DB-suite: Experiences with Three Intelligent, Web-based Database Tutors

ANTONIJA MITROVIC
PRAMUDITHA SURAWEERA
BRENT MARTIN
AMALI WEERASINGHE

Intelligent Computer Tutoring Group
Computer Science Department
University of Canterbury
Private Bag 4800, Christchurch, New Zealand
http://www.cosc.canterbury.ac.nz/~tanja/ictg,html

Abstract: E-learning is becoming more and more popular with the widespread use of computers and the Internet in educational institutions. Current e-learning courses are nearly always developed using course management systems (CMS), such as WebCT or Blackboard. Although CMS tools provide support for some administrative tasks and enable instructors to provide online instructional material, they offer no deep support for learning: students have access to on-line material, simple multi-choice quizzes and chat tools, but there is no ability to track student's progress and adapt the learning material and instructional session to the individual student. In this paper we present our experiences with three Web-based intelligent tutoring systems in the area of databases. SQL-Tutor teaches the SQL query language, NORMIT is a data normalization tutor, and KERMIT teaches conceptual database modelling using the Entity-Relationship data model. All three tutors in DB-suite have been used and evaluated in the context of genuine teaching activities. We present the most important features of these systems, as well as evaluation results. The DB-suite tutors have proved to be very effective in supporting deep learning, and are well liked by students.

Keywords: Web-based intelligent tutoring systems, computational intelligence in learning and authoring tools, student modelling in Web-based education, evaluation of intelligent Web-based teaching and learning systems

INTRODUCTION

Intelligent Tutoring Systems (ITS) have been proven to be very effective in domains that require extensive practice (Corbett et al., 1998; Koedinger et al., 1997; Mitrovic & Ohlsson, 1999). In this paper, we present DB-suite, consisting of three Web-enabled ITSs

that teach various database skills to university students. Databases are ubiquitous in today's information systems. Our tutors are Web-enabled, and thus are classroom and platform independent (Vasilakos et al., 2004). The most mature of the three systems is SQL-Tutor (Mitrovic, 1998a; 1998b; Mitrovic & Ohlsson, 1999; Mitrovic et al., 2001), an ITS that teaches the SQL query language. KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) (Suraweera & Mitrovic, 2001) teaches conceptual database modelling, while NORMIT (NORMalization Intelligent Tutor) teaches database normalization (Mitrovic, 2003). All three tutors comprising DB-suite are problem-solving environments, where the system presents problems to solve and offers adaptive problem-solving support and feedback.

The DB-suite tutors are based on Constraint-Based Modeling (CBM) (Ohlsson 1994). The Intelligent Computer Tutoring Group (ICTG) has also developed other constraint-based tutors: for example, CAPIT (Mayo & Mitrovic, 2001) is a MS Windows-based, standalone tutor that teaches punctuation and capitalization rules in English, and LBITS (Martin & Mitrovic, 2002b) teaches vocabulary skills to elementary school children. Based on our experiences developing these tutors, we have also implemented WETAS (Martin & Mitrovic, 2002a; 2003), an authoring shell for developing constraint-based tutors. WETAS is now being used for developing new tutors, including a Web-enabled version of KERMIT.

We start by briefly describing CBM and our database tutors. The following three sections are devoted to SQL-Tutor, KERMIT and NORMIT respectively. The effectiveness and the students' perception of DB-suite tutors were evaluated in several empirical evaluation studies. We present these studies, which demonstrate the effectiveness of the systems for student's learning. Finally, we present the conclusions and directions for future work.

CONSTRAINT-BASED TUTORS

Intelligent tutoring systems are developed with the goal of automating one-to-one human tutoring, which is the most effective mode of teaching (Bloom, 1984). ITS offer greater flexibility in contrast to non-intelligent software tutors since they can adapt to each individual student. Although ITSs have been proven to be effective in a number of domains, the number of ITSs used in real courses is still extremely small (Mitrovic, Martin & Mayo, 2002). Our goal when developing DB-suite was twofold: to provide our students with a flexible learning environment that will adapt to their needs, and to develop a powerful methodology for developing constraint-based tutors. Our methodology is based on Ohlsson's (1996) theory of learning from performance errors.

The typical architecture of constraint-based tutors is given in Figure 1. The tutors are developed in AllegroServe Web server, an extensible server provided with Allegro Common Lisp. All student models are kept on the server. At the beginning of interaction, a student is required to enter his/her name, which is necessary in order to establish a session. The session manager requires the student modeller to retrieve the model for the student, if there is one, or to create a new model for a new student. DB-suite tutors identify the students by their login name, which is embedded in a hidden tag of HTML forms. All student actions are sent to the session manager, to be linked to the appropriate session and stored in the student's log. The action is then sent to the pedagogical module

(PM). If the submitted action is a solution to the current step, the PM sends it to the student modeller, which diagnoses the solution, updates the student model and sends the result of the diagnosis back to the PM, which generates feedback.

SQL-Tutor and NORMIT are Web-enabled tutors with a centralized architecture, with all tutoring functions performed on the server side. In these two domains, solutions produced by students are textual, and the amount of information to be sent to the server is small, so that the centralized architecture is suitable. In KERMIT, students draw diagrams, and some tutoring functions related to drawing are performed on the client side. The tutoring functions are therefore distributed between the server and the Java applet, as described later.

Domain knowledge consists of a set of constraints. Constraint-Based Modeling (CBM) (Ohlsson, 1994; Mitrovic & Ohlsson, 1999) is a student modeling approach that is not interested in the exact sequence of states in the problem space the student has traversed, but in what state he/she is in currently. As long as the student never reaches a state that is known to be wrong, they are free to perform whatever actions they please. The domain model is a collection of state descriptions of the form: *If <relevance condition> is true, then <satisfaction condition> had better also be true, otherwise something has gone wrong*.

The knowledge base consists of constraints used for testing the student's solution for syntax errors and comparing it against the system's ideal solution to find semantic errors. The knowledge base enables the tutor to identify student solutions that are identical to the system's ideal solution. More importantly, this knowledge also enables the system to identify valid alternative solutions, i.e. solutions that are correct but not identical to the system's solution. Each constraint specifies a fundamental property of a domain that must be satisfied by all solutions. Constraints are problem-independent and modular, and therefore easy to evaluate. They are written in Lisp, and can contain built-in functions as well as domain-specific ones. For examples of constraints, please see (Mitrovic, 1998a, 2002; 2003; Suraweera & Mitrovic, 2001; 2002; Martin & Mitrovic, 2003; Mitrovic, Koedinger & Martin, 2003). If the satisfaction condition of a relevant constraint is met by the student solution, the solution is correct. In the opposite case, the student will be given feedback on errors.

