Feedback Micro-engineering in EER-Tutor

Konstantin ZAKHAROV[1]
Antonija MITROVIC[1]
Stellan OHLSSON[2]

[1]Intelligent Computer Tutoring Group, University of Canterbury, Christchurch, New Zealand
[2]Department of Psychology, University of Illinois at Chicago

Abstract: Although existing educational systems are based on various learning theories, these theories are rarely used when developing feedback. Our research is based on the theory of learning from performance errors, which suggests that feedback should provide long and short-term learning advantages through revision of faulty knowledge in the context of learners' errors. We hypothesized that principled, theory-based feedback would have a positive impact on learning. To test the hypothesis we performed an experiment with EER-Tutor, an intelligent tutoring system that teaches database design. The results of the study support our hypothesis: the students who learned from theory-based feedback had a higher learning rate than their peers. We conclude that learning theories should be used to formulate design guidelines for effective feedback.

1. Introduction

Although research in the area of Artificial Intelligence in Education is abundant, there has not been much said about designing effective feedback. Most effort in the area has focused on student modelling, providing problem-solving support and developing pedagogical strategies such as problem selection. Some researchers have investigated the effect of the timing of feedback on learning [6] (i.e. whether immediate feedback is more beneficial than delayed feedback), but advice on how to phrase feedback in order to maximize its impact on learning is hard to find. McKendree [7] compared goal-oriented feedback to pointing out errors and explaining the causes of errors, with the former type of feedback resulting in increased performance and transfer. Most existing educational systems seem to provide what we call *common-sense feedback*. By this, we assume feedback messages generated by system developers based on their intuition and experience. Very often, such feedback tells students what to do, or points out some mistakes in the student's solution.

However, existing educational systems are almost invariably based on some learning theory (such as [1]). Learning theories propose various views on learning, and can be used to develop feedback design guidelines. We believe that in most educational systems feedback is not in line with the underlying learning theory, and propose that principled, theory-based feedback will be more effective that the common-sense one.

In order to test our hypothesis, we performed a study in the context of EER-Tutor, a constraint-based tutor that teaches EER modelling. As is the case with other constraint-based systems developed within the ICTG, EER-Tutor is based on the theory of learning from performance errors, which we briefly overview in Section 2. Section 3 presents the most important features of EER-Tutor, while section 4 describes how feedback messages were re-engineered. The experiment involved two versions of EER-Tutor: the original version which provided common-sense feedback, and a new version providing feedback based on the underlying theory. Section 5 presents the experiment and the results derived from it. Finally, we present the conclusions and the area of future work in the final section.

