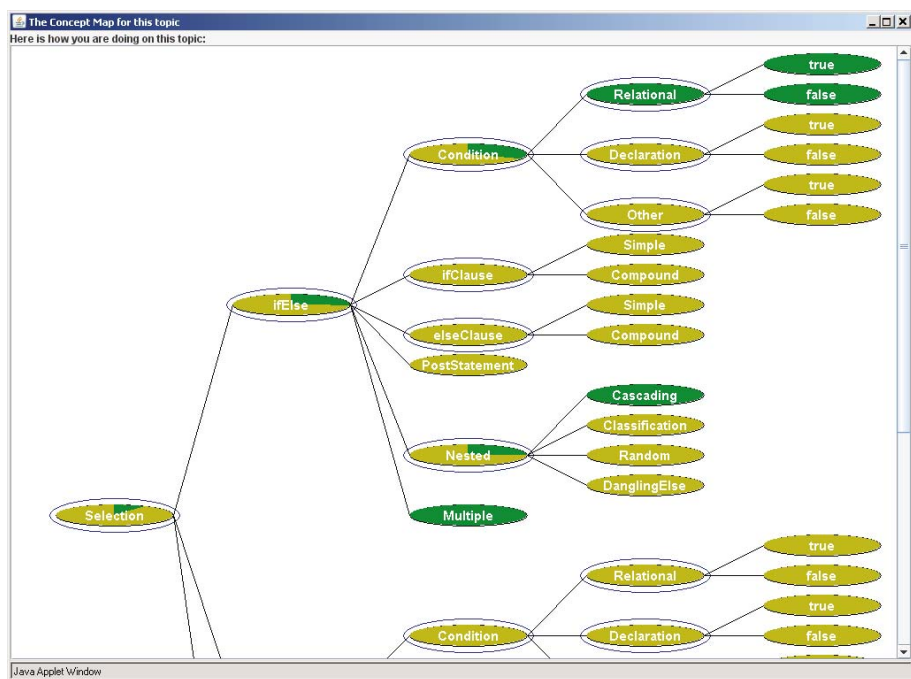
Fig. 2. A part of the domain hierarchy on *if/if-else* statements in Java. Learning objectives are associated with each concept in the hierarchy.

follow-up question, which can take one of two forms: "What will be the final value of the marked variable?" or "What will be printed by the program to the console window?" Upon evaluation of the student's answer, QuizJET provides brief feedback specifying the correctness of the answer and the right answer in case a student has made a mistake.

Fig. 3 demonstrates the student interface of QuizJET. The Java programs constituting QuizJET questions can consist of one or several classes. To switch between classes, QuizJET implements tab-based navigation. The driver class containing the main function (the entry point to the program) is always placed in the first tab, which also presents the question itself, processes the student's input and presents the system's feedback.

The important feature of QuizJET is parameterized questions. One or more numbers in the code of a driver class are dynamically replaced with a random value every time the question is delivered to a student. As a result, the students can practice QuizJET questions multiple times, and every time the question will be different and have a different correct answer.

Every QuizJET question is indexed by a number of concepts from the Java ontology. A concept in a question can play one of two roles: it acts either as a prerequisite for a question (if it is introduced earlier in the course), or as a question outcome (if the concept is first introduced by this question). Fig. 4 presents an extract from the Java ontology.
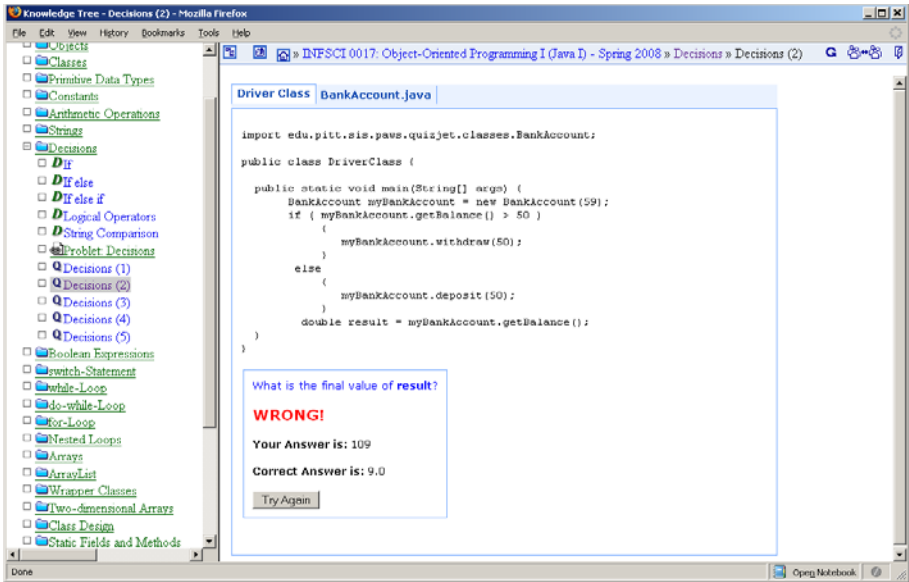
Fig. 3. An example of QuizJET question on Decisions in Java accessed through the Knowledge Tree Learning Portal

3.3   Integration Details

Both Problets and QuizJet questions rely on conceptual content models that provide detailed representation of underlying domain knowledge. In order to maintain consistent interpretation of the evidence reported by these two types of learning content, perform unifying user modeling and implement adaptive mechanisms taking into account a student's work with both systems we need to integrate the underlying domain models on the level of concepts constituting them.

Unlike QuizJET questions that are indexed with the concepts from the same ontology, each Problet relies on a separate model of learning objectives. These models cover six large topics of Java programming language: (1) Arithmetic Expressions, (2) Relational Expressions, (3) Logical Expressions, (4) *if/if-else* Statements, (5) *while* Loops, and (6) *for* Loops.

