From Modelling Domain Knowledge to Metacognitive Skills: Extending a Constraint-based Tutoring System to Support Collaboration

Nilufar BAGHAEI and Antonija MITROVIC

Department of Computer Science and Software Engineering
University of Canterbury, Private Bag 4800, Christchurch, New Zealand
{n.baghaei, tanja}@cosc.canterbury.ac.nz

Abstract. Constraint-based tutors have been shown to increase individual learning in real classroom studies, but would become even more effective if they provided support for collaboration. COLLECT-*UML* is a constraint-based intelligent tutoring system that teaches object-oriented analysis and design using Unified Modelling Language. Being one of constraint-based tutors, COLLECT-*UML* represents the domain knowledge as a set of constraints. However, it is the first system to also represent a higher-level skill such as collaboration using the same formalism. We started by developing a single-user ITS. The system was evaluated in a real classroom, and the results showed that students' performance increased significantly. In this paper, we present our experiences in extending the system to provide support for collaboration as well as problem-solving. The effectiveness of the system was evaluated in a study conducted at the University of Canterbury in May 2006. In addition to improved problem-solving skills, the participants both acquired declarative knowledge about good collaboration and did collaborate more effectively. The results, therefore, show that Constraint-Based Modelling is an effective technique for modelling and supporting collaboration skills.

1 Introduction

Constraint-based tutors are Intelligent Tutoring Systems (ITS) which use Constraint-Based Modelling (CBM) [15] to represent domain and student models. These tutors have been proven to provide significant learning gains for students in a variety of instructional domains. As is the case with other ITSs [4], constraint-based tutors are problem-solving environments; in order to provide individualized instruction, they diagnose students' actions, and maintain student models, which are then used to provide individualized problem-solving support and generate appropriate pedagogical decisions. Constraint-based tutors have been developed in domains such as SQL (the database query language), database modelling, data normalization [13], punctuation [11] and English vocabulary [10].

All constraint-based tutors developed so far support individual learning. This paper describes extending COLLECT-*UML* [1, 3], a constraint-based ITS, to support the acquisition of collaboration skills. COLLECT-*UML* teaches Object-Oriented (OO) analysis and design using Unified Modelling Language (UML). The system provides feedback on both collaboration issues (using the collaboration model, represented as a set of meta-constraints) and task-oriented issues (using the domain model, represented as a set of syntax and semantic constraints).

We start with a brief overview of related work in Section 2. The architecture of COLLECT-$\mathcal{UML}$ and its interface are discussed in Section 3. Section 4 describes the collaborative model, which has been implemented as a set of meta-constraints. In Section 5, we present the results of an evaluation study conducted recently. Conclusions are given in the last section.

2 Related Work

In the last decade, many researchers have contributed to the development of computer-supported collaborative learning (CSCL) and advantages of collaborative learning over individualised learning have been identified. Some particular benefits of collaborative problem-solving include: encouraging students to verbalise their thinking; encouraging students to work together, ask questions, explain and justify their opinions; increasing students' responsibility for their own learning; increasing the possibility of students solving or examining problems in a variety of ways; and encouraging them to elaborate and reflect upon their knowledge [17]. These benefits, however, are only achieved by active and well-functioning learning teams [8]. Various strategies for computationally supporting online collaborative learning have been proposed and used, but more studies are needed that test the utility of these techniques [9].

CSCL systems can be classified into three categories based on the collaboration support they provide [9]. The first category includes systems that reflect actions; this basic level of support makes students aware of each others' actions. The systems in the second category monitor the state of interactions; some of them aggregate the interaction data into a set of high-level indicators, and display them to the participants (e.g. Sharlok II [14]), while others internally compare the current state of interaction to a model of ideal interaction, but do not reveal this information to the users (e.g. EPSILON [18]). In the latter case, this information is either intended to be used later by a coaching agent, or analysed by researchers in order to understand the interaction [9]. Finally, the third class of systems offer advice on collaboration. The coach in these systems plays a role similar to that of a teacher. The systems can be distinguished by the nature of the information in their models, and whether they provide feedback on strictly collaboration issues or both social and task-oriented issues. An example of the systems focusing on the social aspects is Group Leader Tutor [12], while COLER [5] addresses both social and task-oriented aspects of group learning.

Although many tutorials, textbooks and other resources on UML are available, we are not aware of any attempt at developing a CSCL environment for UML modelling. However, there has been an attempt [18] at developing a collaborative learning environment for OO design problems using Object Modeling Technique (OMT), a precursor of UML. The system monitors group members' communication patterns and problem solving actions in order to identify situations in which students effectively share new knowledge with their peers while solving problems. The system dynamically assesses a group's interaction, and determines when and why the students are having trouble learning new concepts they share with each other. The system does not evaluate the OMT diagrams and an instructor or intelligent coach's assistance is needed in mediating group knowledge sharing activities. In this regard, even though the system is effective as a collaboration tool, it would probably

not be an effective teaching system for a group of novices with the same level of expertise, as the students may agree on the same flawed argument.