2. Learning from Performance Errors and Constraint-Based Modeling

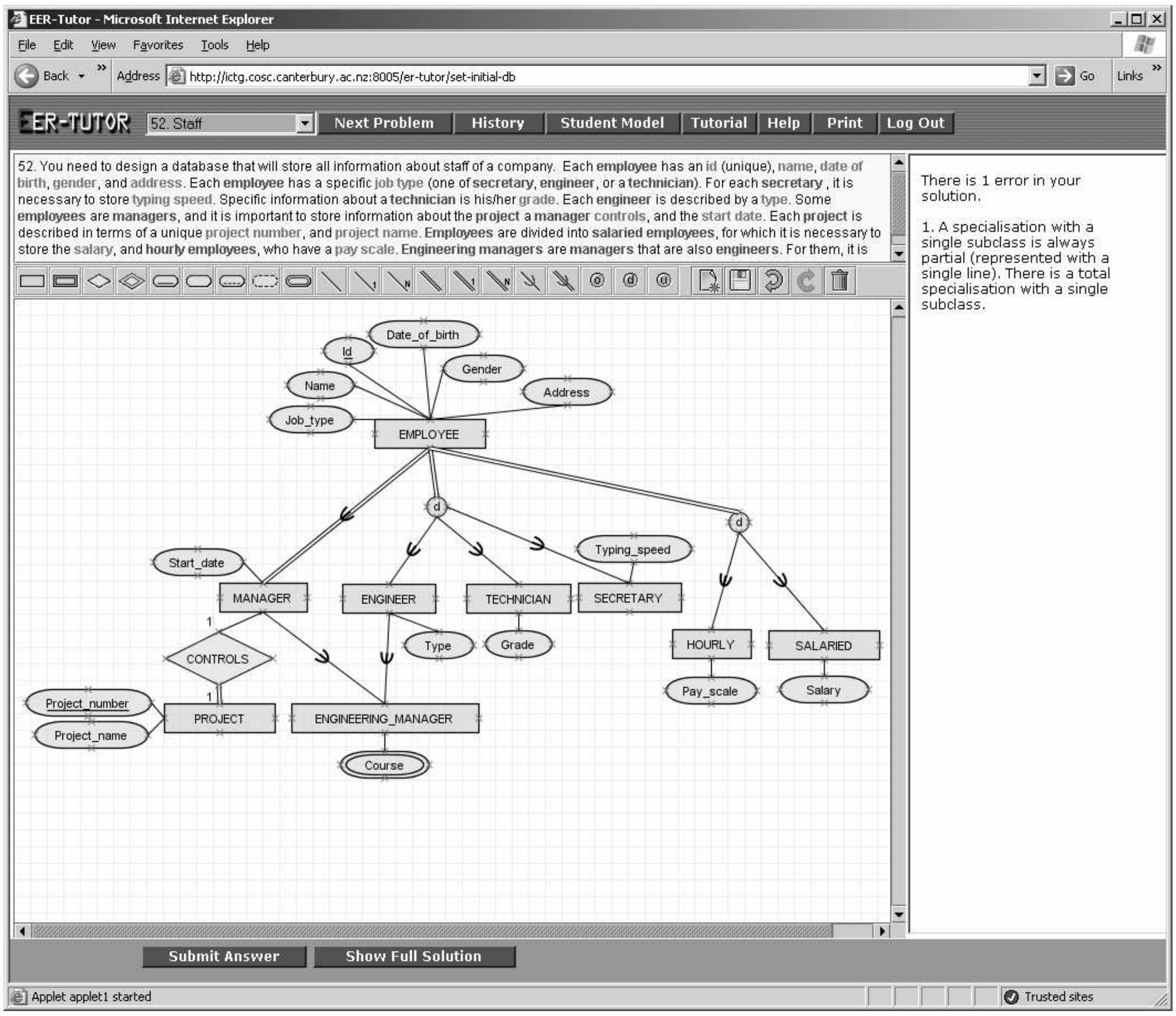
The theory of learning from performance errors [10] proposes that we often make mistakes when performing a task, even when we have been taught the correct way to do it. According to this theory, we make mistakes because the declarative knowledge we have learned has not been internalized in our procedural knowledge, and so the number of decisions we must make while performing the procedure is sufficiently large that we make mistakes. By practicing the task, and catching ourselves (or being caught by a mentor) making mistakes, we modify our procedure to incorporate the appropriate rule that we have violated. Over time, we internalize all declarative knowledge about the task, and so the number of mistakes we make is reduced. The theory views learning as consisting of two phases: *error recognition* and *error correction*. A student needs declarative knowledge in order to detect an error. Only then can the error be corrected so that the solution used is applicable only in situations in which it is appropriate.

Constraint-Based Modeling (CBM) is a student modeling approach [9,8] arising from the above theory. CBM starts from the observation that all correct solutions are similar in that they do not violate any domain principles. CBM is not interested in the exact sequence of states in the problem space the student has traversed, but only in the current state. As long as the student never reaches a state that is known to be wrong, they are free to perform whatever actions they please. Constraints define equivalence classes of problem states. An equivalence class triggers the same instructional action; hence all states in an equivalence class are pedagogically equivalent. It is therefore possible to attach feedback messages directly to constraints. 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.* In other words, if the student solution falls into the state defined by the relevance condition, it must also be in the state defined by the satisfaction condition in order to be correct.

