Int J Artif Intell Educ (2016) 26:448–456
DOI 10.1007/s40593-015-0084-6

ARTICLE

Teaching Database Design with Constraint-Based Tutors

Antonija Mitrovic[1] · **Pramuditha Suraweera**[2]

Published online: 2 December 2015

Abstract Design tasks are difficult to teach, due to large, unstructured solution spaces, underspecified problems, non-existent problem solving algorithms and stopping criteria. In this paper, we comment on our approach to develop KERMIT, a constraint-based tutor that taught database design. In later work, we re-implemented KERMIT as EER-Tutor, and extended its instructional domain. Several evaluation studies performed with KERMIT and EER-Tutor show that they are effective Intelligent Tutoring Systems (ITSs). We also comment on various extensions made to EER-Tutor over the years. There are several contributions of our research, such as developing effective problem-solving support for conceptual database design in terms of interface design. Our database design tutors deal with large solution spaces efficiently by specifying constraints that capture equivalent solution states, and using ideal solutions to capture the semantics of the problem. Instead of requiring a problem solver, the ITS checks whether the student's database schema is correct by matching it to constraints and the ideal solution. Another contribution of our work is in guidelines for developing effective feedback to the student.

Keywords KERMIT · EER-Tutor · Teaching design tasks · Ill-defined tasks

Introduction

Work on a constraint-based tutor to teach database design started in 2000. The motivation for this work was twofold. Firstly, at the time, Mitrovic was teaching (and still does) an introductory course on relational databases. The course includes a large component on database design, including conceptual database design using the Entity-Relationship

✉ Antonija Mitrovic
Tanja.mitrovic@canterbury.ac.nz

Pramuditha Suraweera
pramudi@gmail.com

[1] Intelligent Computer Tutoring Group, Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand

[2] Research Insights and Analytics, Chief Marketing Office, Telstra, Australia

(ER) model (Chen 1976). Mitrovic observed that students had many difficulties learning ER modeling. The second motivation for developing KERMIT was to continue our work on Constraint-Based Modeling (CBM) (Ohlsson 1992). We start the paper by discussing these two motivations.

Mitrovic and Weerasinghe (2009) made a distinction between instructional domains and instructional tasks when discussing ill-definedness. The ER model itself is well-defined; it consists of only a handful of constructs with well-defined syntax. However, conceptual database design using the ER model is an ill-defined task; it consists of mapping database requirements, usually provided to students in the form of English text, to ER diagrams. Database requirements are usually underspecified, and can be ambiguous. In order to understand them fully, students need a good understanding of the application domain, common sense and background knowledge, which are often missing. Furthermore, the outcome (i.e. the ER schema) is defined in abstract terms: the student is required to develop a high quality schema that meets the requirements. Database design is a demanding task, as there is no algorithm students can use to produce ER schemas. Additionally, the stopping criterion (i.e. knowing when the solution is reached) is unclear – how can a student evaluate the quality of the produced ER diagram? It is possible for the student to check the syntactic correctness of the schema by making sure that all components satisfy the integrities of the ER model, although even this task is complex for students new to database design. Checking semantic correctness is much harder, as it requires the student to make sure the produced database schema covers requirements completely and that it is of high quality.

All of these features of database design are typical for other design tasks. Several criteria have been identified to describe design tasks in general (Goel and Pirolli 1988, 1992; Reitman 1964). Design tasks require extensive domain expertise, use of artificial symbol systems and incremental development. These tasks are characterized by underspecified start and goal states and problem-solving algorithms, large solution spaces, lack of operators for changing states, large solutions, and lack of a definite test to decide whether the goal has been attained, and consequently, there is no best solution, but rather a family of solutions for each problem. Lynch et al. (2009) discuss alternative definitions of ill-defined domains and problems, and suggest similar definitions to those proposed in (Mitrovic and Weerasinghe 2009).