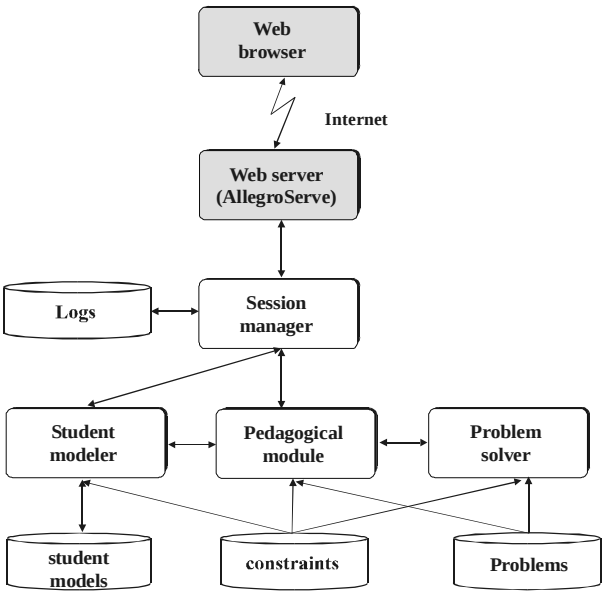

Fig. 1. The architecture of DB-suite tutors

One of the advantages of CBM over other student modeling approaches (Mitrovic, Koedinger & Martin, 2003) is its independence from the problem-solving strategy employed by the student. CBM models students' evaluative, rather than generative knowledge and therefore does not attempt to induce the student's problem-solving strategy. CBM does not require an executable domain model, and is applicable in situations in which such a model would be difficult to construct (such as database design or SQL query generation). Furthermore, CBM eliminates the need for bug libraries, i.e. collections of typical errors made by students. On the contrary, CBM focuses on correct knowledge only. If a student performs an incorrect action, that action will violate some constraints. Therefore, a CBM-based tutor can react to misconceptions although it does not represent them explicitly. A violated constraint means that student's knowledge is incomplete/incorrect, and the system can respond by generating an appropriate feedback message. Feedback messages are attached to the constraints, and they explain the general principle violated by the student's actions. Feedback can be made very detailed, by instantiating parts of it according to the student's action.

The student modeller evaluates the student's solution against the knowledge base and updates the student model. The short-term student model consists of a list of violated and a list of satisfied constraints for the current attempt. The long-term model records the history of usage for each constraint. This information is used to select problems of appropriate complexity for the student, and to generate feedback.

All DB-suite tutors contain predefined database problems. KERMIT and SQL-Tutor also contain a pre-specified ideal solution for each problem, as there are no problem solvers for these two tutors. NORMIT, on the other hand, contains a problem solver, and is capable of solving both pre-specified problems and the problems entered by students.

The pedagogical module (PM) is the driving engine of the whole system. Its main tasks are to generate appropriate feedback messages for the student and to select new practice problems. PM individualizes these actions to each student based on their student model. Unlike ITSs that use model tracing (Anderson et al., 1996; Corbett et al., 1998; Koedinger et al., 1997), constraint-based tutors do not follow each student's solution step-by-step: a student's solution is only evaluated once it is submitted, although the student may submit a partial solution to get ideas on how to progress.

The feedback is grouped into six levels according to the amount of detail: *correct*, *error flag*, *hint*, *detailed hint*, *all errors* and *solution*. The first level of feedback, *correct*, simply indicates whether the submitted solution is correct or incorrect. The *error flag* indicates the type of construct (e.g. entity, relationship, etc.) that contains the error. *Hint* and *detailed hint* offer a feedback message generated from the first violated constraint. *Hint* is a general message such as "There are attributes that do not belong to any entity or relationship". On the other hand, *detailed hint* provides a more specific message such as "The 'Address' attribute does not belong to any entity or relationship", where the details of the erroneous object are given. Not all detailed hint messages give the details of the construct in question, since giving details on missing constructs would give away solutions. A list of feedback messages on all violated constraints is displayed at the *all errors* level. Finally, the complete solution is displayed at the *solution* level.

Initially, when the student begins to work on a problem, the feedback level is set to the *correct* level. As a result, the first time a solution is submitted, a simple message indicating whether or not the solution is correct is given. This initial level of feedback is deliberately low, as to encourage students to solve the problem by themselves. The level of feedback is incremented with each submission until the feedback level reaches the

detailed hint level. Automatically incrementing the levels of feedback is terminated at the *detailed hint* level to encourage to the student to concentrate on one error at a time rather than all the errors in the solution. Moreover, if the system automatically displays the solution to the student on the sixth attempt, it would discourage them from attempting to solve the problem at all, and may even lead to frustration. The system also gives the student the freedom to manually select any level of feedback according to their needs.

When selecting a new problem, the PM firsts decides what concept is appropriate for the student on the basis of the student model. The concept that contains the greatest number of violated constraints is targeted. We have chosen this simple problem selection strategy in order to ensure that students get the most practice on the concepts with which they experience difficulties. In situations where there is no obvious "best" concept (i.e. a prominent group of constraints to be targeted), the next problem in the list of available problems, ordered according to increasing complexity, is given. We have also experimented with alternative problem-selection strategies, using Bayesian nets (Mayo & Mitrovic 2000; 2001) and neural networks (Wang & Mitrovic, 2002).

SQL-TUTOR

SQL-Tutor is our most heavily developed constraint-based tutor. The motivation for developing this tutor came from our teaching experience. SQL is usually taught in classrooms, by solving problems on the blackboard, complemented by lab exercises. Students experience many problems when learning SQL. Some errors come from the burden of having to memorize database schemas; others come from misconceptions in the student's understanding of the elements of SQL and the relational data model in general. Some of the concepts students find particularly difficult to grasp are grouping and restricting grouping. Join conditions and the difference between aggregate and scalar functions are another two common sources of confusion. Furthermore, students find that it is not easy to learn SQL directly by working with a RDBMS, because error messages are very often hard to understand, and are limited to the syntax only.

The Web-enabled version of SQL-Tutor has been used in regular courses at the University of Canterbury since 1999. For a detailed discussion of the system, see (Mitrovic, Martin & Mayo, 2002); here we present only some of its features. The system contains definitions of several databases and a set of problems and their ideal solutions. SQL-Tutor contains no problem solver. The interface, illustrated in Figure 2, has been designed to be robust, flexible, and easy to use. It reduces the memory load by displaying the database schema and the text of a problem, by providing the basic structure of the query, and also by providing explanations of the elements of SQL. The top area contains the buttons students can use to request a new database/problem, see the history of the current session or their student model, ask for help and run their query. The middle left section displays the text of the problem being solved and students can remind themselves easily of the elements requested in queries. The middle left part also contains the clauses of the SELECT statement, thus visualizing the goal structure. Students need not remember the exact keywords used and the relative order of clauses. The middle right part is where the feedback and other help messages are displayed. The bottom part displays the schema of the current database. Schema visualization is very important; all database users are painfully aware of the constant need to remember table and attribute names and the corresponding semantics. Students can get the descriptions of databases, tables or attributes. The motivation here is to remove from the student some of the

cognitive load required for checking the low-level syntax, and to enable the student to focus on higher-level, query definition problems.

SQL-Tutor checks the student's solution by comparing it to the correct solution using domain knowledge represented in the form of more than 600 constraints. The student may select problems in several ways: they may work their way through a series of problems for each database (ordered by their complexity), ask the system to select a problem on the basis of their student model, select a problem from a list, or select the type of problem they wish to work on, where the system then selects an individual problem of that type on the basis of their student model.