Constraint-based tutors evaluate student solutions by matching them against the constraint set. Firstly, all relevance patterns are matched against the problem state. Secondly, the satisfaction components of relevant constraints are tested. If a satisfaction pattern matches the state, the constraint is satisfied, otherwise, it is violated. The short-term student model consists of all satisfied and violated constraints. Long-term student model mainly consists of the list of all constrains used by the student and the history of constraint usage.

3. EER-Tutor

Conceptual database modelling, in particular Enhanced Entity-Relationship (EER) modelling [3] is a design task. Goel and Pirolli [4] define generic design (i.e. domain-independent characterization of design tasks) as a radical category, which is described in terms of prototypical examples and some unpredictable variations of them. Design tasks are ill-structured, because their start/goal states and problem-solving algorithms are underspecified. The start state is usually described in terms of ambiguous and incomplete specifications. The problem spaces are typically huge, and operators for changing states do not exist. The goal state is also not clearly stated, but is rather described in abstract terms. There is no definite test to use to decide whether the goal has been attained, and consequently, there is no best solution, but rather a family of solutions. Design tasks typically involve huge domain expertise, and large, highly structured solutions. For these reasons, EER modelling presents a considerable learning challenge. The learner is given an abstract definition of a good solution. In database modelling, a good solution is defined as an EER schema that matches the requirements, and



52. You need to design a database that will store all information about staff of a company. Each **employee** has an id (unique), name, date of birth, gender, and address. Each **employee** has a specific job type (one of **secretary**, **engineer**, or a **technician**). For each **secretary** , it is necessary to store typing speed. Specific information about a **technician** is his/her grade. Each **engineer** is described by a type. Some **employees** are **managers**, and it is important to store information about the **project** a **manager** controls, and the start date. Each **project** is described in terms of a unique project number, and project name. **Employees** are divided into **salaried employees**, for which it is necessary to store the salary, and **hourly employees**, who have a pay scale. **Engineering managers** are **managers** that are also **engineers**. For them, it is

There is 1 error in your solution.

1. A specialisation with a single subclass is always partial (represented with a single line). There is a total specialisation with a single subclass.

Fig. 1. A screenshot of EER-Tutor

satisfies all the integrity rules of the chosen data model. We have previously showed that CBM is an effective domain and student modelling approach for design tasks [11].

We developed EER-Tutor, a constraint-based tutor that teaches EER modelling. EER-Tutor is a successor KERMIT, which was shown to significantly increase students' performance [11]. KERMIT is a stand-alone system and recently it has been re-implemented as a web-based tutoring system using WETAS, our web-based authoring shell [5]. For details of the system's architecture, please see [11]. Being a web-based application, EER-Tutor is divided into server and client modules. The server processes learners' solutions, generates feedback, and records all user actions. The client-side can be viewed in any common web browser as a set of dynamic HTML pages. The main page contains a Java applet shown in the centre of the browser window (Figure 1). The applet provides a set of drawing tools for creating EER diagrams. The navigation frame provides controls for stepping between problems, viewing session history and student model, opening EER-Tutor tutorial and help, printing current diagram and terminating the session. The frame at the bottom provides controls for submitting answers and viewing solutions to problems. Feedback is provided when the student submits the solution.

4. Re-engineering feedback for EER-Tutor

The specifics of EER modelling make it difficult to identify errors at the early stages of learning. A novice learner is in a vicious circle: trying to improve performance in some skill, the learner naturally does not intend to make errors but he/she is unable to detect errors, because of the lack of experience and knowledge. The same problem applies to the error correction stage: the learner must revisit faulty knowledge, but with open-ended tasks like EER modelling, the learner will have difficulty identifying relevant knowledge to correct.