The other motivation for developing KERMIT was to further research on CBM in terms of its applicability in various instructional domains. Before KERMIT, Mitrovic and colleagues developed SQL-Tutor, the first constraint-based tutor which teaches students how to query relational databases (see Mitrovic and Ohlsson 2015), and CAPIT, a constraint-based tutor which teaches elementary school children about punctuation and capitalization rules in English (Mayo and Mitrovic 2001). These two tutors proved that CBM is an effective student modeling approach, and answered many questions raised by Ohlsson when he originally proposed CBM in his 1992 paper (see also Ohlsson's commentary in this issue, 2015). ER modeling differed sufficiently from the previous two instructional domains to be interesting from this point of view. Punctuation and capitalization rules in English are very straightforward in comparison to ER modeling. Writing queries in SQL is an ill-defined design task, and therefore has some similarities with ER modeling. However, solutions are more structured in SQL than in ER modeling; queries consist of six clauses, thus defining the elementary structure for the solution space, while in the case of ER

diagrams, there is no pre-specified solution structure (apart from solutions having to contain entities, relationships, attributes and other components of the ER data model).

We start the paper by briefly discussing the highly cited IJAIED paper on KERMIT in the next section, and then discuss EER-Tutor, an enhanced Web-enabled version of the tutor. EER-Tutor enabled many exciting research projects, which investigated meta-cognitive skills, affect-sensitive pedagogical agents, and tutorial dialogues. We present those projects in the last section.

KERMIT: A Knowledge-Based ER Modeling Tutor

Similar to other constraint-based tutors, KERMIT was designed as a complement to database courses; we assumed that the student has already acquired some knowledge via lectures and labs. KERMIT allowed students to practice, by providing database requirements for which students developed ER diagrams. The system, presented in Fig. 1, was originally developed in Visual Basic as a stand-alone application. The student was given the database requirements for the chosen problem at the top, and could use various tools to draw ER components on the drawing pane. When the student submitted a solution, the system analysed it, and provided feedback on identified mistakes (if any). Students could request more detailed feedback messages depending on their needs. The animated pedagogical agent (the Genie, implemented using Microsoft Agent) presented feedback verbally, using audio and speech bubbles.

