KERMIT: a Constraint-based Tutor for Database Modeling

Pramuditha Suraweera, Antonija Mitrovic

Intelligent Computer Tutoring Group
Computer Science Department, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
pramu16@hotmail.com, tanja@cosc.canterbury.ac.nz

Abstract: KERMIT is an intelligent tutoring system that teaches conceptual database design using the Entity-Relationship data model. Database design is an open-ended task: although there is an outcome defined in abstract terms, there is no procedure to use to find that outcome. So far, constraint based modelling has been used in a tutor that teaches a database language (SQL-Tutor) and a system that teaches punctuation and capitalisation rules (CAPIT). Both systems have proved to be extremely effective in evaluations performed in real classrooms. In this paper, we present experiences in using CBM in an open-ended domain. We describe system's architecture and functionality. KERMIT has also been evaluated in the context of genuine teaching activities. We present the results of an evaluation study with students taking a database course, which show that KERMIT is an effective system. The students enjoyed the system's adaptability and found it a valuable asset to their learning.

1. Introduction

In previous work, we have shown that Constraint-Based Modeling (CBM) [14] is extremely effective. We have implemented SQL-Tutor [11], an Intelligent Tutoring System (ITS) for the SQL database language, and CAPIT [10], a punctuation and capitalization tutor. This paper presents our experiences in implementing another constraint-based tutor, this time in the area of database design. This domain is different from the ones we have previously worked in, as it is an open-ended domain. Although the final database design is described in abstract terms (i.e. the features of a good quality design are known generally), there is no procedure to use to arrive at the final solution. We therefore wanted to test CBM in such a domain.

The Entity-Relationship (ER) data model, proposed by Chen [3], is the most widely used model for conceptual database design. Although the ER model is relatively simple, students have many problems developing ER diagrams. The text of the problem is often ambiguous and incomplete. ER modelling is not a well-defined process. There is no single best solution for a problem, and often there are several possible schemas for the same requirements. Although the traditional method of learning ER modelling in a classroom environment may be sufficient as an introduction to the concepts of database design, students cannot gain expertise in the domain by attending lectures only. In tutorials, a single tutor must cater for the needs of the entire group of students, and it is inevitable that they obtain only limited personal assistance. Therefore, the existence of a computerized tutor, which would support students in acquiring database design skills, would be highly important.

We start by reviewing related work. Section 3 describes the overall architecture of the system. Section 4 presents the evaluation study that showed the effectiveness of the system. The conclusions are given in the last section.

2. Related Work

There have been only two attempts at developing ITSs for DB modelling. ERM-VLE [9] is a text-based virtual learning environment for ER modelling, in which students design databases by navigating the virtual world and manipulating objects. The virtual world consists of different rooms, such as entity creation rooms and relationship creation rooms. The authors claim that the organisation of the environment reflects the task structure. The student issues commands such as *pick up*, *drop*, *name*, *evaluate*, *create* and *destroy* to manipulate objects. The effect of a command is determined by the location in which it was issued. For example, a student creates an entity whilst in the entity creation room.

The interface of ERM-VLE contains the definition of the problem, and a graphical representation of the solution, but the student does not directly interact with the graphical representation. The student interacts with the virtual world solely by issuing textual commands. The problem's ideal solution is embedded in the virtual world. The learner is only allowed to create objects that correspond to the ones in the ideal solution. When the system was evaluated, the experienced designers felt that the structure of the virtual world had restricted them [8]. On the other hand, novices felt that they had increased their understanding of ER modelling. However, these comments cannot be treated as a proof of the system's effectiveness since the system has not been evaluated properly.

ERM-VLE restricts the learner since he/she is forced to follow the identical solution path to the ideal one. This method has a high tendency to encourage shallow learning as users are prevented from making errors and they are not given explanation about their mistakes. Moreover, a text-based virtual reality environment is not a natural environment in which to construct ER models. Students who learn to construct ER models using ERM-VLE would struggle to become accustomed to modelling databases outside the virtual environment.

The other tutor for database modelling is COLER [5,6], a web-based collaborative learning environment for ER modelling. Students initially solve problems individually and then join a group to develop a group solution. The designers argue that this process helps to ensure that students participate in discussions and that they have the necessary raw material for negotiating differences with other members of the group. The student's individual solution is constructed in the private workspace, whereas the collaborative solution is created in the shared workspace. Students are provided with a chat window through which they can communicate with each other. The private workspace also allows the student to experiment with different solutions. Once a group of students agree to be involved in collaboratively solving a problem, the shared workspace is activated. Only a single member can edit the shared workspace at any time. After each change in the shared workspace, the students are required to express their opinions by voting, with either *agree*, *disagree* or *not sure*. The personal coach resident in the interface gives advice in the chat area based on the group dynamics: student participation and the group's ER model construction.