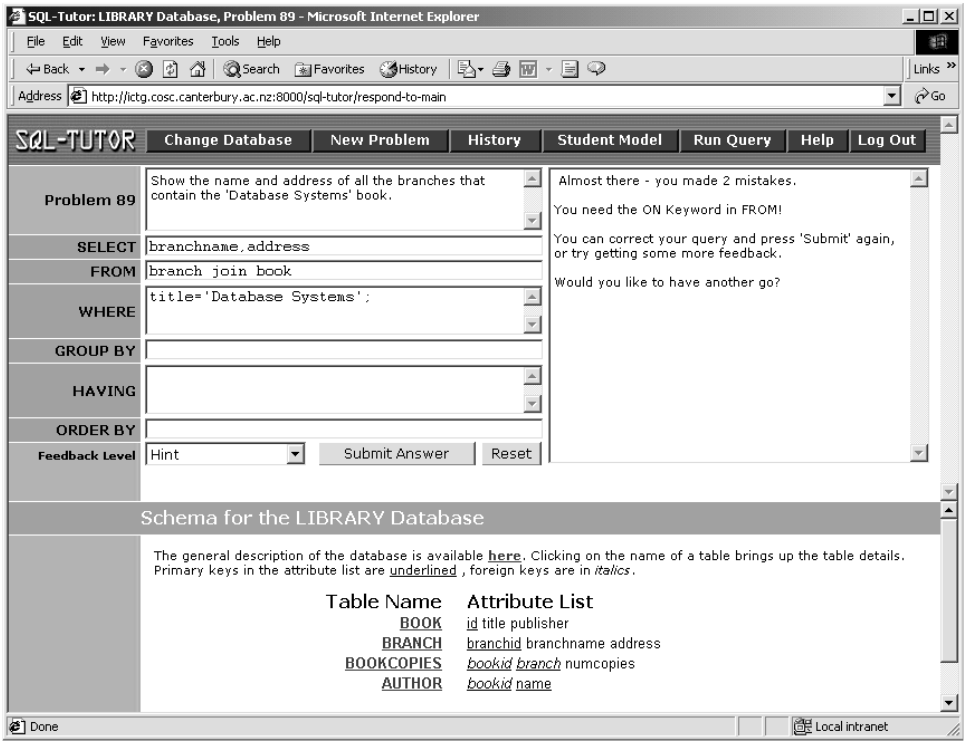

Fig 2. A screenshot from SQL-Tutor

KERMIT: A KNOWLEDGE-BASED ER MODELLING TUTOR

Learning how to develop good quality databases is a core topic in the Computer Science curriculum. Database design is a process of generating a database schema using a specific data model. The quality of conceptual schemas is of critical importance for database systems. Most database courses teach conceptual database design using the Entity-Relationship (ER) model, a high-level data model originally proposed by Chen (1976). Although the traditional method of learning ER modeling in a classroom environment may be sufficient as an introduction to the concepts of database design, students cannot

gain expertise by attending lectures only: like other design tasks, extensive practise is necessary. KERMIT assists students in this task. The system is designed to complement classroom teaching, and therefore assumes that students are already familiar with the fundamentals of database theory. In KERMIT, students construct ER schemas that satisfy a given set of requirements. The system assists students during problem solving and guides them towards the correct solution by providing tailored feedback.

The system is designed for individual work. The student is given a textual description of the requirements of the database, and uses the ER modelling notation to construct an ER schema, as shown in Figure 3. KERMIT's interface consists of three main components. The top part contains the controls for the student to ask for a new problem, look at the history of the current session, explore their student model, ask for help or log out. The middle component is the Java applet, which displays the text of the problem. It also provides an ER modeling workspace where students create ER diagrams. The lower window displays feedback from the system in textual form. The ER diagram is constructed using the workspace integrated into KERMIT's interface. Whenever a new object is created, the system asks for it to be named by highlighting a phrase from the problem text. This interface has two benefits: the student is forced to think about the requirements in terms of the original problem text, and it is also easier for the tutor to understand the semantics of the constructs in the student's diagram. Once the student has completed the problem or requires guidance from the system, the solution is evaluated. Depending on the results of the evaluation, the system may either congratulate the student or offer hints on their errors.

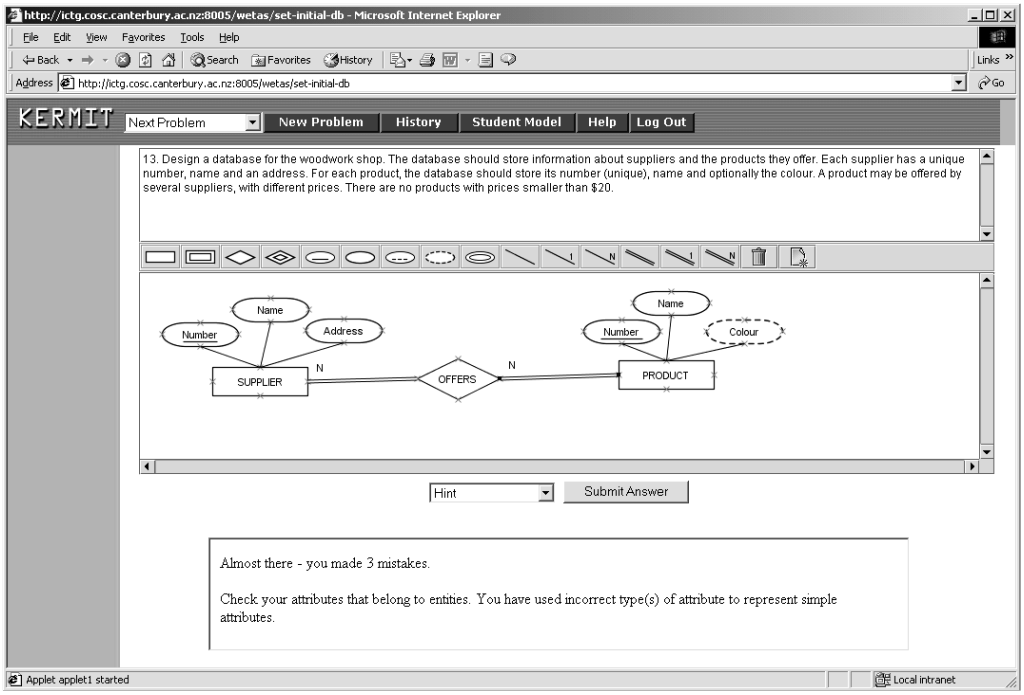

Fig. 3. KERMIT's interface

The Web-enabled version of KERMIT was developed in WETAS (Martin & Mitrovic, 2003). The domain knowledge of KERMIT is represented as a set of constraints, which is used for testing the student's solution (for syntax errors) and comparing it to the ideal solution. Currently KERMIT's knowledge base consists of 135 constraints. Most syntactic constraints of KERMIT were formulated by analysing the target domain of ER modelling through the literature (Elmasri & Navathe, 1994). Due to the nature of the domain, the acquisition of syntactic constraints was not straightforward. Since ER modelling is an ill-defined domain, descriptions of its syntax in textbooks are informal. This process was conducted as an iterative exercise in which the syntax outline was repeatedly refined by adding new constraints. Semantic constraints are even harder to formulate: we analysed sample ER diagrams and compared them against their problem specifications to derive the basic semantic constraints.