The combined scope of these topic models is several times more narrow than the one of the Java ontology. At the same time, the granularity of Problets' models is much higher. The total number of concepts in the Java ontology is approximatelly 500; the cumulative number of nodes in the Problets' models is more than 250. The most important problem we had to deal with is the difference in the modeling approaches (or different focus of modeling) used in Java ontology and Problets' domain models. Every learning objective models the application of a concept in a particular learning situation (e.g. different objectives model the simple *if* clause in the *if-else*-statement and the simple *if* clause in the *if*-statement). In other words, a learning objective can be described as a concept put in a context. In order to properly map the
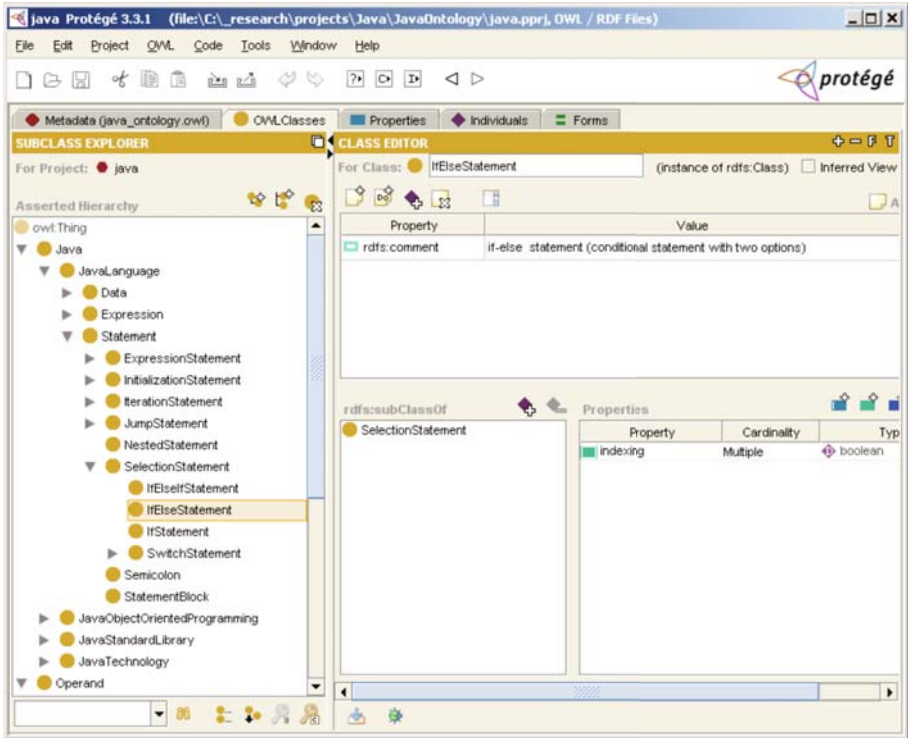
Fig. 4. An extract from the Java ontology

context of a learning objective we often had to connect one learning objective to sev-
eral concepts from the Java ontology. To prevent aggressive evidence propagation to
the concepts modeling context of learning objectives, we also provided weights (from
0 to 1) that define how much knowledge of a particular concept defines the profi-
ciency of the learning objective. An example of mapping a learning objective to con-
cepts is given ion Fig. 5. This terminal-level learning objective from the Selection
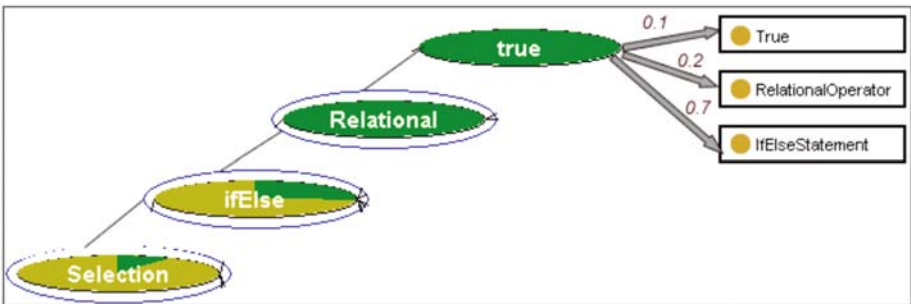

Fig. 5. An example of mapping between learning objectives and ontology concepts

topic defines the application of *if-else* statement, when the condition part of the statement evaluates to *true* value. To properly match this particular situation, we need to use three concepts from the Java Ontology. The assigned weights indicate that the main concept is still *IfElseStatement*, although the evidence of mastering this learning objective will contribute slightly to the knowledge of concepts *RelationalOperator* and *True*. Once this mapping is done for all Problets' learning objectives, any evidence of students' progress reported by any Problet in terms of learning objectives can be interpreted in terms of the ontology-based student model maintained by CU-MULATE and used by QuizJET.

4   Complex Evidence-Based Integration

This section describes a more complex case of evidence-based integration. Two systems implement adaptive support of learning SQL. One of the integrated systems, SQL-Guide, models user knowledge as an overlay of a domain ontology, while the other, SQL-Tutor, employs constraint-based student modeling. While both modeling approaches try to represent elementary knowledge in the domain (with concepts and constraints), the difference between these two models is significant, which results in many-to-many mappings of high modality. Another integration problem occurs due to the fact that learning events in SQL-Tutor trigger knowledge level updates for many constraints. As a result, multiplicative mapping propagations over a number of constraints lead to large user model updates even from a single learning event. The next subsections describe the details of participating systems and overview the implemented integration mechanisms.