COLER encourages and supervises collaboration, and we believe it has the potential in helping students to acquire collaboration skills. However, it does not evaluate the ER schemas produced, and cannot provide feedback regarding their correctness. In this regard, even though the system is effective as a collaboration tool, the system would not be an effective teaching system for a group of novices with the same level of expertise. From the authors' experience, it is very common for a group of students to agree on the same flawed argument. Accordingly, it is very likely that groups of students unsupervised by an expert may learn flawed concepts of the domain. In order for COLER to be an effective teaching system, an expert should be present during the collaboration stage.

3. K*ER*MIT: A Knowledge-based ER Modelling Tutor

K*ER*MIT [12] is a problem-solving environment, in which students construct ER schemas that satisfy a given set of requirements. The system provides feedback tailored towards each student's knowledge. The system supports the ER model as defined in [7]. The architecture of the system is given in Figure 1. The main components of K*ER*MIT are its user interface, pedagogical module and student modeller, discussed in this section. K*ER*MIT contains a number of predefined database problems and ideal solutions, specified by a human expert. Each problem describes the requirements of a database that the student is to design. The problem text is represented internally with embedded tags that specify the mapping to the objects in the ideal solution. The tags are not visible to the student since they are extracted before the problem is displayed.

Users interact with K*ER*MIT's interface to construct ER schemas for the problems presented to them by the system. The pedagogical module drives the whole system by selecting the instructional messages and problems that best suit the particular student. The student modeller evaluates the student's solution. In contrast to typical ITSs, K*ER*MIT does not have a problem solver, as developing a problem solver for ER modelling is extremely difficult. One of the major obstacles that would have to be overcome is natural language processing (NLP), as the problems in the domain are presented using natural language text. NLP would have to be used to extract the requirements of the database from the problem text. However, the NLP problem is far from being solved. Other complexities arise from the nature of the task. There are assumptions that

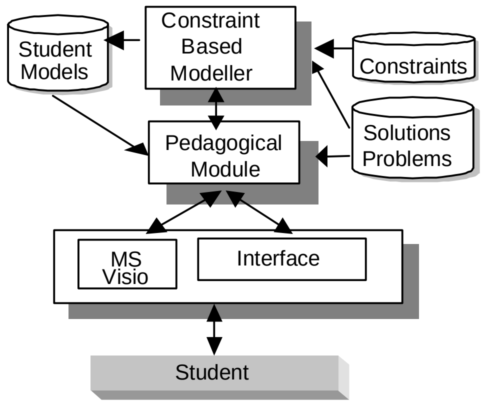


Fig. 1. Architecture of K*ER*MIT

need to be made during the composition of an ER schema. These assumptions are outside the problem description and are dependent on the semantics of the problem itself. Although this obstacle can be avoided by explicitly specifying these assumptions within the problem description, ascertaining these assumptions is an essential part of the process of constructing a solution and would over simplify the problems.

Although there is no problem solver, K*ER*MIT is able to diagnose students' solutions by using its domain knowledge represented as a set of constraints. The system contains an ideal solution for each of its problems, which is compared against the student's solution according to the system's knowledge base. 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. K*ER*MIT's knowledge base enables the system to identify student solutions that are identical to the system's ideal solution. More importantly, this knowledge also enables the system to identify alternative correct solutions, i.e. solutions that are correct but not identical to the system's solution.

K*ER*MIT's knowledge base consists of 92 constraints. Each constraint consists of a relevance condition, a satisfaction condition and feedback messages. The feedback messages are used to compose hints that are presented to students when the constraint is violated. The constraints can be roughly divided into syntactic and semantic ones. The syntactic constraints describe the syntactically valid ER schemas and are used to identify syntax errors in students' solutions. These constraints only deal with the student's solution. They vary from simple constraints such as "an entity name should be in upper case", to more complex constraints such as "the participation of a weak entity in the identifying relationship should be total".