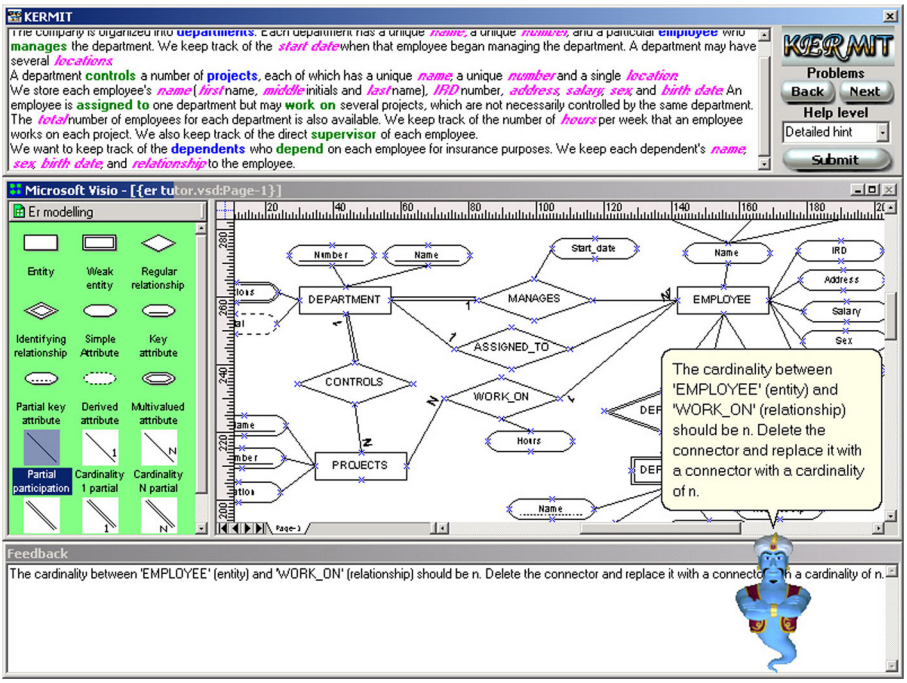

Fig. 1 Screenshot of the KERMIT's interface

KERMIT analyses the submitted solution by matching it to the constraint set (which contained 92 constraints) and the pre-specified ideal solution (Suraweera and Mitrovic 2004). There are two types of constraints: the syntactic constraints check whether the student's solution satisfies the syntax of the ER model, while the semantic constraints check whether the student's solution matches the problem requirements by comparing it to the ideal solution. KERMIT stored only one correct solution per problem (specified by the human teacher). For all but the simplest problems, there could be several correct solutions and an infinite number of incorrect solutions, resulting in huge solution spaces. The ideal solution captures the semantics of the problem without the need for a problem solver (which would be difficult, if not impossible, to develop). Semantic constraints ensure that alternative correct solutions would be recognized as such, by looking for equivalent ways of specifying ER components (Suraweera and Mitrovic 2004). Therefore, constraints and ideal solutions allowed KERMIT to deal with huge solution spaces efficiently.

There were several challenges we faced during design and development of KERMIT. We wanted to support learning, and therefore to lower the working memory load; to achieve that, we showed the full problem text to the student, as well as the set of tools to draw diagrams. When a student creates a new diagram component, he/she needs to specify its name; in KERMIT, the name must be a word or a phrase from the problem text. Although there has been some criticism of this decision as being overly restrictive, we still believe it is beneficial for students' learning, for several reasons. From our own teaching experience, we observed that student often underline words or phrases from the problem text using different colours when they study requirements – therefore, using a similar approach in the ITS is not unnatural. Furthermore, using different colours for different component types makes it easier for the student to go over the problem text and see how much of it is already covered, and what is still outstanding. Another important reason in favour of such naming policy comes from Software Engineering, where the use of the customer's language is strongly encouraged. By using parts of the problem text, we prevent students from inventing their own labels, which might be difficult for anyone else to interpret. Finally, this decision made the interpretation of students' solutions much easier; we know exactly what each component represents, and avoid problems with natural language understanding which we would face if the students named components freely.

The 2004 paper presented the approach taken to develop KERMIT, as well as the findings from the pilot and evaluation studies conducted in 2001, which showed that KERMIT was highly effective. The full study compared the experimental group (26 participants using KERMIT) to the control group (31 participants) who used a version of the system with limited feedback (no feedback was given on the student's solution, only the full solution was provided after each problem). There was a statistically significant difference in the learning gains (post-test score – pre-test score) of the two groups, with the effect size d of 0.63, after students spent an average of only 66 min with the system. The students liked the system, and rated its feedback highly.

Expanding KERMIT

After showing that CBM is an effective approach to teach conceptual data modeling, we turned to other interesting research questions, such as modelling and supporting

metacognitive skills. Open learner models have been proposed as tools to support metacognitive activities, such as reflection and self-assessment (Bull and Kay 2007, 2010). Hartley and Mitrovic (2002) reported on an extension of KERMIT (named e-KERMIT) with two visualizations of the student model. We added skill meters to the interface (circled in Fig. 2), which presented a high-level summary of the student model in terms of the student's knowledge of the ER notation and ability to identify entities, attributes and relationships. Skill meters presented continuous feedback on progress, and also served as an aid to remind and motivate students to further inspect their models. A more detailed, hierarchical open model (Fig. 3) was available on request, when the student clicked on the *Show Me More* button. The hierarchical open student model presented a visualization of the student's knowledge on a finer granularity level; for each domain concept in the hierarchy, the model shows the student's progress in terms of the percentage that the student covered (i.e. the percentage of corresponding constraints the student used) and the percentage mastered (i.e. the percentage of corresponding constraints the student used correctly). The student could expand a concept to show concepts on the lower level of the category (e.g. *Type* under *Attribute identification* in Fig. 3).