4.1 SQL-Tutor and Constrained-Based User Modeling

SQL-Tutor is a constraint-based intelligent tutoring system [33] designed to help students learn SQL. It is part of a family of tools created and maintained by the Intelligent Computer Tutoring Group (ICTG[1]) [34]. SQL-Tutor has been evaluated in twelve studies since 1998 and has been shown to be effective in supporting students' learning.

    SQL-Tutor contains approximately 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 state. It does so by combining its knowledge of the student, the domain (including meta-information about each problem, such as the complexity level), and the implemented teaching strategies. Students have the freedom to ignore the system's suggestions and choose other problems.

    The SQL-Tutor interface is shown in Fig. 6 and contains the problem definition area, the solution workspace, the feedback message pane, controls, and the problem context area. The problem definition area presents the details of the problem (usually in text form). The student enters their solution in the solution workspace. The controls enable the student at any time to submit their solution, request more help, view their student model, execute their query on a real database, and view their session history.

---

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

Fig. 6. The SQL-Tutor interface

The problem context provides information about the problem; the student can view the database schema, information about each relation (including detailed information about all the attributes), and the data in each table. The interface is designed to reduce the working memory load on the student by providing the appropriate information for each problem, while helping the student visualize the current goal structure. This enables students to balance their cognitive load by focusing on learning higher-level query definition problems rather than on checking low-level syntax.

After evaluating a submitted solution and identifying mistakes, SQL-Tutor provides students with adaptive feedback. Students can also request further help from one of the six feedback levels; this includes the option of viewing the ideal solution.

The domain module contains domain knowledge represented 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. 7 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. 7. Two example constraints

The short-term student model in SQL-Tutor consists of the list of relevant, satisfied and violated constraints. The long-term student model consists of the general information about the student. In addition, this model contains the history of usage of each constraint found relevant in submissions made by a particular student. The history is a record of how the constraint was used on each occasion it was relevant. The long-term model also contains an estimate of the student's knowledge of each constraint. This model is used for adaptive problem selection.

4.2 SQL-Guide and SQL Ontology

SQL-Guide is an adaptive hypermedia system helping students to practice SQL skills. A typical SQL-Guide problem description contains a set of predefined databases and a desired output, for which a student is asked to write a matching query (see Fig. 8). The system evaluates the student's answer and provides simple feedback. All problems in SQL-Guide are dynamically generated using a set of parameterized templates. An average template is capable of generating several dozens of unique SQL problems with the predefined level of difficulty and the same set of related concepts.

To assist students in choosing the appropriate problem to practice, SQL-Guide employs an adaptive hypermedia technique called adaptive annotation. Every problem in SQL-Guide is annotated with an adaptive icon reflecting the progress of the student with the learning material underlying this problem. The CUMULATE user modeling server keeps track of all answers the student has given to SQL-Guide's problems and computes the long-term model of student knowledge for the related concepts. SQL-Guide requests the state of the model from CUMULATE and dynamically annotates links to topics and problems with the appropriate icons. The student's progress is double-coded: as the knowledge level grows, the icon fades and the bar level rises. By means of this abstraction, SQL-Guide delivers to a student two kinds of information: where the progress has been made (higher bar level) and where the attention should be focused (brighter target color). The checkmarks over the problem icons designate problems that have been solved correctly at least once. To help a student understand the meaning of annotations, QuizGuide dynamically generates mouse-over hints for all icons. A more detailed description of the system can be found in [35], [36].
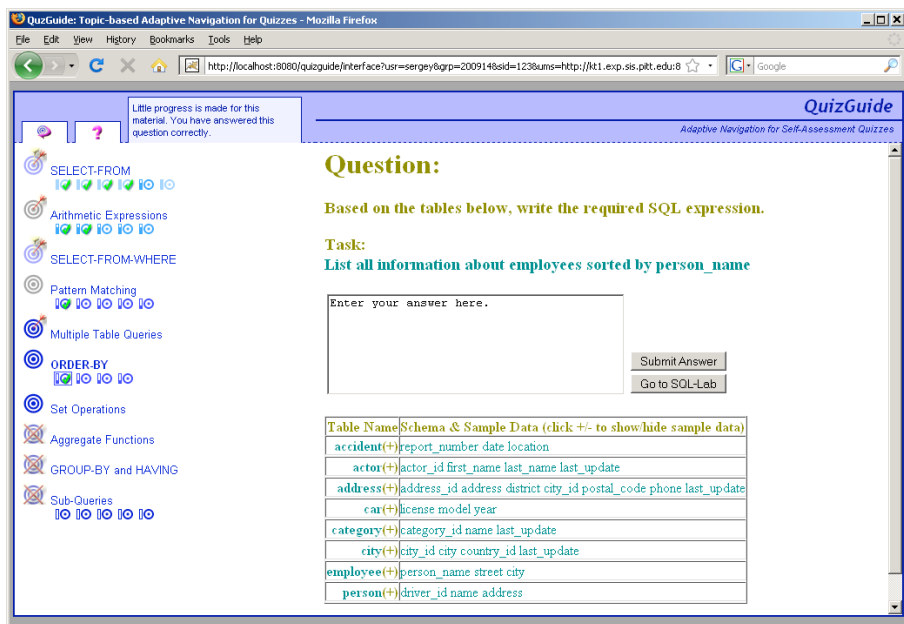
Fig. 8. The interface of SQL-Guide

Every problem template (and, naturally, every problem) in SQL-Guide is indexed with several concepts from the SQL Ontology, which was developed as a collaborative effort between the PAWS Lab of the University of Pittsburgh, and the ICT Group of the University of Canterbury [37]. The main purpose of this ontology is to support the development of adaptive educational content for SQL and facilitate the integration of educational systems in this domain, while ensuring the objective modeling of SQL semantics. 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 than 200 classes connected via three 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. 9 gives some examples of the SQL ontology relations.