Semantic constraints compare the student's solution to the ideal one. These constraints are usually more complex than syntactic constraints. For example. constraint 67 deals with composite multivalued attributes. Since such attributes can also be modelled as weak entities, the constraint has to compare a composite multivalued attribute in the ideal solution to a similar one or a weak entity in the student's solution. This constraint illustrates the ability of the system to deal with alternative correct student solutions that are different from the ideal solution specified by a human expert. K*ER*MIT knows about equivalent ways of solving problems, and it is this feature of the knowledge base that gives K*ER*MIT considerable flexibility.

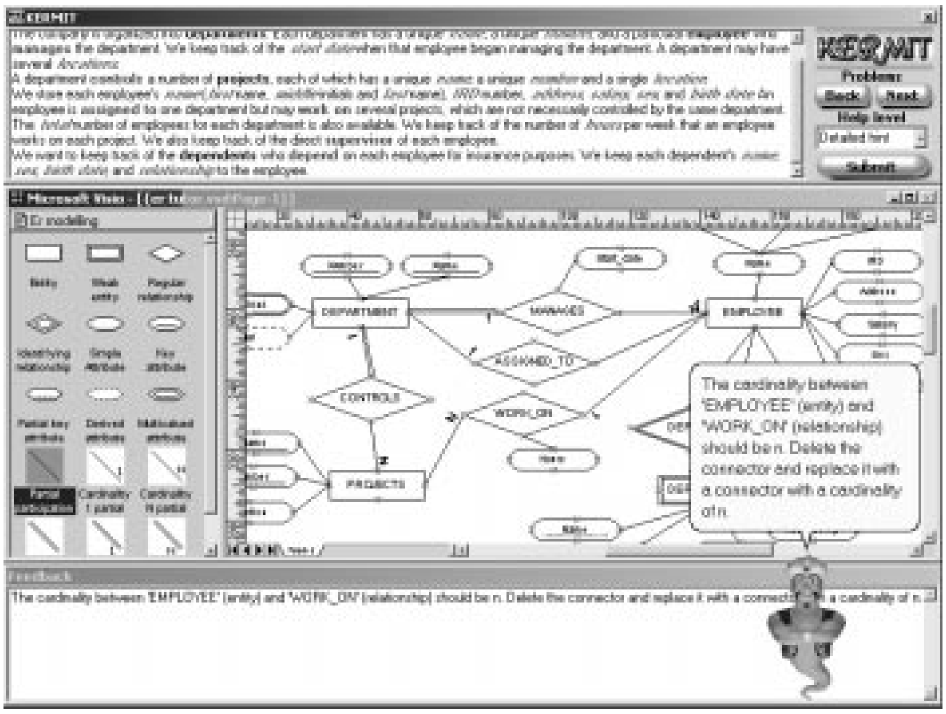
K*ER*MIT maintains two kinds of student models: short-term and long-term ones. Short-term models are generated by matching student solutions to constraints and the ideal solutions. The student modeller iterates through each constraint, checking whether the current problem state satisfies its relevance condition. If that is the case, the satisfaction component of the constraint is also verified against the current problem state. Violating the satisfaction condition of a relevant constraint signals an error. The pedagogical module uses the short-term student model to generate feedback to the student. On the other hand, the long-term student model is implemented as an overlay model. It keeps a record of each constraint's history: how often the constraint was relevant, and how often it was satisfied or violated. The pedagogical module uses these data to select new problems.

3.1. Interface

Students interact with K*ER*MIT via its user interface (Figure 2) to view problems, construct ER diagrams, and view feedback. The top window displays the text of the current problem. The middle window is the ER modelling workspace where students create ER diagrams. The workspace was developed by integrating *Microsoft Visio* [15] with K*ER*MIT. Feedback is presented in the textual form in the lowest window, and also verbally, through the animated pedagogical agent.

K*ER*MIT's interface reduces the burden on the student's memory by showing the text of the problem, and also by showing the available constructs. The student can easily remind her/himself of the elements of the problem and the concepts of the ER model. Furthermore, this interface reinforces ER modelling by requiring the student to highlight the appropriate part of the problem text whenever a new construct is added to the ER diagram. The highlighted words are coloured depending on the type of object. When the student highlights a phrase as an entity name, the highlighted text turns bold and blue. Similarly the highlighted text turns green for relationships and pink for attributes. The feature is advantageous from a pedagogical point of view, as the student must follow the problem text closely. Many of the errors in students' solutions occur because they have not comprehensively read and understood the problem. These mistakes would be minimised in K*ER*MIT, as students are required to focus their attention on the problem text every time they add a new object.