CBM comes to the rescue. When an error occurs, the task of error detection and blame assignment is carried out by the system. The system should refer the learner to the relevant part of the domain knowledge. Consequently, an effective feedback message should tell the user (a) where exactly the error is, (b) what constitutes the error (perform blame allocation), and (c) refer the user to the underlying concept of the correct solution (revise underlying knowledge).

The above observations constitute the central focus of our experiment. Existing constraint-based tutors do not utilise these observations in feedback design. Feedback messages in EER-Tutor, as well as other tutors, merely tell the student to check a certain aspect of the solution and accompany a suggestion for correcting the problem. For example, consider the feedback message shown in Figure 1. The student has made a mistake when specifying a specialization of EMPLOYEE into MANAGER: this specialization should be partial (displayed as a single line in the diagram), while the students has specified a total specialization (double line). The intuitive feedback message that we have defined for this situation (coming from the violated constraint) is "*Check how you use subset connectors. In single subclass specialisations, subset connectors should be drawn with single lines.*" The student erroneously used a total specialization due to the lack of experience in extracting the modelling requirements from the problem statement. In particular, the phrase "*Some employees are managers*" in the problem text implies that the specialization should be partial. The error message partially allocates the blame and tells the student what has to be done to correct the error. However, this message does not point out the domain concept that the student has violated, and therefore does not offer help with the revision of underlying faulty knowledge. The message simply tells the learner what to do in order to correct the solution; this is insufficient for successful learning. On the contrary, the following message would (theoretically) have a greater impact: "*A* *specialisation with a single subclass is always partial (represented with a single line). Your solution contains a total specialisation with a single subclass".* This message starts with the general concept which caused the error, aimed at specialising the corresponding rule in the procedural memory, so that next time when a similar situation arises, the learner will hopefully be able to differentiate correctly between choosing partial or total participation. The second sentence ties the concept to the situation at hand, simultaneously pointing out the error and allocating the blame. The error correction information is not essential for the given problem, since there are only two options for specifying participation in a relationship. The careful engineering of every feedback message should theoretically influence learning. For the purposes of our study, we have redefined all feedback messages.

We suspect that common-sense feedback might result in *shallow learning*, which refers to failure in internalising the knowledge and poor knowledge transfer. In other words, the student might learn how to produce solutions that are correct from the system's points of view. However, the student would not be able to perform equally well in a different environment, as he/she does not really understand underlying domain concepts. This point is supported by research proving that learning how to play an educational game does not necessarily imply learning the target instructional domain [2]; learning happens only when students actively build the connections between his/her actions and underlying knowledge. In this light, we expect that the micro-engineered feedback in EER-Tutor will result in better knowledge transfer and deeper learning [12]. Another argument in support of the new feedback style originates from the ACT-R theory [1], but is equally applicable to CBM. The fourth principle of the ITSs design states that a tutoring system should promote an abstract understanding of problem-solving knowledge. This principle was motivated by the observation that students often develop overly specific knowledge from particular problem-solving examples; this is also related to shallow learning and poor knowledge transfer.

5. Evaluation

We performed a study at the University of Canterbury in August 2004. Second year students enrolled in an introductory database course were invited to participate. The students learned EER modelling concepts prior to the study during three weeks of lectures and had some practice during two weeks of tutorials. EER-Tutor was briefly introduced to the class in a lecture. The first session took place in a scheduled laboratory session. The participants were randomly allocated to one of the two versions of the system (referred to as control and experimental condition), differing only in the feedback style. The students were free to use EER-Tutor over two weeks. EER-Tutor contained 56 problems ordered in increasing order of difficulty. The students were not restricted in their choice of problems.

The first session started with an on-line pre-test and at the end of the two week period the students sat an on-line post-test. In this way, most students sat the pre-test in a supervised environment, but the post-test was offered to students in an uncontrolled environment. Two tests of comparable difficulty were interchangeably used for pre-and post-tests.