LEARNING DATA NORMALIZATION IN NORMIT

Database normalization is the process of refining a relational database schema in order to ensure that all tables are of high quality (Elmasri & Navathe, 1994). Normalization is usually taught in introductory database courses in a series of lectures, and later practised on paper by looking at specific databases and applying the definitions. NORMIT is a problem-solving environment, which complements traditional classroom instruction. The emphasis is therefore on problem solving, not on providing information. However, the

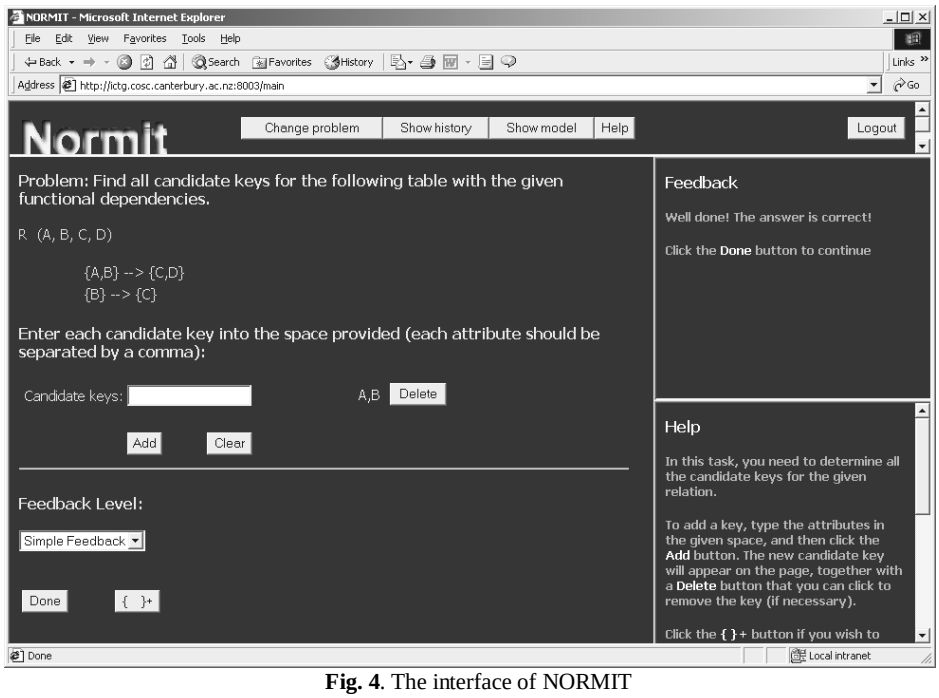

Fig. 4. The interface of NORMIT

system does provide help about the basic domain concepts, when there is evidence that the student does not understand them, or has difficulties applying knowledge.

Database normalization is a procedural task: the student goes through a number of steps to analyze the quality of a database. We described the tasks NORMIT supports in detail elsewhere (Mitrovic, 2002; 2003). NORMIT requires the student to determine candidate keys (illustrated in Figure 4), the closure of a set of attributes, prime attributes, simplify functional dependencies, determine normal forms, and, if necessary, decompose the table. The sequence is fixed: the student will only see a Web page corresponding to the current task. The student may submit a solution or request a new problem at any time. He/she may also review the history of the session, or examine their student model.

NORMIT currently contains over 80 problem-independent constraints that describe the basic principles of the domain. Some constraints check the syntax of the solution, while others check the semantics by comparing the student's solution to the ideal solution, generated by the problem solver. In order to identify constraints, we studied material in textbooks, such as (Elmasri & Navathe 1994), and also used our own experience in teaching database normalization.

NORMIT also provides support for self-explanation, one of the most effective learning strategies. In self-explanation, the student solves a problem (or explains a solved problem) by specifying why a particular action is needed and how it contributes toward the solution. Existing ITSs that support self-explanation, such as Geometry Explanation Tutor (Aleven & Koedinger, 2002) and SE-Coach (Conati & VanLehn, 2000), require the student to explain every problem-solving step. Instead, NORMIT requires an explanation

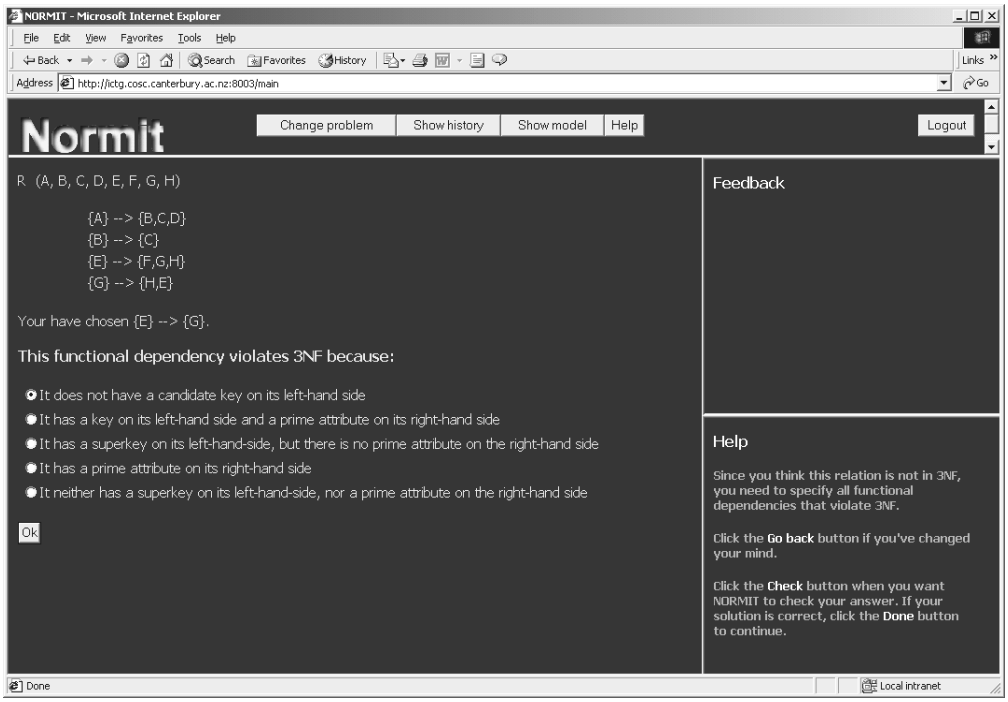

Fig. 5. Asking the student to explain the current action

for actions being performed for the *first time only*. For subsequent actions of the same type, explanation is required only if the action is performed incorrectly. This strategy reduces the burden on the more able students (by not asking them to provide the same explanation every time an action is performed correctly), and at the same time provides enough situations for students to develop and improve their self-explanation skills. Figure 5 shows a situation when the student has specified a functional dependency that violates the third normal form (3NF) incorrectly. The tutor asks the student to specify the reason for selecting this functional dependency. If the student's explanation is incorrect, they will be given another question, asking them to define the underlying domain concept. The purpose of the questions is to require the student to relate their problem-solving actions (generative knowledge) to declarative knowledge, thus supporting the acquisition of deep knowledge.

In addition to the model of the student's knowledge, NORMIT also stores information about the student's self-explanation skills. For each constraint, the student model contains information about the student's explanations related to that constraint. The student model stores the history of student's explanation of each domain concept.

EVALUATION OF DB-SUITE TUTORS

We believe that the credibility of an ITS can only be gained by proving its effectiveness in a classroom environment, with typical students (Mitrovic, Mayo & Martin, 2002). This section presents the results of several evaluation studies performed on the presented tutors.

Evaluating SQL-Tutor

The stand-alone version of the system was evaluated in 1998 (Mitrovic & Ohlsson, 1999), showing that the system had a significant effect on students' knowledge after a single two-hour session. Here we report on evaluation studies performed on the Web-enabled version of SQL-Tutor. General information about the studies is given in Table 1. In all studies, students had 4-6 lectures and labs before using the system. Their performance was measured by a pre/ and a post-test. Every action performed by a student was logged, and the logs were later analysed. All studies were carried out at the University of Canterbury, with Computer Science students enrolled in database courses. Each study had a specific focus. In this paper, we report on two dimensions: usability and learning.