Besides being useful from the pedagogical point of view, highlighting is also


The company is organized into **departments**. Each department has a unique *name*, a unique *number*, and a particular **employee** who manages the department. We keep track of the *start date* when that employee began managing the department. A department may have several *locations*.
A department controls a number of **projects**, each of which has a unique *name*, a unique *number* and a single *location*.
We store each employee's *name* (*first name*, *middle initials* and *last name*), *IRD number*, *address*, *salary*, *sex* and *birth date*. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. The *total* number of employees for each department is also available. We keep track of the number of *hours* per week that an employee works on each project. We also keep track of the direct supervisor of each employee.
We want to keep track of the **dependents** who depend on each employee for insurance purposes. We keep each dependent's *name*, *sex*, *birth date* and *relationship* to the employee.

The cardinality between 'EMPLOYEE' (entity) and 'WORK_ON' (relationship) should be n. Delete the connector and replace it with a connector with a cardinality of n.

The cardinality between 'EMPLOYEE' (entity) and 'WORK_ON' (relationship) should be n. Delete the connector and replace it with a connector with a cardinality of n.

Fig. 2. User interface of K*ER*MIT

useful from the point of view of the student modeller for evaluating solutions. There is no standard that is enforced in naming entities, relationships or attributes, and the student has the freedom to use any synonym or similar word/phrase as the name of a particular object. Since the names of the objects in the student solution (SS) may not match the names of construct in the ideal solution (IS), the task of finding a correspondence between the constructs of the SS and IS is difficult. This problem is avoided in K*ER*MIT by forcing the student to highlight the word or phrase that is modelled by each object in the ER diagram.

3.2. Pedagogical Module

The pedagogical module 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. K*ER*MIT individualises both these actions to each student based on their student model.

There are are six levels of feedback, 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 not. 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. A list of hints on all violated constraints is displayed at the *all errors* level. The ER schema of the complete solution is displayed at the final level (*solution* level).

When the student gets a new problem, the feedback level is set to *correct*. The level of feedback is incremented with each submission until it reaches the *detailed hint* level. The system also gives the student the freedom to manually select the level of feedback, thus providing a better feeling of control.

When selecting a new problem, the student model is examined to find the constraints that have been violated most often. We have chosen this simple problem selection strategy in order to ensure that students get the most practice on the constructs with which they experience difficulties.

4. Evaluation

We performed an evaluation of K*ER*MIT at the University of Canterbury, Christchurch in August 2001. This section presents the procedure and the results.

4.1. Procedure

The study involved sixty-two volunteers from students enrolled in the Introduction to Databases course. 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. This study involved a comparison of a group of students learning ER modelling by using the fully functional K*ER*MIT against a control group who used a cut-down version of the system, referred to as ER-Tutor. The interfaces of both systems were similar, but ER-Tutor did not provide any feedback except for the complete solution. That way, both groups of students worked in the lab, but the ER-Tutor group had the feedback that is comparable to a classroom condition. The participants were randomly allocated to the control or the experimental group. Initially each student sat a pre-test and then interacted with the system in a single, two-hour session. The participants worked individually, solving problems at their own pace. There were 6 problems in total, presented to the two groups in the same order. All the important events such as logging in, submitting a solution and requesting help were recorded in a log specific to each student. Finally, the participants were given a post-test and a questionnaire.

A pre- and post-test were used to evaluate the students' knowledge before and after the session. To minimise any prior learning effects, we designed two tests (A and B) of approximately the same complexity. They contained two questions: a multiple choice question to choose the ER schema that correctly depicted the given scenario and a question that involved designing a small ER schema. In order to reduce any bias, the first half of each group was given test A as the pre-test and the remainder were given B as the pre-test. The students who had test A as their pre-test were given test B as their post-test and vice versa.

The questionnaire contained a total of fourteen questions. Initially students were questioned on previous experience in ER modelling and in using CASE tools. 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.

4.2. Objective Analysis

Table 1 presents a few statistics about the study. The experimental group students spent more time interacting with the system than the control group. Although the difference is not significant, it is encouraging to note that students were more willing to interact K*ER*MIT. The average times for completing a problem for both groups were very similar. These findings suggest that even though the students using K*ER*MIT were forced to indicate the semantic meaning of each construct by highlighting a word in the problem text, their performance was not degraded.