3 COLLECT-*UML*

COLLECT-*UML* is a problem-solving environment implemented in Allegro Common Lisp, in which students construct UML class diagrams that satisfy a given set of requirements. It assists students during problem solving, and guides them towards the correct solution by providing feedback. The system is designed as a complement to classroom teaching and when providing assistance, it assumes that the students are already familiar with the fundamentals of UML.

We started by developing a constraint-based tutoring system which supported students working individually. Being a Web-enabled system, its interface is delivered via a Web browser. The system consists of a session manager that manages sessions and student logs, a student modeller that maintains student models, the constraint set and a pedagogical module. We performed an evaluation study in a real classroom, and the results showed that students' performance increased significantly. For details on the architecture, functionality and the evaluation studies of this version please refer to [1, 3].

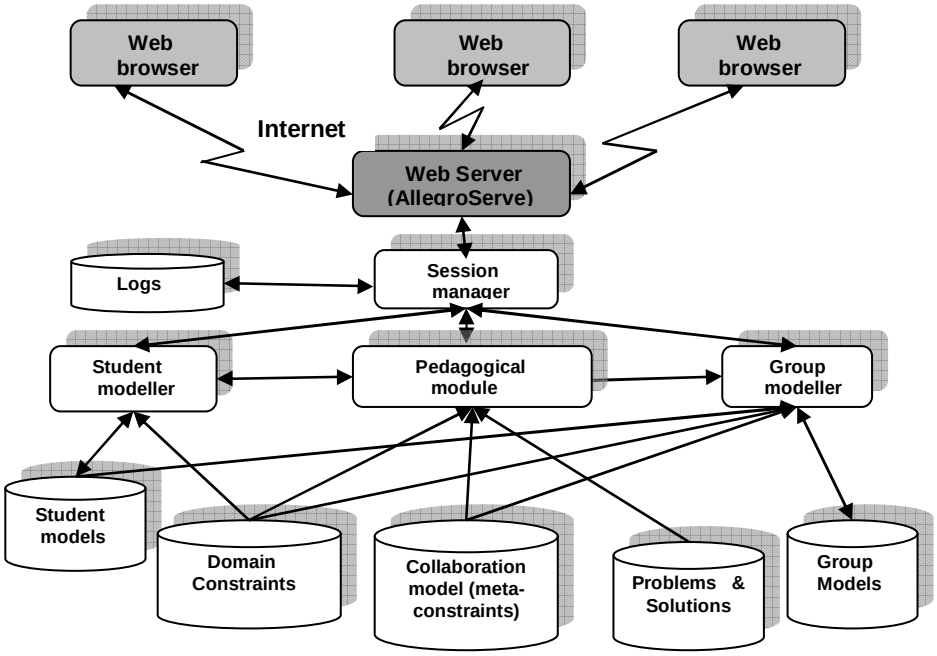

Fig. 1. The architecture of COLLECT-*UML*

The architecture of the collaborative version of the system (Figure 1) introduces the group modeller, a new component responsible for creating and maintaining group models. The pedagogical module uses both the student model and the group model in order to generate pedagogical actions. The student model records the history of usage for each constraint (both for domain constraints and the constraints

from the collaboration model), while the group model records the history of group usage for each domain constraint.

COLLECT-*UML* contains an ideal solution for each problem, which is compared to the student's solution according to the system's domain knowledge, represented as a set of constraints [15]. The system's domain model contains a set of 133 constraints defining the basic domain principles, a set of problems and their solutions [3]. In order to develop constraints, we studied material in textbooks, such as [7], and also used our own experience in teaching UML and OO analysis and design. Figure 2 illustrates a constraint from the UML domain, which checks whether the student has defined all the methods necessary for the current problem. The relevance condition identifies a method in the ideal solution (IS) and then checks whether the class it belongs to also exists in the student's solution (SS). The student's solution is correct if the satisfaction condition is met, when the matching method also exists in the student's solution. The constraint also contains a message which would be given to the student if the constraint is violated.

```
(54
  "Check whether you have defined all the methods as specified
   by the problem. You are missing some methods."
  (and (match IS METHODS (?* "@" ?tag ?name ?class_tag ?*))
       (match SS CLASSES (?* "@" ?class_tag ?*)))
  (match SS METHODS (?* "@" ?tag ?name2 ?class_tag ?*))
  "methods"
  (?class_tag))
```

Fig. 2. Example of a domain constraint

The student interface is shown in Figure 3. The problem text describes a situation that needs to be modelled by a UML class diagram. Students construct their individual solutions in the private workspace (right). They use the shared workspace (left) to collaboratively construct UML diagrams while communicating via the chat window (bottom). The private workspace enables students to try their own solutions and think about the problem before they start discussing it in the group.

The group diagram is initially disabled. It is activated after a specified amount of time, and the students can start placing components of their solutions in the shared workspace. This may be done by either copying/pasting from private diagram or by drawing new components in the group diagram. The private and shared workspaces can be resized. The students need to select the component names from the problem text by highlighting or double-clicking on the words/phrases. The *Group Members* panel shows the team-mates already connected. Only one student, the one who has the pen, can update the shared workspace at a given time. The control panel provides two buttons to control this workspace: *Get Pen and Leave Pen*, and shows the name of the student who has the control of this area. The chat area enables students to express their opinions by selecting one of the sentence openers, and typing their statement.