Study	Timing	Students	Length	Purpose of study
1	May 1999	33	2 hours	Feedback evaluation
2	October 1999	34	2 hours	Animated pedagogical agent; Probabilistic student model
3	Sep-Oct 2000	70	7 weeks	Meta-cognitive skills
4	Sep-Oct 2001	77	1 month	Open student model
5	Sep-Oct 2002	100	1 month	Problem selection

Table 1. Details of the evaluation studies

The first two studies involved a single, 2-hour long session. We refer the interested reader to (Mitrovic & Suraweera, 2000) for the details of the evaluation of the pedagogical agent, and to (Mayo & Mitrovic, 2000) for the details of the evaluation of the probabilistic student model. Studies 3, 4 and 5 were longer. In each of these studies, SQL-Tutor was demonstrated in a lecture. The course involved a test on SQL a month and a half after the system was introduced. The experiments were set up this way so that the students may use the system over several weeks. The goal of study 3 was to analyze students' metacognitive skills, and the results are described in (Mitrovic, 2001). In study 4, we introduced an open student model, which presented an overview of student's knowledge. The goal of that study was to see whether this open model supports learning and self-assessment skills (Mitrovic & Martin, 2002). Finally, in study 5 we analysed whether students can learn to select problems well (Mitrovic & Martin, 2003).

We analyzed the student logs to evaluate how well SQL-Tutor supports learning. Since we represent knowledge of SQL in terms of constraints, we looked at how students acquire and apply them. In earlier work (Mitrovic & Ohlsson, 1999), the evaluation of **SQL-Tutor** showed that constraints represented psychologically appropriate units of knowledge; learning followed a smooth curve when plotted in terms of constraints. We performed the same analysis for **SQLT-Web**. Figure 6 shows the decrease in the number of violated constraints as a function of the number of times each constraint was relevant. The degree of mastery of a given constraint is a function of the amount of practice on that unit. There is not much difference between the three student populations, as the graphs for three evaluation studies are close to each other. In other words, the students from each of the three studies tended to acquire constraints at approximately the same rate.

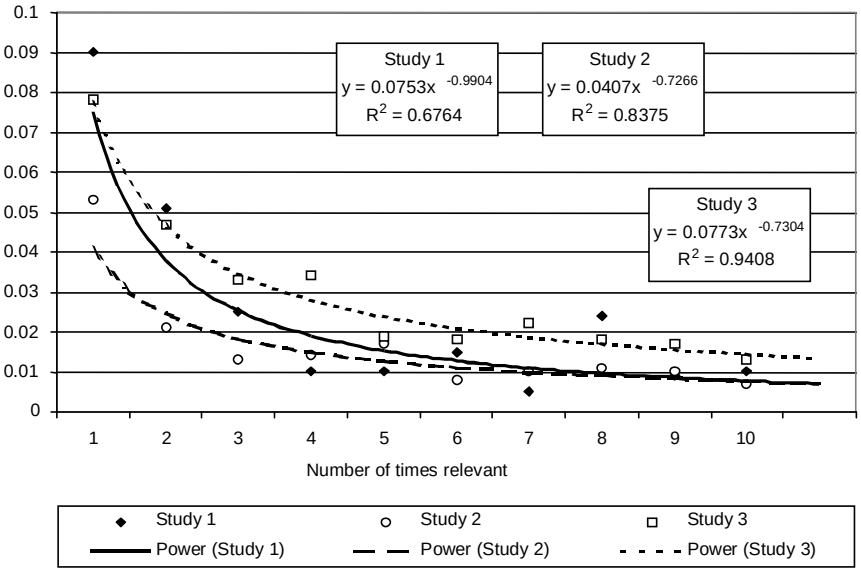

Fig. 6. Mastery of SQL-Tutor's constraints

In the third study we compared the performance of students who used SQL-Tutor (experimental group) to the rest of the class (control group). These two groups listened to

exactly the same number of lectures and labs, and sat the same post-test. The pre-test was administered on-line, when the students logged on to the system for the first time. The results of the experimental group on the post-test are higher than the results of the control group, and the difference is significant ($p<0.005$). However, this result is not irrefutable, as the experiment was not controlled. The experimental group consisted of volunteers, who are usually more motivated students.

Group	Students	Pre-test mean (SD)	Post-test Mean (SD)
Experimental	70	4.02 (1.52)	5.01 (1.24)
Control	62		4.3 (1.6)

Table 2. Pre- and post-test results for study 3

The goal of study 4 was to determine the effect of a simple open student model on students' learning and self-assessment skills. Let us first describe the way we visualize the student model. The student model in SQL-Tutor is implemented as an overlay on top of the constraint base. There are currently more than 600 constraints in the system, and therefore it is not possible to visualize information about each constraint. Instead, we generalized the student model to resemble the structure of the SELECT statement: the student is shown six "skill-ometers", which show the student model in terms of the six clauses of the answer. For each clause we find all the relevant constraints and compute the *coverage* (the percentage of constraints that the student has used) and *correctness* (i.e. the percentage of all relevant constraints that the student has used correctly). These two percentages are visualized as shown in Figure 7.

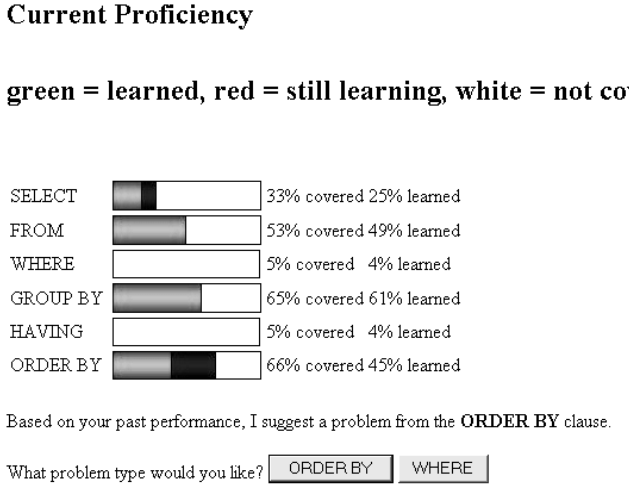

Fig. 7. The visualization of the student model

Study 4 (Mitrovic & Martin, 2002) was ablative: the experimental group had access to the open student model, while the control group had not. Although we did not see any significant difference in the post-test scores of the control and the experimental group, the *less able* students from the experimental group have scored significantly higher than

comparable students from the control group. Further, the more able students who had access to their models abandoned significantly less problems than their counterparts from the control group and had stronger opinions on what they should work on next, which often varied from the system's suggestions. Overall, these results suggest that the open model may have improved the performance of less able students and boosted the self-confidence of more able students, such that they abandoned fewer problems and judged their own abilities more readily.

The goal of study 5 (Mitrovic & Martin, 2003) was to investigate whether students can learn how to select problems with the support of an open student model and scaffolded problem selection. For this study we developed three versions of the system, differing from each other in the problem selection strategy. We wanted the student to reflect on their knowledge, in order to identify the type of problems they have difficulties with. To support reflection, we open the student model to the users in the same way as in study 4. The three versions of the system used in the study support different problem selection strategies. In the first version, the system selects the appropriate type of problem for the student on the basis of the student model. When the student asks for a new problem, they get a page showing their student model, and a message specifying what type of problem is selected by the system. In the second version, the student is always asked to select a type of problem. In the last version, problem selection is faded. For novices, the student is asked to select the type of the problem. If the student's selection differs from what the system prefers, the student receives a new page, showing the student model and specifying the system's preference. Once the student's level increases over the threshold, the student is allowed to select the type of problems without system's intervention. We hypothesized that this version would support less able students in acquiring metacognitive skills, by opening the problem-selection strategy to them and supporting reflection on their knowledge via the overview of their student model. Once the type of problem has been determined in one of the previous three ways, the system searches for problems of the appropriate type that have not been solved yet. The system then selects one that is at the appropriate level of complexity for the student's current state. Table 3 summarizes the experimental design. We assessed students' abilities by a pre-test. More able students were randomly allocated to versions where problems were

Ability	**Problem selection**		
More able	System	Student	N/A
Less able	System	Student	Faded

Table 3. The five groups in study 5

selected by the student or by the system. Less able students were randomly allocated to one of the three versions of the system. We hypothesized that less able students would do the best in the faded condition, and worst when selecting problems on their own. We further hypothesized that less able student would only be able to acquire problem-selection skills in the faded condition.