The level of granularity of the terminal concept in the SQL ontology was chosen to support the adequate modeling of students' knowledge with the necessary details. At the same time, our goal was not the comprehensive representation of the current SQL standard, therefore certain parts of the domain remain out of the scope of this ontology (Fig. 10).

```
<sql:WhereClause>        <rdfs:subClassOf>    <sql:Clause>
<sql:SelectStatement>    <rdfs:subClassOf>    <sql:Statement>
<sql:WhereClause>        <sql:isUsedIn>       <sql:SelectStatement>
<sql:SelectStatement>    <sql:uses>           <sql:WhereClause>
```
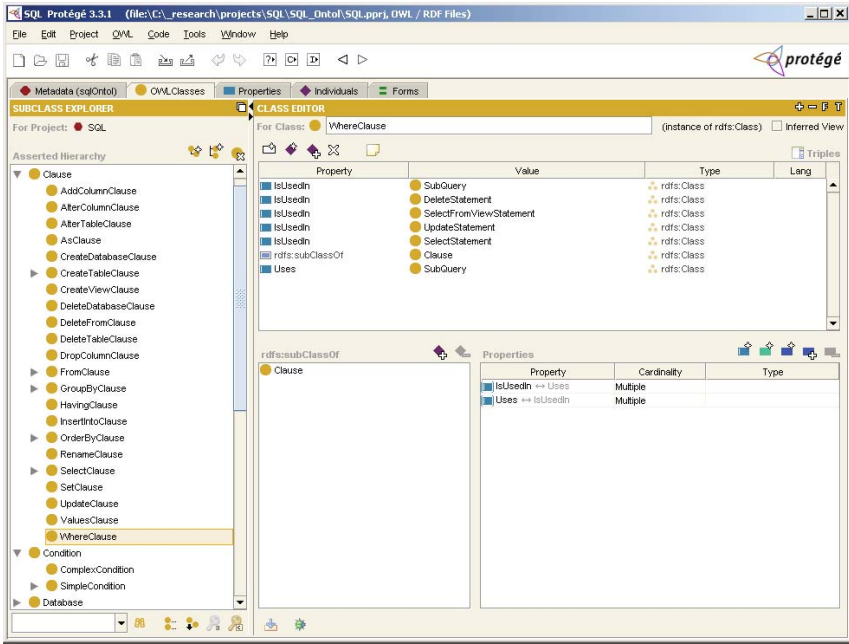
Fig. 9. Example of relations from SQL Ontology

Fig. 10. SQL Ontology

4.3   Integration Details

SQL-Tutor is an independent (stand-alone), web-based, intelligent tutoring system. To use SQL-Tutor within the context of a complex evidence-based integration structure, we created a new system, SQL-Tutor Resource Component (STRC) containing four main modules. We describe the architecture of the new system first, followed by the details of its components and integration.

The SQL-Tutor Resource Component (STRC). The four modules of STRC are shown in Fig. 11 and include SQL-Tutor, the mapping module, the authentication module, and the external communications module. The STRC makes it possible for SQL-Tutor to be used as a teaching resource within the framework of a larger teaching system.

Within the STRC, the core engine and modules of SQL-Tutor are treated as a "black box". A simple internal API allows for basic control requests (for example, requesting a particular problem from SQL-Tutor) while the SQL-Tutor solution evaluator reports student progress.

The Mapping Module. The fundamental differences in the domain models of SQL-Tutor and SQL-Guide 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
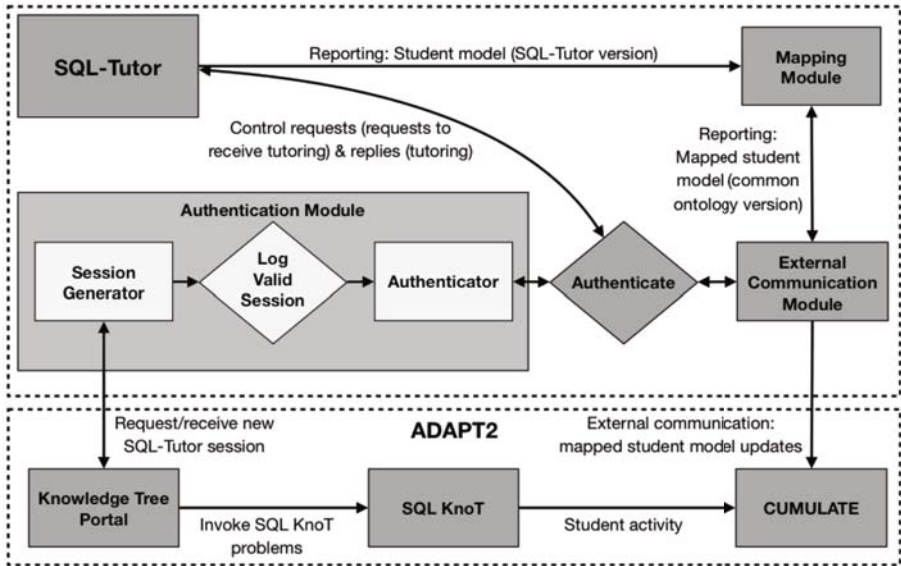
Fig. 11. High-level view of the SQL-Tutor Resource Component (STRC)

syntactic or semantic relations between various concepts. The development of the algorithm even for partial resolution of the modeling discrepancies between ontologies and constraint-based models is not a trivial task.

SQL-Tutor models students' knowledge in terms of constraints. When a student submits his/her solution, SQL-Tutor evaluates it and reports on the correctness of the submission as a set of satisfied and violated constraints (i.e. the short-term student model). Feedback on the student's solution is displayed directly in the student's browser while the report is sent to the mapping module (see Fig. 11).