Besides summarizing the student's knowledge, the open model also presented a high level view of the instructional domain, which supported the student's understanding of the domain structure. When students inspected the open models, they could reflect on

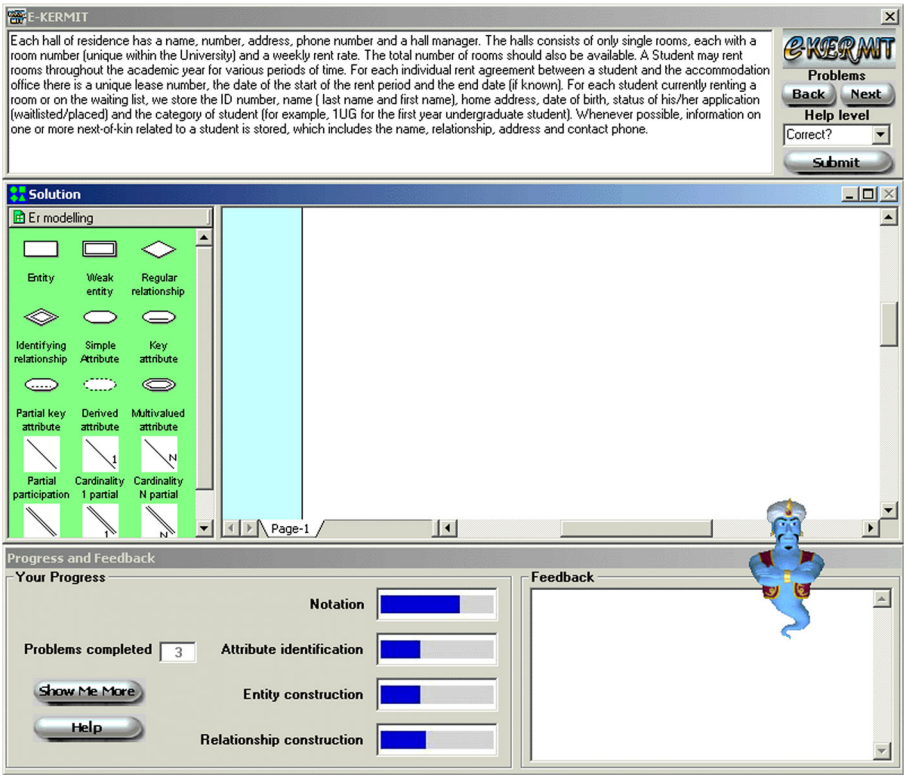

Fig. 2 The interface of E-KERMIT

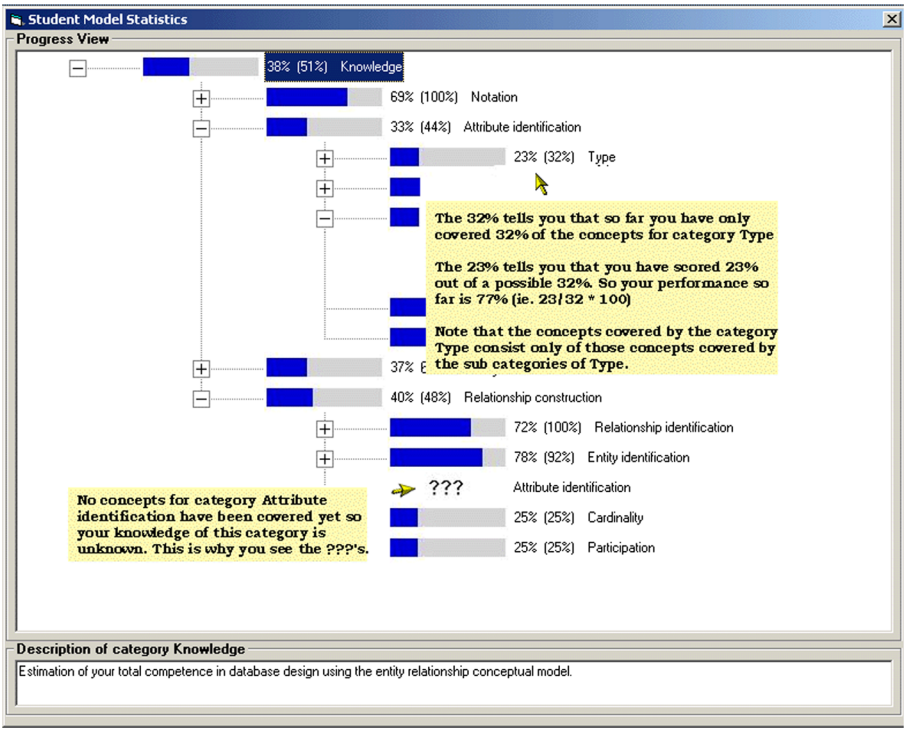

Fig. 3 The expanded open student model

their knowledge and reconsider their beliefs about domain concepts. By providing such visualizations, the student model was not just a source of knowledge about the student valuable to the system, but became an important learning resource on its own. The study that compared the standard version of KERMIT to e-KERMIT showed that the open student model had a positive effect on learning, and that the majority of the students explored the expanded open student model and found it an effective additional tool for learning (Mitrovic and Martin 2007). Later on, we conducted studies on additional visualizations of student models (Duan et al. 2010; Mathews et al. 2012).

After the initial success with KERMIT, we developed EER-Tutor, a Web-enabled tutor. Figure 4 shows its interface, which has many similarities to the original version. The requirements are shown at the top of the interface, below which is the toolbar showing the tools corresponding to the components of the EER model. Students can submit their solutions whenever they want, after which they get feedback shown in the right pane of the interface. The system also highlights in red incorrect parts of the solution in the drawing area.