Table 4 gives the results on the pre- and post-test for students who have sat both. The two more able groups achieved higher results on the pre-test than on the post-test, but the difference is not significant. In previous studies with SQL-Tutor, more able students either improved (Mitrovic & Martin, 2002) or achieved slightly lower scores on the post-test (Mitrovic, 2001). All three less able groups improved on the post-test, but the improvement is significant for the *faded* group only. This supports our hypothesis that

less able students are not good in problem selection, and therefore learn more when they do not need to select problems by themselves.

Group	Students	Pre-test mean (SD)	Post-test mean (SD)
More able - system	6	7.17 (1.17)	5.83 (1.47)
More able - student	6	6.67 (1.03)	5.17 (1.94)
Less able - system	6	3.33 (0.52)	4.67 (1.86)
Less able - student	3	3.67 (1.15)	4 (2)
Less able - faded	9	4.22 (0.97)	5.55 (1.51)

Table 4. Pre/post test results

The experimental results did not support our first hypothesis: more able students appeared to be no better at problem selection than their less able counterparts, with *all* students benefiting from system assistance at problem selection. However, the results *did* highlight the importance of problem selection: students that had system help performed best on the post-test. It also appears that attempts to coach students in the skill of problem selection were successful: the students in the faded group improved their selection accuracy, and performed better at selection than the students who were not coached.

Evaluating KERMIT

An evaluation study was carried out at the University of Canterbury in August 2001. The study involved sixty-two volunteers from students enrolled in the Introduction to Databases course (COSC 226) offered by the Computer Science department. This second-year course teaches ER modelling as outlined by Elmasri and Navathe (1994). The students had learnt ER modelling concepts during two weeks of lectures and had some practice during two weeks of tutorials prior to the study.

The evaluation study was conducted in two streams of two-hour laboratory sessions. The participants interacted with either KERMIT (experimental group) or ER-Tutor (control group), a cut-down version of the system that provided no feedback on students' solutions. The set of problems and the order in which they were presented was identical for both groups. A total of six problems were ordered in increasing complexity. Each session proceeded in four distinct phases. Initially each student was given a document that contained a brief description of the study and a consent form. The students sat a pre-test and then interacted with the system. Finally, the participants were given a post-test and a questionnaire. The questionnaire contained fourteen questions. Initially students were questioned on previous experience in ER modelling. Most questions asked the participants to rank their perception on various issues on a Likert scale with five responses ranging from *very good* (5) to *very poor* (1), and included the amount they learnt about ER modelling by interacting with the system and the enjoyment experienced. The students were also allowed to give free-form responses. Finally, suggestions were requested for enhancing the system.

Table 5 displays a summary of the questionnaire responses. Both groups required approximately the same time to learn the interface, thought to have learnt the same amount, and enjoyed the system similarly. The free-form comments from the experimental group emphasized the importance of feedback for their learning. The students who used ER-Tutor rated its interface easier to use in comparison to the students who used KERMIT. The difference of 0.46 in favour of ER-Tutor's interface is statistically significant ($p<.01$). This result was expected since KERMIT's interface is more complex

than ER-Tutor's. The mean rating for the usefulness of feedback is significantly higher for the experimental group ($p<.01$). These results are analogous with our expectations due to the difference in the information content presented as feedback from each system. KERMIT provides individualised feedback, while the students who used ER-Tutor only had the option of viewing the completed solution to each problem. 74% of the students who used ER-Tutor indicated the need for more detailed help other than the complete solution, compared to 61% of the students who used KERMIT.

	KERMIT		**ER-Tutor**	
	mean	**s. d.**	**mean**	**s. d.**
Time to learn interface (min.)	11.50	11.68	11.94	14.81
Amount learnt	3.19	0.65	3.06	0.89
Enjoyment	3.45	0.93	3.42	1.06
Ease of using interface	3.19	0.91	3.65	1.08
Usefulness of feedback	3.42	1.09	2.45	1.12

Table 7. Mean responses from the user questionnaire for the evaluation study

We evaluated how student's learnt in KERMIT by analysing the student logs and identifying each problem-state in which a constraint was relevant, the same way as in SQL-Tutor. The results are shown in Figure 8. The power curve displays a close fit with an R^2 power-law fit of 0.88. The probability of 0.23 for violating a constraint at its first occasion of application has decreased to 0.12 at its sixteenth occasion of application displaying a 53% decrease in the probability.

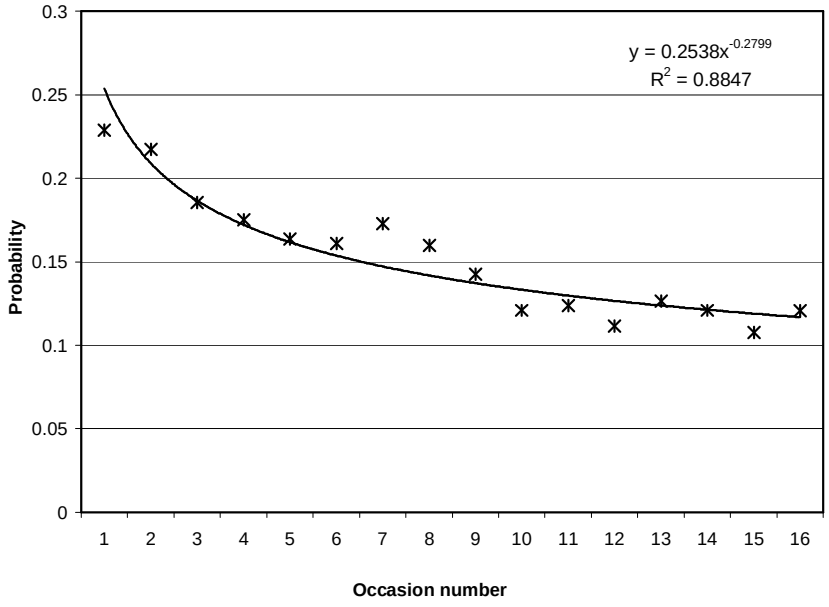

Fig. 8. Probability of violating a constraint as a function of the occasion when that constraint was relevant, averaged over all participants in the pilot study

The mean scores of the pre- and post-test (out of a possible 22) are shown in Table 6. The difference in scores on the pre-test is statistically insignificant, confirming that the two groups are comparable. The experimental group achieved significantly higher score on the post-test ($p<.01$). Conversely, the difference in pre- and post-test scores of the group who used ER-Tutor is statistically insignificant. The difference in post-test scores of the two groups is statistically significant ($p<.05$). We can conclude from these results that students who used KERMIT learnt more than the control group students. The effect size of the experiment is 0.63, which is comparable with the effect sizes of 0.63 published by Albacete and Vanlehn (2000) and 0.66 published in (Mitrovic et al., 2002). Both published results are also results from experiments that spanned a two-hour session. An effect size of 0.63 with the students interacting with the system for approximately an hour is an excellent result.