Table 1. Mean system interaction details

	K*ER*MIT		**ER-Tutor**	
	Mean	**s. d.**	**mean**	**s. d.**
Time spent on problem solving (min.)	66:39	21:22	57:58	34:38
Time spent per completed problem (min.)	23:36	6:55	23:46	21:40
No. of attempted problems	4.36	1.45	4.10	2.55
No. of completed problems	1.75	1.14	1.97	1.20

We also analysed the logs to see how students acquired constraints, using the same approach as in [13]. We identified each problem-state in which a constraint was relevant. Each constraint relevance occasion was rank ordered from 1 up. For each occasion, we recorded whether a relevant constraint was satisfied or violated. We calculated, for each participant, the probability of violating each individual constraint on the first occasion of application, the second occasion and

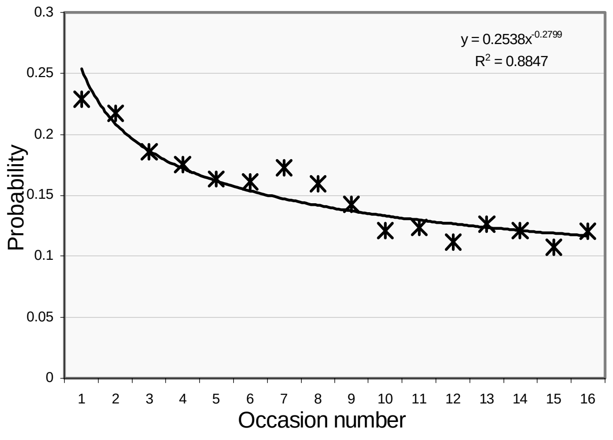


Fig. 3. Probability of violating a constraint as a function of the occasion when that constraint was relevant

so on. The probabilities were averaged across all the constraints in order to obtain an estimation of the probability of violating a given constraint C on a given occasion. The probabilities were then averaged across all participants and plotted as a function of the number of occasions when C was relevant, as shown in Figure 3. To reduce individual bias, only the occasions in which at least two thirds of the total population of participants had a relevant constraint were used. The data points show a regular decrease. The power curve displays a close fit with an R^2 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. The results of the mastery of constraints further strengthen the claim that the students learn ER modelling by interacting with K*ER*MIT.

4.3. Questionnaire Analysis

Table 2 displays a summary of the responses. The students in both groups required approximately the same time to learn the interface. Since K*ER*MIT's interface is more complicated, we expected that students who used K*ER*MIT would require longer to learn its interface. The difference in mean responses on the amount learnt is not significant. Both groups rated their enjoyment of the system on a similar scale. The control group students rated the interface easier to use in comparison to the students who used K*ER*MIT. The difference is statistically significant ($t = 1.78$, $p < 0.01$). This result was expected since K*ER*MIT's interface is more complex than ER-Tutor's.

The difference for the ratings of the usefulness of feedback is statistically significant ($t = 3.45$, $p < 0.01$). These results are analogous with our expectations due to the difference in the information content presented as feedback from each system. Students who used K*ER*MIT also had a better perception of the system as a whole. This was shown in their responses to whether they would recommend the system to others, where approximately 84% of the experimental group students indicated that they would, while the percentage of the control group students who had the same opinion was lower, approximately 68%.

Table 2. Means for the user questionnaire, pre- and post-test

	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
Pre-test	16.16	1.82	16.58	2.86
Post-test	17.77	1.45	16.48	3.08
Gain score	1.65	1.72	-0.10	2.76

4.4. Pre- and Post-test performance

Table 2 also contains the results the students achieved in the pre- and post-tests. The difference in pre-test scores is insignificant, confirming that the two groups are comparable. The experimental group scored significantly higher on the post-test ($t = 4.91$, $p < 0.01$). Conversely, the difference in pre- and post-test of the group who used ER-Tutor is statistically insignificant. The difference in gain scores of the two groups is statistically significant ($t = 3.07$, $p < 0.01$). We can conclude from these results that students who used K*ER*MIT learnt more than students who used the control system.

The effect size and power for the experiment were also calculated. Effect size is a standard method of comparing the results of one pedagogical experiment to another. The common method to calculate the effect size in the ITS community is to subtract the control group's mean gains score from the experimental group's mean gain score and divide by the standard deviation of the gain scores of the control group [2]. This gives the effect size of 0.63 for our experiment, which is comparable with the effect size of 0.63 in [1] and 0.66 in [11], where the session lengths were also 2 hours.