The purpose of the mapping module is to take the short-term student model and convert it to a report based on a common ontology used by a particular external server. The mapping module therefore consists of the mappings between constraints and the common ontology, a student knowledge score calculator, and functions to convert the SQL-Tutor report to the mapped report.

The Mapping. Each constraint links to one or more concepts from the common SQL ontology. The degree to which each concept is associated to the constraint is called the weight, such that a concept with a higher weight has higher relevance in that constraint. Weights are small (1), medium (2), or large (3). Domain experts manually created the ontology while an expert in both SQL and Constraint-based Modeling (CBM) manually created the mapping. For more detail on the process used to derive the mapping, please refer to [37], [38].

Fig. 12 shows a part of the mapping, which is implemented as a list of lists. Each list contains a constraint ID followed by one or more concept/weight lists. For example, constraint 705 maps to two concepts, the *CommaCharacter* (with weight 1) and the *OrderByClause* (with weight 2).

```
Mapping
(...  (2   ("SelectClause" 3))
      (77  ("OrderByClause" 2)("SubQuery" 2))
      (80  ("OrderByClause" 1)("SelectClause" 1))
      (84  ("OrderByClause" 1))
      (525 ("CommaCharacter" 1)("SelectClause" 2))
      (803 ("OrderByClause" 1)("GroupByClause" 1))
      (705 ("CommaCharacter" 1)("OrderByClause" 2))

      (192 ("Column" 3)("SelectClause" 3)("FromClause" 1))
      (673 ("SelectClause" 1)("AsteriskCharacter" 2)) ...)
```

Fig. 12. Part of the mapping found in the mapping module

Calculating the Student's Evidence of Knowledge (Knowledge Score). On each attempt, the mapping module receives a report of the short-term student model consisting of two sets of constraints: satisfied and violated. Two sets of concepts (satisfied and violated) are created by parsing the sets of constraints through the mapping described above. As with constraints, the same concept can appear multiple times in both sets depending on the context in which they were satisfied or 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.

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

The mapped student model (the report of mapped concepts with associated calculated knowledge scores) is then sent to the external communications module, which is converted into the right format before sending it to CUMULATE user modeling server. Fig. 13 shows a part of SQL-Tutor student model report, containing the two lists of satisfied and violated constraint ID numbers. To keep the example uncluttered, only a small portion of each list is shown.

Satisfied constraints (partial list)
(... 2 77 80 84 525 705 803 ...)
Violated constraints (partial list)
(... 192 673 ...)

Fig. 13. Part of the student model showing the satisfied and violated constraints

Using the mapping (Fig. 12) and equation 1, a knowledge score is calculated for each concept. In our example, the *OrderByClause* has a knowledge score of 6/6, i.e. 1, as all constraints relating to it were satisfied. On the other hand, the *SelectClause* has a knowledge score of (6-4)/10 i.e. 0.2. The list of concepts and related knowledge scores form the mapped short-term student model. This information is then sent to CUMULATE, which integrates it with the global student model, as described in Section 5.3.

The Authentication Module. The authentication module contains the session generator and the authenticator and provides basic authentication at the server level to 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 already 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.

The purpose of this module is:

- to correctly identify and recognize the external server. This allows the STRC to adapt to the needs of each external server. This includes the particular communication protocols agreed upon between STRC and the external server, inclusion of specific information about each student (relevant to the external server), and potentially even the type of mapping (e.g. mapping to a different common ontology).
- to correctly identify and recognize each student. A username is unique within the domain of each external server. Recognizing each individual student is an essential part of providing customized content.
- to provide basic security. Unauthorized tampering with an educational system could significantly reduce its tutoring performance.

The External Communications Module. The external communications module is responsible for all communications (apart from the session code request) between the STRC and external servers. Communications adhere to the agreed-upon protocols defined within this module. This module also converts generated reports (such as the mapped student model reports) to the appropriate format for each external server. This allows STRC to be connected to multiple external servers.

5   ADAPT^2 and Knowledge Integration in CUMULATE

In this section, we describe the ADAPT^2 architecture that hosts all of the applications discussed above and provides the means for their integration. Special attention is given to CUMULATE – a centralized user modeling server. We explain how user knowledge is computed and integrated.

5.1   ADAPT^2 – Architecture for Semantic Integration of Adaptive Educational Systems

ADAPT^2 (read adapt-square; stands for Advanced Distributed Architecture for Personalized Teaching & Training [39]) is an extension of the earlier KnowledgeTree architecture [4]. ADAPT^2 provides a general framework for organizing multiple adaptive and non-adaptive educational tools into a distributed learning environment. The four main types of components in this framework are:

– Learning Portal, which organizes the learning material and provides students and teachers with the fanctionality necessary for participating in learning process;
– User Modeling Server, which stores students' activity and infers information about their characteristics;
– Activity Server, which implements one or more kinds of learning activities in either an adaptive or non-adaptive manner;
– Value-added Service, which adds some additional capabilities to the raw content provided by Activities Servers, e.g. it can provide adaptive navigation support or add annotation mash-up, etc.

All applications described in this paper act as components of ADAPT^2. Several of them have been developed for ADAPT^2 specifically (SQL-Guide, QuizJET) and are able to submit learning evidences to CUMULATE and request user model reports from it. Others have been enhanced to make them compatible with ADAPT^2 (Problets, SQL Tutor) by implementing ADAPT^2 authentication and event-reporting components. From the student perspective, all applications are accessible through the single entry point – the Knowledge Tree portal. Knowledge Tree employs a folder-document paradigm and is a link level aggregator for a variety of educational resources. Knowledge Tree provides authentication, authorization, and access to the resources. The conceptual integration of the components is provided by the CUMULATE server as described in the next section.