EER-Tutor was developed in WETAS, the authoring shell developed by Brent Martin (Martin and Mitrovic 2002). It is one of our three tutors that have been available on Addison-Wesley's DatabasePlace Web portal, and used by more than 10,000 students worldwide. EER-Tutor supports the Enhanced Entity Relationship model as defined by Elmasri and Navathe (2011), which extends the basic ER model by introducing specializations and categories. The current version of EER-Tutor contains 57 problems, and over 200 constraints.

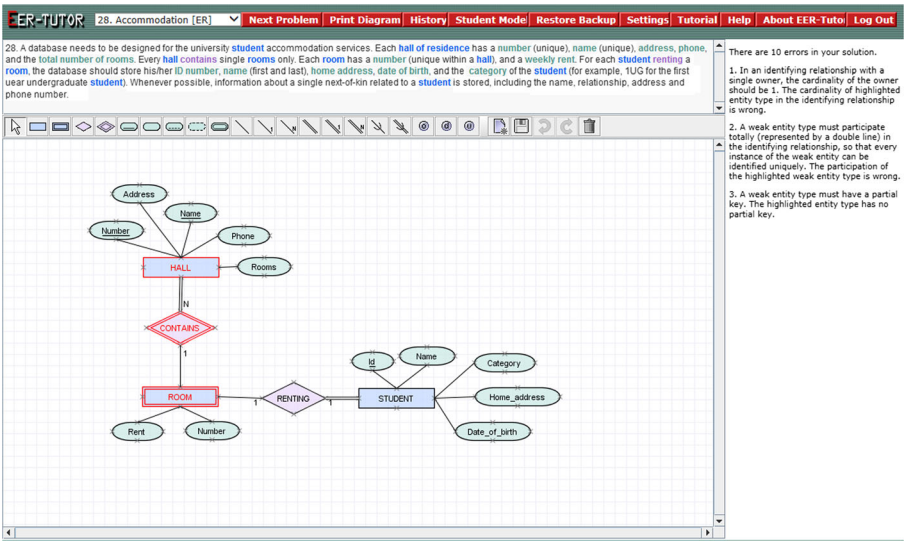

Fig. 4 The interface of EER-Tutor

We conducted several other projects in the context of EER-Tutor. In one of them, we wanted to explore further the effect of feedback. Usually feedback for ITSs is defined by designers by using their intuition, which was the case with the feedback provided by KERMIT and the first version of EER-Tutor. However, the theory of learning from performance errors (Ohlsson 1996), on which CBM is based, states that the role of the feedback should be to identify the error and explain what constitutes the error (blame assignment), as well as to re-iterate the domain principle violated by the student's solution (remediation). Therefore, we re-engineered feedback for EER-Tutor and conducted a study comparing EER-Tutor to a version of the system that provided theory-based feedback. The results of evaluation showed that theory-based feedback is more effective in supporting learning: students who received such feedback learned constraints faster than the students who received the original EER-Tutor feedback (Zakharov et al. 2005).