In order to maximise the effect of feedback, we introduced three restrictions to the users' interaction with the system. The system provided only one level of feedback, listing the messages of the first three errors at most. The students could not see the complete solution for the current problem unless they made at least five attempts at it. If the student saw the solution, the system would not allow further submissions for that problem.

105 students (82% of the class) participated in the study, the general statistics of which are given in Table 1. The maximum numbers of attempted and solved problems were 52 and 43 respectively, while interaction time ranged from 10 minutes to 45 hours. There are no significant differences between the two groups on all these measures. The difference between pre-test results is insignificant, indicating that the two groups had comparable prior knowledge.

Table 1. Statistics from the study

	Students	Time (hours)	Attempted problems	Solved problems	Feedback messages	Pre-test %	No Post-tests	Post-test %
Control	53	16.9 (12.6)	15.5 (11.4)	13.2 (10.3)	24.4 (22.1)	64.2 (26.7)	46	16.6 (7.3)
Exper.	52	15.9 (10.5)	15.2 (10.7)	12.9 (10.2)	23.5 (20)	59.6 (28.7)	45	26.5 (22)

As the post-test was administered on-line, not all students have submitted it, as reported in the table. The low post-test scores are due to many students not taking time to answer the questions. The log files show that many students submitted the post-test only a few seconds after the system displayed it. Even when the time between login and post-test submission is longer, we can not tell apart the situations when students did not answer questions at all or answered them incorrectly. The reason for this is that in the encoding scheme for the post-test results, both a no answer and an incorrect submission were recorded as zero. Consequently, we are unable to use the post-test results to compare the two groups.

We then analyzed how students learned constraints. If constraints represent appropriate units of domain knowledge, the learning should follow a smooth curve [1]. From the logs, we identified all relevant constraints for every attempt. Each constraint relevance occasion was rank-ordered from 1 up. We calculated, for each participant, the probability of violating each constraint at each attempt. The probabilities were then averaged across all the constraints all participants. The cut-off point is set at 50% of the initial number of relevant constraints. The resulting learning curves are shown in Figure 2.a. The probability of constraint violation for both groups decreases regularly (as evidenced by good fits to the power curves). The experimental group violated fewer constraints, and learned constraints faster: their learning rate (-0.2978) is higher than that of the control group (-0.2681).

Using the same approach, we calculated probabilities only for those constraints whose feedback messages had been seen by students, in order to focus on the effect of feedback. The resulting curves are shown in Figure 2.b. Power curve fits for the two groups are lower than in

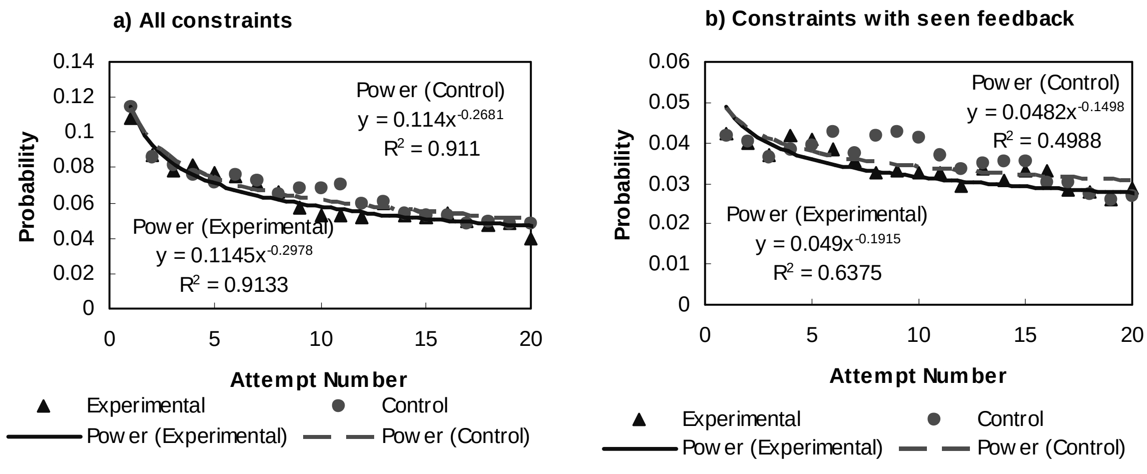


Fig. 2. Learning curves

Figure 2.a, but the probability is much lower, showing that students do learn from getting feedback. The learning rate is still higher for the experimental group.