Another way to calculate the effect size is the ω^2 value [4]. For our experiment, ω^2 is 0.12, which is a relatively large effect size. We also calculated the power, measured as the fraction of experiments that would produce significant results for the same design, the same number of participants and the same effect size. Chin [4] recommends that researchers should strive for a power of 0.8. The power of this experiment was calculated as 0.75 at significance 0.05, which is an excellent result.

5. Conclusions

This paper presented K*ER*MIT, an ITS for ER modelling. K*ER*MIT's effectiveness in teaching ER modelling was shown in a classroom experiment. The participants who used the full version of K*ER*MIT showed significantly better results in both the subjective and objective analysis in comparison to the students who practiced ER modelling with a conventional drawing tool.

The student modelling technique used in K*ER*MIT (CBM) has previously been used to represent domain and student knowledge in SQL-Tutor [11,13] and in CAPIT [10]. In both cases, the analysis of students' behaviour while interacting with these systems proved the sound psychological foundations of CBM and the appropriateness of constraints as the basic units of knowledge. The research presented in this paper demonstrated that CBM can also be used to effectively represent knowledge in domains with open-ended tasks such as database modelling. This result further strengthens the credibility of CBM.

There are a number of future avenues that can be explored to further improve K*ER*MIT. The current system only presents general hint messages on the errors in the student's solution. The feedback of the system could be enhanced to provide support for deep learning. We have recently started a new project, which will enhance K*ER*MIT to support self-explanation.

Acknowledgements
The work presented here was supported by the University of Canterbury research grant U6430. We thank Ken Koedinger for advising on the evaluation study.

References

1. Albacete, P.L., VanLehn, K. The Conceptual Helper: an Intelligent Tutoring System for Teaching Fundamenatal Physics Concepts. In: Gauthier, G., Frasson, C. and VanLehn, K. (eds.). Proc. ITS'2000, Springer-Verlag Berlin (2000) 564-573
2. Bloom, B. S. The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational Researcher, 13 (1984) 4-16
3. Chen, P. P. The Entity Relationship Model - Toward a Unified View of Data. ACM Transactions Database Systems, 1 (1976) 9-36
4. Chin, D. N. Empirical Evaluation of User Models and User-adapted Systems. User Modeling and User Adapted Interaction, 11, (2001) 181-194
5. Constantino-Gonzalez, M., Suthers, D. A Coached Collaborative Learning Environment for Entity-Relationship Modeling. In: Gauthier, G., Frasson, C. and VanLehn, K. (eds.). Proc. ITS'2000, Montreal (2000) 324-333
6. Constantino-Gonzalez, M., Suthers, D., Icaza., J. Designing and Evaluating a Collaboration Coach: Knowledge and Reasoning. In: Moore, J. D., Redfield, C. L. and Johnson, W. L. (eds.). Proc. AIED 01, San Antonio, Texas, IOS Press (2001) 176-187
7. Elmasri, R., Navathe, S. B. Fundamentals of Database Systems. Addison Wesley (1994)
8. Hall, L., Gordon, A. Synergy on the Net: Integrating the Web and Intelligent Learning Environments. Proc. of WWW-based Tutoring Workshop at ITS'2000 (2000) 25-29
9. Hall, L., Gordon, A. (1998) A Virtual Learning Environment for Entity Relationship Modelling. SIGCSE bulletin, 30 (1998) 345-353.
10. Mayo, M., Mitrovic, A. Optimising ITS Behaviour with Bayesian Networks and Decision Theory. Int. Journal on Artificial Intelligence in Education, 12 (2001) 124-153.
11. Mitrovic, A., Martin, B., Mayo, M. Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. UMUAI, 12 (2002) (in press)
12. Mitrovic, A., Mayo, M., Suraweera, P., Martin, B. Constraint-based Tutors: a Success Story. In: Monostori, L., Vancza, J. and Ali, M. (eds.). Proc. IEA/AIE-2001, Budapest, Springer-Verlag Berlin (2001) 931-940
13. Mitrovic, A., Ohlsson, S. Evaluation of a Constraint-based Tutor for a Database Language. Int. Journal on Artificial Intelligence in Education, 10 (1999) 238-256
14. Ohlsson, S. Constraint-based Student Modelling. In: Greer, J.E., McCalla, G (eds) Proc. of Student Modelling: the Key to Individualized Knowledge-based Instruction, Springer-Verlag Berlin (1994) 167-189
15. Visio, http://www.microsoft.com/office/visio/