EER-Tutor paved the way for further research; its code was used as the foundation to develop COLLECT-UML, a constraint-based system that teaches UML design and supports pairwise collaboration between students (Baghaei et al. 2007). Kon Zakharov developed an affect-sensitive agent for EER-Tutor (Zakharov et al. 2008), which tracked the student's affective states from the student's facial features and problem-solving actions, and responded to them by modifying its facial expressions and providing affect-sensitive feedback. For example, the agent encouraged the student to continue with problem solving when there was only one error found in the solution by saying "*Just a little more effort and you get there – it will make you feel great*!", or provided advice for a struggling student by saying "*It seems you are somewhat frustrated. Would it help if you started working on one of the simpler problems*?"

Amali Weerasinghe developed tutorial dialogues for KERMIT (Weerasinghe and Mitrovic 2003), and later a model for providing adaptive tutorial dialogues for EER-Tutor, effectively providing substep instruction (VanLehn 2011). Recently, that model

was extended to provide adaptive tutorial dialogs (Weerasinghe et al. 2009, 2011). The dialogs are selected adaptively, on the basis of the student model, by identifying the concepts that the student had most difficulties with, and then selecting the tutorial dialogs corresponding to those concepts. Additionally, there are adaptation rules which individualize the dialogs to suit the student's knowledge, in terms of the length of the dialog and the exact content of the dialog. In response to the generated dialog, learners are able to provide answers by selecting the correct option from a list provided by the tutor. In a study conducted in March 2010, the students who had adaptive dialogs outperformed their peers who only received non-adaptive dialogs, with the effect size 0.69 on learning gains after approximately 100 min of interaction with the system (Weerasinghe et al. 2010). The obtained effect size is remarkable because the only difference between the two groups was the adaptivity of the dialogues. Additionally, Myse Elmadani has looked more deeply into how students interact with tutorial dialogues in EER-Tutor by capturing and analyzing eye-tracking data (Elmadani et al. 2013).

Our papers on teaching database design and on SQL-Tutor also contributed to popularization of CBM, which has been used since by many researchers in addition to those coming from our research group – see (Mitrovic and Ohlsson 2015). Our future plans include enhancing EER-Tutor with other forms of learning, such as learning from examples and providing motivational support to students.

Acknowledgments The work reported here could not have been done without the support of other members of the Intelligent Computer Tutoring Group. We thank them all for the discussions and friendship over the years. We thank Brent Martin, Konstantin Zakharov, Moffat Mathews and Jay Holland for their work on EER-Tutor.

References

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *Computer-Supported Collaborative Learning*, *2*(2–3), 159–190.

Bull, S., & Kay, J. (2007). Student models that invite the learner in: the SMILI open learner modelling framework. *International Journal of Artificial Intelligence in Education*, *17*(2), 89–120.

Bull, S., & Kay, J. (2010). Open learner models. R. Nkambou, J. Bourdeau, R., Mizoguchi (Eds.) *Advances in intelligent tutoring systems* (pp. 301–322). Springer Berlin Heidelberg.

Chen, P. P. (1976). The entity relationship model - toward a unified view of data. *ACM Transactions Database Systems*, *1*, 9–36.

Duan, D., Mitrovic A., & Churcher, N. (2010) Evaluating the effectiveness of multiple open student models in EER-Tutor. S. L. Wong et al. (Eds.) Proc. 18th Int. Conf. Computers in Education (pp. 86–88).

Elmadani, M., Mitrovic, A., Weerasinghe, A. (2013) Understanding Student Interactions with Tutorial Dialogues in EER-Tutor. L. H. Wong, C-C Liu, T. Hirashima, P. Sumedi, M. Lukman (Eds.) *Proc. 21st Int. Conf. on Computers in Education* (pp. 30–40).

Elmasri, R., & Navathe, S.B. (2011) Fundamentals of Database Systems. Pearson Education, 6th edition.

Goel, V., & Pirolli, P. (1988). Motivating the notion of generic design with information processing theory: the design problem space. *AI Magazine*, *10*, 19–36.

Goel, V., & Pirolli, P. (1992). The structure of design problem spaces. *Cognitive Science*, *16*, 395–429.