5.2   Student Modeling with CUMULATE

CUMULATE [9] is a second-generation user modeling server developed for ADAPT^2. CUMULATE accepts reports of user activity from ADAPT^2 systems and infers overlay user knowledge model for a related domain. CUMULATE maintains awareness about users and educational content by storing and/or caching several types of information: user identities and credentials, user memberships in groups (classes), identities of the resources, with which ADAPT^2 users interact, domain ontologies with concept hierarchies, and resource-concept metadata indices.

CUMULATE accepts and processes two kinds of activity reports. For the learning activities with a fixed set of domain concepts, CUMULATE can accept *brief event reports*, which mention only user, group, and resource IDs. For processing brief reports, the ontological metadata for the application's resources needs to be known in advance. CUMULATE caches the resource metadata and uses it to determine the activated domain concepts as soon as the evidence of user activity with a particular resource arrives. For the dynamic learning resources with mutable sets of concepts (can be different for different attempts), CUMULATE requires *extended reports*, which include a full set of activated domain concepts. In addition, CUMULATE keeps per-resource progress measures, tracking user advancement in working with a particular problem or exercise.

Following each positive activity report, which provides evidence of student knowledge, CUMULATE updates the state of all confirmed concepts related to the activity. For the brief reports, CUMULATE performs a cache lookup to determine activated concepts and then updates knowledge of the determined concepts. For the extended reports, the respective knowledge is updated directly.

To update user knowledge based on evidences received through activity reports, CUMULATE applies a specific inference mechanism that builds upon the paradigm of power-law learning. The idea of this approach is that with every successful attempts to apply a certain concept, the increment of actual user knowledge is diminishing, asymptotically approaching 100%. The specific version of this approach implemented in CUMULATE was designed to meet the following guidelines:

– Knowledge of a concept is updated with every successful solution to the problem involving this concept. There is no knowledge decay or punishment applied for incorrect answers.
– Knowledge level updates for an activated concept are directly related to the weight between the concept and the solved problem. This update is inversely related to the sum of weights of all activated concepts.
– Knowledge level updates for an activated concept are inversely related to the number of successful attempts for a particular problem. It was designed to encourage users to access different problems instead of trying to increase their knowledge by solving just one.

The current state of user knowledge represented by CUMULATE can be requested by any ADAPT[2] component. These requests can be general; for example, a snapshot of a full domain model can be acquired. Or they can be specific, requesting only a limited subset of the user model. The format of the reports can be plain text, XML, or Java Objects.

5.3 Knowledge Integration in CUMULATE

Evidence information can be processed by CUMULATE in three different ways. The student action reports received from QuizJET or SQL-Guide problems (which are ADAPT[2] native applications) go through a straightforward knowledge modeling cycle. They are combined with the ontology-based metadata to identify the activated concepts and then the knowledge levels for these concepts are recalculated based on the modeling formula taking into account the status of the report (success/failure), the concept weights (from metadata entries) and the historical information (previous knowledge for a concept, number of successful attempts for a problem). The propagation path of user modeling information for this case is shown in Fig. 14.

In the case of Problets, the learning objectives are mapped to ADAPT[2]'s Java ontology, i.e., each Problet's objectives are related to some ontology concept. Instead of reporting problem solving events, Problets report the evidence in terms of learning objectives. Each time a learning objective is evaluated, Problets report it to CUMULATE using brief reports and indicating the unique ID of a learning objective. To process this evidence, CUMULATE registers every learning objective as a "virtual problem". Hence, the mapping is captured in relations between the virtual exercises and the mapped ontology concepts. The strength of the relation between a learning objective and an ontology concept is denoted by a weight. When a new report of a user's work with Problets arrives, CUMULATE identifies activated concepts and performs a knowledge update in the same way as for ADAPT[2] native applications.
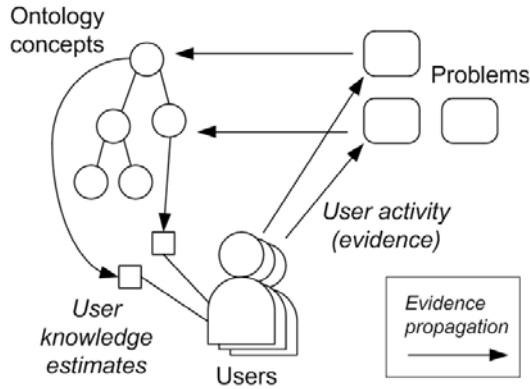
Fig. 14. Propagating reports of user activity (evidence) in CUMULATE

SQL-Tutor also has its own domain model. However, unlike Problets, each problem solving attempt in SQL-Tutor results in dozens of new evidences about learned or violated constraints. Hence, the simple conversion on the CUMULATE side, as implemented for Problets, is not feasible. Instead, a dedicated conversion component is developed on the SQL-Tutor side (refer to Section 4.3 for details). SQL-Tutor uses the extended report protocol to augment the simple evidence information (whether or not the problem has been solved correctly) with a list of activated SQL concepts from the ADAPT[2] SQL ontology. Along with concepts it also reports values between -1 and 1. These values signify the change in knowledge levels computed based on the constraints activated inside SQL-Tutor and the strength of relations between these constraints and ontology concepts. The negative values and corresponding concepts designate student's mistakes and are ignored by CUMULATE, since it does not allow negative evidence propagation. The knowledge levels for the filtered list of concepts are updated using the same modeling formula.

6   Discussion

In this paper, we discussed several ways of integrating adaptive learning systems, focusing on evidence integration, a lightweight solution for integration of user modeling information collected by different educational systems. The resulting infrastructure allows two applications developed by different research teams and relying on considerably different domain models to be used by students in the same course. The applications separately collect the evidence about student knowledge and communicate it to the user modeling server, which allows to maintain more holistic user models.