	Pre-test	**s. d.**	**Post-test**	**s. d.**
KERMIT	16.16	1.82	17.77	1.45
ER-Tutor	16.58	2.86	16.48	3.08

Table 6. Mean pre- and post-test scores for the evaluation study

The results show that students' knowledge increased by using KERMIT. Students who interacted with KERMIT achieved significantly higher scores on the post-test, suggesting that they acquired more knowledge in ER modelling. Subjective evaluation shows that the students in the experimental group felt they learnt more than their peers in the control group. It is surprising to record a high mean ranking of approximately 3 for the control group, when asked how much they learnt from ER-Tutor. This may be due to the typical student misconception of assuming that they learnt a lot by analysing the complete solution. The student responses to the questionnaire suggested that most students appreciated the feature of being able to view the complete ER model.

Evaluating NORMIT

A preliminary evaluation of NORMIT was performed in the second half of 2002, with the students enrolled in an introductory database course at the University of Canterbury. Our hypothesis was that self-explanation would have positive effects on both procedural knowledge (i.e. problem solving skills) and conceptual knowledge. Prior to the experiment, all students listened to four lectures on data normalization. The system was demonstrated in a lecture on October 14, 2002 (during the last week of the course), and was open to the students a day later. The accounts for students were generated before the study, and randomly allocated to one of the two versions. The students in the control group used the basic version of the system, while the experimental group used NORMIT-SE, the version of the system that supports self-explanation. The participation was voluntary, and 29 out of 151 students enrolled in the course used the system. The students were free to use NORMIT when and for how long they wanted. There were 10 students in the control, and 19 in the experimental group. The sizes of the groups are different, as not all students who showed interest in participating have actually used the system.

When a student logged on to the system for the first time, he/she was presented with a pre-test. The post-test was also administered on-line, the first time a student logged on to

the system on or after November 1, 2002. The date for the post-test was chosen to be just one day before the exam. We developed two tests, which consisted of four multichoice questions each. The first two questions required students to identify the correct solution for a given problem, while for the other two the students needed to identify the correct definition of a given domain concept. Each student got one of these two tests randomly as the pre-test, and the other one as the post-test.

We collected data about each session, including the type and timing of each action performed by the student, as well as the feedback obtained from NORMIT. There were three students who logged on to the system, but have not attempted any problems. We excluded the logs of these three students from analyses. The results on the pre- and post-tests are given in Table 7. The groups are comparable, as there is no significant difference on the pre-test performance. Only three students from the control group sat the post-test, and we have not analysed their results, as the sample was too small. On the other hand, a paired t-test for the students in the experimental group who sat both tests shows that their performance improved significantly (p=0.08), confirming the first part of our hypothesis.

	No of pre-tests	Pre-test % (sd)	No of post-tests	Post-test % (sd)
NORMIT	8	65.62 (36.3)	3	79.17 (25)
NORMIT-SE	18	75 (25.88)	13	89.1 (17.8)

Table 7. Pre- and post-test results

To test the second part of our hypothesis, we analysed student's explanations. Due to imperfection of the logging mechanism, we do not have all information about self-explanations that were problem-specific (those problems have been fixed meanwhile). From the data we have, it can be seen that some constraints are much more difficult for students to learn than others. For example, out of the total of 29 situations when students who were asked to explain why a set of attributes is a candidate key, the correct answer was given in only two cases. However, we do have data about students' self-explanations related to domain concepts. Seven out of 11 concepts NORMIT tracks have been covered by all students. The remaining 4 concepts have been covered only by some students, because these concepts do not appear in every problem, and the problems students attempted vary significantly. Figure 9 shows the probability of giving an incorrect explanation. Please note that students were asked to explain domain concepts only when their problem-specific explanations were incorrect (the total of 147 cases). The probabilities of incorrect answers on the first and subsequent occasions were averaged over all concepts and all students. There is a very good fit to the power curve, which indicates that students do learn by explaining domain concepts.

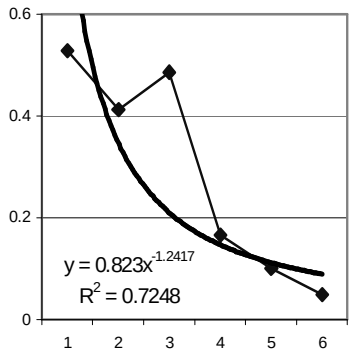

Fig. 9. Defining domain concepts

CONCLUSIONS

The Web has introduced a new paradigm for building widely accessible intelligent educational systems. Web-enabled tutors can be used from any place, and at any time. A very important aspect of Web-based tutors is platform-independence. This paper has discussed the design and implementation of three database tutors. DB-suite consists of three tutors that teach SQL, data normalization and conceptual data modelling to university level students. DB-suite tutors are available from the ICTG Web server, and also on the Addison-Wesley's DatabasePlace, a Web portal supporting several text books in the area of databases (http://www.aw-bc.com/databaseplace/). The effectiveness of DB-suite tutors was evaluated in several experiments. The results demonstrate that the presented tutors are effective educational tools. The participants who used the full version of KERMIT showed significantly better results in both subjective and objective analyses in comparison to the students who practiced ER modelling with a drawing tool. In NORMIT, the students who self-explained improved significantly both in problem solving and in answering questions about domain knowledge. The evaluation studies performed on SQL-Tutor show that it supports both learning and the acquisition of metacognitive skills.

There are several avenues to be explored to further enhance the presented tutors. We have already carried out several projects focusing on supporting meta-cognitive skills, such as self-explanation (Weerasinghe & Mitrovic, 2002; 2003) and reflection. All DB-suite tutors present a summary of the student model, thus supporting self-assessment and deeper understanding of the domain. For details of evaluating the effects of such open model, please see (Mitrovic & Martin 2002; 2003; Hartley & Mitrovic, 2002). We plan to add more sophisticated, adaptive support for reflection and self-explanation to all tutors.

As mentioned earlier, we have also developed WETAS, an authoring shell for developing constraint-based tutors (Martin & Mitrovic, 2002a; 2003). For text-based tutors, WETAS basically requires the author to provide only the knowledge base and a set of problems and their solutions; all other functions are provided by WETAS. For graphical domains (such as KERMIT), they also need to provide the interface. Although WETAS is a powerful ITS *engine*, studies of how novice authors use WETAS to develop new tutors (Martin & Mitrovic, 2003) indicate that more support is required for authoring. We are currently looking at ways to enhance WETAS to ease the authoring process, including automated knowledge acquisition, support for structuring the domain model into ontologies and tools for assisting the re-use of existing domain models.

Acknowledgements

This research could not have been done without the support of other past and present members of ICTG. The work presented here was supported by the University of Canterbury research grants U6430 and U6532. We also thank our students for putting their time and effort into trying out our tutors and commenting on them.

References

Albacete, P. L. & VanLehn, K. (2000) The Conceptual Helper: an Intelligent Tutoring System for Teaching Fundamenatal Physics Concepts. *In* Gauthier, G., Frasson, C. and VanLehn, K.

(eds.). *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Montreal, Springer, pp. 564-573.

Aleven, V. & Koedinger, K. (2002) An Effective Metacognitive Strategy: Learning by Doing and Explaining with a Computer-based Cognitive Tutor. *Cognitive Science*, 26, pp. 147-179

Anderson, J. R., Corbett, A., Koedinger, K. & Pelletier, R. (1996) Cognitive Tutors: Lessons Learned. *Journal of Learning Sciences*, *4* (2), pp. 167-207.

Bloom, B. S. (1984) The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*, pp. 4-16.

Chen, P. P. (1976) The Entity Relationship Model - Toward a Unified View of Data. *ACM Transactions Database Systems*, *1* (1), 1976, pp. 9-36.