Hartley, D., & Mitrovic, A. (2002) Supporting learning by opening the student model. In: S. Cerri, G. Gouarderes, F. Paraguacu (Eds.) *Proc. 6th Int. Conf. Intelligent Tutoring Systems*, LCNS 2363 (pp. 453–462).

Lynch, C., Ashley, K. D., Pinkwart, N., & Aleven, V. (2009). Concepts, structures, and goals: redefining ill-definedness. *Artificial Intelligence in Education*, *19*(3), 253–266.

Martin, B., & Mitrovic, A. (2002) WETAS: a Web-based authoring System for constraint-based ITSs. In: P. de Bra, P. Brusilovsky and R. Conejo (Eds.) *Proc. 2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web-based Systems*, LCNS 2347 (pp. 543–546).

Mathews, M., Mitrovic, A., Lin, B., Holland, J., & Churcher, N. (2012) Do your eyes give it away? Using eye tracking data to understand students' attitudes towards open student model representations. S.A. Cerri and B. Clancey (Eds.) *Proc. 11th Int. Conf. Intelligent Tutoring Systems* LCNS 7315 (pp. 424–429). Crete: Springer-Verlag.

Mayo, M., & Mitrovic, A. (2001). Optimising ITS behaviour with Bayesian networks and decision theory. *Artificial Intelligence in Education*, *12*(2), 124–153.

Mitrovic, A., & Martin, B. (2007). Evaluating the effect of open student models on self-assessment. *Artificial Intelligence in Education*, *17*(2), 121–144.

Mitrovic, A., & Ohlsson, S. (2015) Implementing CBM: SQL-tutor after fifteen years, *artificial intelligence in education*, *Artificial Intelligence in Education*, *25*(2).

Mitrovic, A., & Weerasinghe, A. (2009) Revisiting the ill-definedness and consequences for ITSs. Dimitrova, V., Mizoguchi, R., du Boulay, B., Graesser, A (Eds.) *Proc 14th Int. Conf. Artificial Intelligence in Education* (pp. 375–382).

Ohlsson, S. (1992). Constraint-based student modelling. *Artificial Intelligence in Education*, *3*, 429–429.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, *103*, 241–262.

Ohlsson, S. (2015). Constraint-based modeling: from cognitive theory to computer tutoring – and back again. *Artificial Intelligence in Education*. doi:10.1007/s40593-015-0075-7.

Reitman, W. R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M. W. Shelly, & G. L. Bryan (Eds.), *Human judgements and optimality*. New York: Wiley.

Suraweera, P., & Mitrovic, A. (2004). An intelligent tutoring system for Entity Relationship modelling. *Artificial Intelligence in Education*, *14*(3–4), 375–417.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems. *Educational Psychologist*, *46*(4), 197–221.

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* (pp. 512–514). IOS Press.

Weerasinghe, A., Mitrovic, A., & Martin, B. (2009). Towards individualized dialogue support for ill-defined domains. *Artificial Intelligence in Education*, *19*(4), 357–379.

Weerasinghe, A., Mitrovic, A., Zijl, M., & Martin, B. (2010). Evaluating the effectiveness of adaptive tutorial dialogues in database design. In S. L. Wong, et al. (Eds.), *Proc. 18th Int. Conf. Computers in Education* (pp. 33–40). APSCE.

Weerasinghe, A., Mitrovic, A., Thomson, D., Mogin, P., Martin, B. (2011) Evaluating a General Model of Adaptive Tutorial Dialogues. In: Biswas, G., Bull, S., Kay, J., Mitrovic, A. (Eds.), *Proc. 15th Int. Artificial Intelligence in Education* (pp. 394–402). Springer, LNAI 6738.

Zakharov, K., Mitrovic, A., & Ohlsson, S. (2005) Feedback micro-engineering in EER-Tutor. In: C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (Eds.) *Proc. Artificial Intelligence in Education* (pp. 718–725). IOS Press.

Zakharov, K., Mitrovic, A., & Johnston, L. (2008) Towards emotionally-intelligent pedagogical agents. B. Woolf et al. (Eds.) *Proc. Intelligent Tutoring Systems* (pp. 19–28).