We also analyzed the number of constraints learned as a function of the number of feedback messages received. If theory-based feedback is better than intuitive style, participants should acquire more knowledge, i.e. more constraints. From the logs, we identified for each participant the number of constraints they learned while interacting with the system. This analysis took into account only those constraints that were not known to the user at the start of the experiment. A constraint is considered as known by the student if the window of five attempts in the constraint history indicated successful application of this constraint in at least 80% cases. The number of learned constraints was then plotted as a function of received instruction, i.e. the number of seen feedback messages (Figure 3.a). The slopes of the trend lines indicate that experimental feedback resulted in more efficient constraint acquisition. Figure 3.b shows the results of the same analysis using a different criterion to test whether a constraint is learned. This time we used a window of three consecutive attempts, and considered a constraint as learned if it was used correctly two or three times.

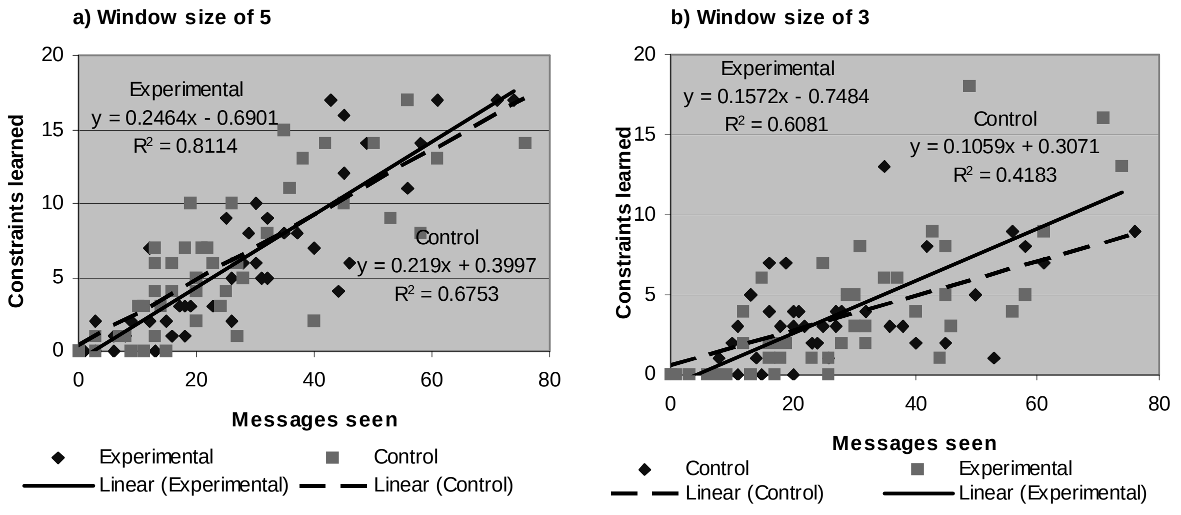


Fig. 3. Learned constraints as a function of feedback received

The participants used the system for a short time, and received a small number of feedback messages. The average number of messages received per one hour of instruction was 1.4, and the average number of learned constraints was 5.9 (sd=5.2). In a realistic situation, the system would be used for hundreds of hours. Using either figure, as participants spend more time with the system, and consequently get more feedback messages, the difference between the two styles of feedback becomes bigger. Extrapolating from Figure 3.a, after 140 feedback messages (based on 100 hours of learning), a student receiving old style feedback would have learned 31.1 constraints, while the theory-based feedback would result in 33.8 learned constraints.