Conati, C. & VanLehn, K. (2000) Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *Int. J. AI in Education*, 11, 389-415.

Corbett, A. T., Trask, H. J., Scarpinatto, K. C. & Hadley, W. S. (1998) A Formative Evaluation of the PACT Algebra II Tutor: Support for Simple Hierarchical Reasoning. *In* Goettl, B. P., Halff, H. M., Redfield, C. L. and Shute, V. J. (eds.). *Proc. of 4th International Conference on Intelligent Tutoring Systems*, San Antonio, Texas, pp. 374-383.

Elmasri, R. & Navathe, S. B. (1994) *Fundamentals of Database Systems*. Addison Wesley, *2nd edition.*

Hartley, D. & Mitrovic, A. (2002) Supporting learning by opening the student model. In Cerri, S., Gouarderes, G. and Paraguacu, F. (eds.) *Proc. 6th International. Conference on Intelligent Tutoring Systems ITS 2002*, Biarritz, France, pp. 453-462.

Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1997) Intelligent tutoring goes to school in the big city. *Int. J. AI in Education*, *8* (1), pp. 30-43.

Martin, B. & Mitrovic, A. (2002a) Automatic Problem Generation in Constraint-Based Tutors. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) *Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002*, Biarritz, France, LCNS 2363, 2002: 388-398.

Martin, B. & Mitrovic, A. (2002b) Authoring Web-Based Tutoring Systems with WETAS. Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, C-H Lee (eds) *Proc. Int. Conf. on Computers in Education ICCE 2002*, Auckland, pp. 183-187.

Martin, B. & Mitrovic, A. (2003) Domain Modeling: Art or Science? In: U. Hoppe, F. Verdejo & J. Kay (ed) *Proc. 11th Int. Conference on Artificial Intelligence in Education AIED 2003*, IOS Press, pp. 183-190.

Mayo, M. & Mitrovic, A. (2000) Using a probabilistic student model to control problem difficulty. *Proc. ITS'2000*, G. Gauthier, C. Frasson and K. VanLehn (eds), Springer, pp. 524-533.

Mayo, M. & Mitrovic, A. (2001) Optimising ITS Behaviour with Bayesian Networks and Decision Theory'. *Int. Journal on Artificial Intelligence in Education*, v12no2, 124-153.

Mayo, M., Mitrovic, A. & McKenzie, J. (2000) CAPIT: An Intelligent Tutoring System for Capitalisation and Punctuation. *In* Kinshuk, Jesshope, C. and Okamoto, T. (eds.). *Proc. of Advanced Learning Technology: Design and Development Issues*, Los Alamitos, CA, IEEE Computer Society, pp. 151-154.

Mitrovic, A. (1998a) Experiences in Implementing Constraint-Based Modelling in SQL-Tutor. *In* Goettl, B. P., Halff, H. M., Redfield, C. L. and Shute, V. J. (eds.). *Proc. of 4th International Conference on Intelligent Tutoring Systems*, pp. 414-423.

Mitrovic, A. (1998b) Learning SQL with a Computerised Tutor. *In Proc. of 29th ACM SIGCSE Technical Symposium*, Atlanta, pp. 307-311.

Mitrovic, A. (2001) Investigating students' self-assessment skills. In: M. Bauer, P.J.Gmytrasiewicz and J. Vassileva (eds.) *Proc. 8th Int. Conference on User Modeling UM 2001*, Berlin: Springer-Verlag LNAI 2109, pp. 247-250.

Mitrovic, A. (2002) NORMIT, a Web-enabled tutor for database normalization. Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, C-H Lee (eds) *Proc. Int. Conf. Computers in Education ICCE 2002*, Auckland, pp. 1276-1280.

Mitrovic, A. (2003) Supporting Self-Explanation in a Data Normalization Tutor. In: V. Aleven, U. Hopppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo, K. Yacef (eds) Supplementary proceedings, AIED 2003, pp. 565-577.

Mitrovic, A., Koedinger, K. & Martin, B. (2003) A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. P. Brusilovsky, A. Corbett, F. de Rosis (Eds.) Proceedings of the Ninth International Conference on User Modeling UM 2003, Springer-Verlag, LNAI 2702, pp. 313-322.

Mitrovic, A. & Martin, B. (2002) Evaluating the effects of open student models on learning. In: P. de Bra, P. Brusilovsky and R. Conejo (eds) *Proc. 2nd Int. Conf on Adaptive Hypermedia and Adaptive Web-based Systems AH 2002*, Malaga Spain, LCNS 2347, pp. 296-305.

Mitrovic, A., Martin, B. (2003) Scaffolding and fading problem selection in SQL-Tutor. In: U. Hoppe, F. Verdejo & J. Kay (ed) *Proc. 11th Int. Conference on Artificial Intelligence in Education AIED 2003*, IOS Press, pp. 479-481.

Mitrovic, A., Martin, B. & Mayo, M. (2002) Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. Int. J. *User Modeling and User-Adapted Interaction*, v12no2-3, 243-279.

Mitrovic, A., Mayo, M., Suraweera, P. & Martin, B. (2001) Constraint-based Tutors: a Success Story. *In* Monostori, L., Vancza, J. and Ali, M. (eds.). *Proc. of 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-2001)*, Budapest, Springer-Verlag Berlin Heidelberg LNAI 2070, pp. 931-940.

Mitrovic, A. & Ohlsson, S. (1999) Evaluation of a Constraint-based Tutor for a Database Language. *International Journal on AIED, 10* (3-4), pp. 238-256.

Mitrovic, A. & Suraweera, P. (2000) Evaluating an Animated Pedagogical Agent. *Proc. ITS'2000*, G. Gauthier, C. Frasson and K. VanLehn (eds), Springer, pp. 73-82.

Ohlsson, S. (1994) Constraint-based Student Modelling. *In Proc. of Student Modelling: the Key to Individualized Knowledge-based Instruction*, Springer-Verlag, Berlin, pp. 167-189.

Ohlsson, S. (1996) Learning from Performance Errors. *Psychological Review*, 103, pp. 241-262.

Suraweera, P. & Mitrovic, A. (2001) Designing an Intelligent Tutoring System for Database Modelling. *In* Smith, M. J. and Salvendy, G. (eds.). *Proc. of 9th International Conference on Human-Computer Interaction (HCII 2001)*, New Orleans, vol. 2, pp. 745-749.

Suraweera, P. & Mitrovic, A. (2002) KERMIT: a Constraint-based Tutor for Database Modeling. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) *Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002*, Biarritz, France, LCNS 2363, pp. 377-387.

Vasilakos, A., Devedzic, V., Kinshuk, Pedrycz, W. (2004) Computational Intelligence in Web-based Education: a Tutorial. *Interactive Learning Research* (this issue).

Wang, T. & Mitrovic, A. (2002) Using neural networks to predict student's behaviour. In: Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson and C-H Lee (eds.) IProc. Int. Conference on Computers in Education ICCE 2002, Los Alamitos, CA: IEEE Computer Society, pp. 969-973.

Weerasinghe, A. & Mitrovic, A. (2002) Enhancing learning through self-explanation. Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, C-H Lee (eds) Proc. ICCE 2002, Auckland, pp. 244-248.

Weerasinghe, A. & Mitrovic, A. (2003) Effects of self-explanation in an open-ended domain. In: U. Hoppe, F. Verdejo & J. Kay (ed) Proc. 11th Int. Conference on Artificial Intelligence in Education AIED 2003, IOS Press, pp. 512-514.

Formatted: Bullets and Numbering