Our approach is based on manual mapping of domain models and timely evidence reports for user modeling. It was implemented within ADAPT[2] architecture, which supports distributed adaptive e-learning systems. While ADAPT[2] was originally developed for distributed student modeling based on the same domain model, the flexibility of our user modeling server CUMULATE allowed us to deal with a more general scenario involving essentially different domain and student models. The

exploration of this approach in two different case studies, featuring Problets and SQL-Tutor allowed us to distill and generalize this approach and argue that it has a broader applicability for the cases where two applications with different domain models have to work with and model the student simultaneously.

Currently, evidence integration is uni-directional: e.g., information about the student's problem solving results within SQL-Tutor is mapped to the target domain model first (converted from constraints to the ontological concepts) and propagated to CUMULATE, which integrates it with other evidence within the global student model. In future work, we plan to develop a "reverse" mechanism for integrating evidence from other components of ADAPT[2] with the local student models maintained by SQL-Tutor.

While the necessity for manual domain model mapping could be considered a shortcoming of cross-model integration approach, we believe that it is necessary in the case of conceptually different model. At the same time, if both domain models are implemented as Semantic Web ontologies, a good quality mapping can be obtained using automatic mapping techniques. In our recent study, this approach was applied for translation between the student knowledge models of related parts of C and Java programming languages. The C and Java ontologies were developed by different research teams and differed significantly in concept naming conventions and granularity. Student knowledge about a small subset of Java and C concepts were evaluated using several quizzes. The resulting models were compared based on the manual mapping provided by a human expert and the mapping produced automatically by an ontology mapping algorithm. The results of that experiment show that the automatic mapping can generate a user model translation, which is not statistically different from the translation done by a human expert [29].

One of the questions, which require further investigation, is the quality of user models obtained in the process of this multi-system modeling. We argue that our solution based on domain model mapping, while introducing some noise, can result in better student modeling than if we simply ignore a stream of evidence coming from a system with a different user model. At the moment, we are running a multi-semester user study to evaluate the quality of multi-system user modeling using predictive validity and other approaches to user model evaluation.

We are also planning to explore the evidence-based integration approach with other adaptive systems, such as University of Malaga's SIETTE [40] and Trinity College's APeLS [41].

References

1. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007)
2. Krüger, A., Baus, J., Heckmann, D., Kruppa, M., Wasinger, R.: Adaptive mobile guides. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 521–549. Springer, Heidelberg (2007)
3. Brusilovsky, P., Peylo, C.: Adaptive and intelligent Web-based educational systems. International Journal of Artificial Intelligence in Education 13(2-4), 159–172 (2003)

4. Brusilovsky, P.: KnowledgeTree: A distributed architecture for adaptive e-learning. In: Proceedings of 13th International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), New York, NY, May 17-22, pp. 104–113 (2004)

5. Niederée, C., Stewart, A., Mehta, B., Hemmje, M.: A Multi-Dimensional, Unified User Model for Cross-System Personalization. In: Proceedings of Workshop on Environments for Personalized Information Access at AVI 2004, Gallipoli, Italy, pp. 34–54 (2004), http://www.di.uniba.it/avi2004/e4pia/EPIA2004_proceedings.pdf

6. Carmagnola, F., Dimitrova, V.: An Evidence-Based Approach to Handle Semantic Heterogeneity in Interoperable Distributed User Models. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 73–82. Springer, Heidelberg (2008)

7. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo - The General User Model Ontology. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 428–432. Springer, Heidelberg (2005)

8. Dolog, P., Nejdl, W.: Semantic Web Technologies for the Adaptive Web. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 697–719. Springer, Heidelberg (2007)

9. Yudelson, M., Brusilovsky, P., Zadorozhny, V.: A User Modeling Server for Contemporary Adaptive Hypermedia: An Evaluation of Push Approach to Evidence Propagation. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 27–36. Springer, Heidelberg (2007)

10. Kobsa, A., Fink, J.: An LDAP-based User Modeling Server and its Evaluation. User Modeling and User-Adapted Interaction 16(2), 129–169 (2006)

11. Kay, J., Kummerfeld, B., Lauder, P.: Personis: A server for user modeling. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 203–212. Springer, Heidelberg (2002)

12. Sosnovsky, S., Dolog, P., Henze, N., Brusilovsky, P., Nejdl, W.: Translation of overlay models of student knowledge for relative domains based on domain ontology mapping. In: Luckin, R., Koedinger, K.R., Greer, J. (eds.) Proceedings of 13th International Conference on Artificial Intelligent in Education, AI-ED 2007, Marina Del Rey, CA, July 9-13, pp. 289–296 (2007)

13. Berkovsky, S., Kuflik, T., Ricci, F.: Cross-technique mediation of user models. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 21–30. Springer, Heidelberg (2006)

14. Trella, M., Carmona, C., Conejo, R.: MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web. In: Proceedings of Workshop on Adaptive Systems for Web-based Education at 12th International Conference on Artificial Intelligence in Education, AIED 2005, Amsterdam, July 18, pp. 27–34 (2005)

15. Brusilovsky, P., Sosnovsky, S., Lee, D.H., Yudelson, M.V., Zadorozhny, V., Zhou, X.: An open integrated exploratorium for database courses. In: Proceedings of 13th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008, Madrid, Spain, June 30-July 2, pp. 22–26 (2008)

16. Denaux, R., Dimitrova, V., Aroyo, L.: Integrating Open User Modeling and Learning Content Management for the Semantic Web. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 9–18. Springer, Heidelberg (2005)

17. Berkovsky, S.: Decentralized Mediation of User Models for a Better Personalization. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 404–408. Springer, Heidelberg (2006)

18. Berkovsky, S., Kuflik, T., Ricci, F.: Cross-Technique Mediation of User Models. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 21–30. Springer, Heidelberg (2006)

19. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo – The General User Model Ontology. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 428–432. Springer, Heidelberg (2005)

20. Denaux, R., Dimitrova, V., Aroyo, L.: Integrating Open User Modeling and Learning Content Management for the Semantic Web. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 9–18. Springer, Heidelberg (2005)

21. Denaux, R., Aroyo, L., Dimitrova, V.: OWL-OLM: Interactive Ontology-based Elicitation of User Models. In: Proceedings of Workshop on Personalisation for the Semantic Web (PerSWeb 2005) at UM 2005, Edinburgh, UK, July 23-29 (2005), http://www.win.tue.nl/persweb/full-proceedings.pdf

22. Aroyo, L., Dicheva, D.: AIMS: Learning and Teaching Support for WWW-based Education. International Journal for Continuing Engineering Education and Life-long Learning (IJCEELL) 11(1/2), 152–164

23. Kay, J., Kummerfeld, R.J., Lauder, P.: Personis: A Server for User Models. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 203–212. Springer, Heidelberg (2002)

24. Melis, E., Goguadze, G., Homik, M., Libbrecht, P., Ullrich, C., Winterstein, S.: Semantic-Aware Components and Services of ActiveMath. British Journal of Educational Technology 37(3), 405–423 (2006)

25. Brusilovsky, P., Sosnovsky, S., Shcherbinina, O.: User Modeling in a Distributed E-Learning Architecture. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 387–391. Springer, Heidelberg (2005)

26. Mitrovic, A., Devedzic, V.: A Model of Multitutor Ontology-based Learning Environments. Continuing Engineering Education and Life-Long Learning 14(3), 229–245 (2004)

27. Trella, M., Carmona, C., Conejo, R.: MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web. In: Proceedings of Workshop on Adaptive Systems for Web-Based Education: Tools and Reusability at AIED 2005, Amsterdam, The Netherlands, pp. 27–34 (2005)

28. Kalfoglou, Y., Schorelmmer, M.: Ontology Mapping: the State of the Art. The Knowledge Engineering Review 18(1), 1–31 (2003)

29. Sosnovsky, S., Dolog, P., Henze, N., Brusilovsky, P., Nejdl, W.: Translation of Overlay Models of Student Knowledge for Relative Domains Based on Domain Ontology Mapping. In: Luckin, R., Koedinger, K.R., Greer, J. (eds.) Proceedings of 13th International Conference on Artificial Intelligence in Education (AIED 2007), Marina Del Ray, CA, USA, July 9-13, pp. 289–296 (2007)

30. Kumar, A.N.: A Scalable Solution for Adaptive Problem Sequencing and its Evaluation. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 161–171. Springer, Heidelberg (2006)

31. Kumar, A.: Explanation of step-by-step execution as feedback for problems on program analysis, and its generation in model-based problem-solving tutors. Technology, Instruction, Cognition and Learning 3 (2006) (in press)

32. Kumar, A.N.: Using Enhanced Concept Map for Student Modeling in a Model-Based Programming Tutor. In: Proceedings of International FLAIRS conference on Artificial Intelligence, Melbourne Beach, FL, May 11-13 (2006)

33. Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-based Tutor for a Database Language. Int. J. on Artificial Intelligence in Education 10(3-4), 238–256 (1999)

34. Mitrovic, A., Martin, B., Suraweera, P.: Intelligent tutors for all: Constraint-based model-ing methodology, systems and authoring. IEEE Intelligent Systems, special issue on Intel-ligent Educational Systems 22(4), 38–45 (2007)

35. Brusilovsky, P., Sosnovsky, S., Lee, D.H., Yudelson, M., Zadorozhny, V., Zhou, X.: An Open Integrated Exploratorium for Database Courses. In: Proceedings of 13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008), Madrid, Spain, June 30 - July 2, pp. 22–26 (2008)

36. Sosnovsky, S., Brusilovsky, P., Lee, D.H., Zadorozhny, V., Zhou, X.: Re-assessing the Value of Adaptive Navigation Support in E-Learning Context. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 193–203. Springer, Heidelberg (2008)

37. Sosnovsky, S., Mitrovic, A., Lee, D.H., Brusilovsky, P., Yudelson, M.: Ontology-based in-tegration of adaptive educational systems. In: Proceedings of 16th International Confer-ence on Computers in Education (ICCE 2008), Taipei, Taiwan, October 27-31, pp. 11–18 (2008)

38. Sosnovsky, S., Mitrovic, A., Lee, D.H., Brusilovsky, P., Yudelson, M., Brusilovsky, V., et al.: Towards Integration of Adaptive Educational Systems: Mapping Domain Models to Ontologies. In: Dicheva, D., Harrer, A., Mizoguchi, R. (eds.) Proceedings of 6th Interna-tional Workshop on Ontologies and Semantic Web for E-Learning (SWEL 2008) at ITS 2008, Montreal, Canada, June 23 (2008),
http://compsci.wssu.edu/iis/swel/SWEL08/Papers/Sosnovsky.pdf

39. Brusilovsky, P., Sosnovsky, S., Yudelson, M.: Ontology-based framework for user model interoperability in distributed learning environments. In: Richards, G. (ed.) Proceedings of World Conference on E-Learning, E-Learn 2005, Vancouver, Canada, October 24-28, pp. 2851–2855 (2005)

40. Conejo, R., Guzman, E., Millán, E.: SIETTE: A Web-based tool for adaptive teaching. In-ternational Journal of Artificial Intelligence in Education 14(1), 29–61 (2004)

41. Conlan, O., Wade, V., Bruen, C., Gargan, M.: Multi-model, metadata-driven approach to adaptive hypermedia services for personalized eLearning. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 100–111. Springer, Heidelberg (2002)