5 Conclusions

This paper reported a project the goal of which was to investigate the role a learning theory might have in formulating feedback for intelligent tutoring systems. We noticed that guidelines for designing effective feedback are rare in research literature, which is strange given the fact that most educational systems claim to be based on various learning theories. Each theory proposes a view on learning, and therefore it should be possible to formulate to identify the principles of effective feedback based on the postulates of the chosen theory.

We have developed a number of constraint-based tutors, starting from the theory of learning from performance errors. This theory can be used by the ITSs to provide learners with extensive support during the learning process. A constraint-based tutor helps the student to identify an error in cases when the student does not have enough experience or knowledge to do that on their own. Effective feedback messages based on this theory should point out the error, and inform the student about the underlying domain principle that has been violated, thus making it possible for the student to revise faulty knowledge.

We hypothesized that principled, theory-based feedback should be more beneficial than intuitive feedback present in most existing systems, including our constraint-based systems. To test the hypothesis, we performed an experiment involving two versions of EER-Tutor, a system that teaches database design. The two version of the systems differed only in the style of feedback give to students. The study showed that feedback developed according to the learning theory provided better learning support, resulting in faster learning rates. The combination of the general domain knowledge relevant to the student’s error, along with the specific details of the error in the given situation provides learning benefits through simultaneous revision of faulty knowledge and strategies.

This paper presented results of a study that lasted only two weeks. We plan to perform a longer study of similar nature. Furthermore, our results seem to have wider consequences, for educational systems based on other learning theories. We believe this is an interesting challenge for the whole IED community.

Acknowledgements: We thank Pramuditha Suraweera, Brent Martin and other members of ICTG for their invaluable help in developing EER-Tutor.

References

1. Anderson, J.R., Lebiere, C. The Atomic Components of Thought. Lawrence Erlbaum Associates, Mahwah, New Jersey, 1998.
2. Conati, C, Zhao, X. Building and Evaluating an Intelligent Pedagogical Agent to Improve the Effectiveness of an Educational Game. In Proc. 9th Int. Conf. on Intelligent User Interface, ACM Press, 2004, pp. 6–13.
3. Elmasri, R., Navathe, S. B. Fundamentals of Database Systems. Addison-Wesley, 2003, 4th edition.
4. Goel, V., Pirolli, P. Motivating the Notion of Generic Design with Information Processing Theory: the Design Problem Space. AI Magazine, v10, 1988, 19-36.
5. Martin, B., Mitrovic, A. 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, 2003.
6. Mathan, S., Koedinger, K. An Empirical Assessment of Comprehension Fostering Features in an ITS. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds) Proc. ITS 2002, Springer, 2002, pp. 330-343.
7. McKendree, J**.** Effective Feedback Content for Tutoring Complex Skills. Human-Computer Interaction, v5no4, 1990, 381-413.
8. Mitrovic, A., Ohlsson, S. Evaluation of a constraint-based tutor for a database language. Int. J. Artificial Intelligence in Education, v10 no3-4, 1999, 238-256.
9. Ohlsson, S. Constraint-Based Student Modelling. In J. Greer and G. McCalla (eds) Student Modelling: The Key to Individualised Knowledge-based Instruction, vol. 125 Computer Systems and Sciences, NATO ASI, Springer-Verlag, 1994, pp. 167–189.
10. Ohlsson, S. (1996) Learning from Performance Errors. *Psychological Review* **103**(2) 241-262.
11. Suraweera, P., Mitrovic, A. An Intelligent Tutoring System for Entity Relationship Modelling. Int. J. Artificial Intelligent in Education, v14no3-4, 2004, 375-417.
12. VanLehn, K., Freedman, R. et al. Fading and deepening: The next steps for ANDES and other model-tracing tutors. In G. Gauthier, C. Frasson, and K. VanLehn (eds) Proc. 5th Int. Conf. ITS 2000, Springer-Verlag, Montreal, 2000, pp. 474-483.