While all group members can contribute to the chat area and group solution, only one member of the group (i.e. the group moderator) can submit the group solution (by clicking on the *Submit Group Answer* button). The system provides feedback on the individual solutions, as well as on group solutions and

collaboration. All feedback messages will appear in the frame located on the right-hand side of the interface.

The domain-level feedback on both individual and group solutions is offered at four levels of detail: *Simple Feedback*, *Error flag*, *Hint* and *All Hints.* In addition, the group moderator has the option of asking for the complete solution, by clicking on *Show Full Solution* button. The collaboration-based advice is given to individual students based on the content of the chat area (i.e. sentence openers the students used), the student's contributions to the shared diagram and the differences between student's individual solution and the group solution being constructed. The system scales to a large number of participants and to large problem spaces. For more details on the interface and justification of using sentence openers, private workspace and turn taking, please refer to [2].

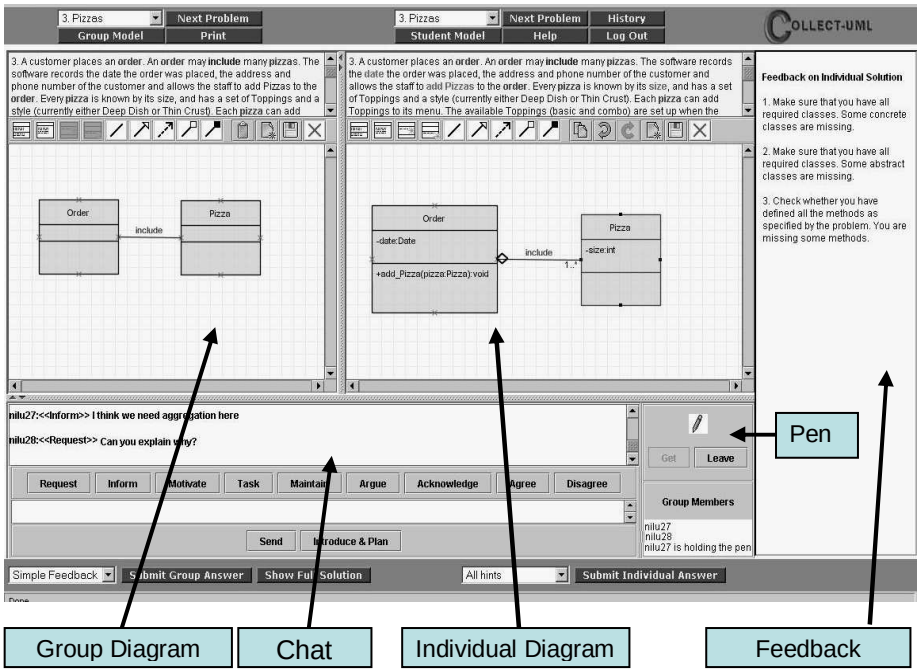

Fig. 3. COLLECT-$\mathcal{UML}$ Interface

4 Modelling Collaboration

Research on learning has demonstrated the usefulness of collaboration for improving student's problem-solving skills. However, simply putting students together and giving them a task does not mean that they will collaborate well. Collaboration is a skill, and, as any other skill, needs to be taught and practised to be acquired. Students learning via CSCL technology need practice, guidance and support in learning the social interaction skills, just as students learning in the classroom need support from their instructor [17].

The goal of our research is to support collaboration by modelling collaborative skills. COLLECT-$\mathcal{UML}$ is capable of diagnosing students' collaborative actions, such as contributions to the chat area and contributions to the group diagram, using an

explicit model of collaboration. This collaboration model is represented using constraints, the same formalism used to represent domain knowledge. A significant contribution of our work is to show that constraint can be used not only to represent domain-level knowledge, but also higher-order skills such as collaboration.

Our model of collaboration consists of set of 25 meta-constraints representing ideal collaboration. The structure of meta-constraints is identical to that of domain-level constraints: each meta-constraint consists of a relevance condition, a satisfaction condition and a feedback message. The feedback message is presented when the constraint is violated. In order to develop meta-constraints, we studied the existing literature on characteristics of effective collaboration [5, 16, 17, 19], and also used our own experience in collaborative work. The collaborative teaching strategy used is based on the socio-cognitive conflict theory [6]. According to this theory, social interaction is constructive only if it creates a confrontation between students' divergent solutions.

The meta-constraints are divided into two main groups: constraints that monitor students' contributions to the group diagram (making sure that students remain active, encouraging them to discuss the differences between their individual diagrams and the group diagram, etc.), and constraints that monitor students' contributions to the chat area and the use of sentence openers.

Figure 4 illustrates two meta-constraints. The relevance condition of constraint 223 focuses on aggregation relationships that exist in the student's individual solution between certain classes, when the same classes also exist in the group solution (GS). For this constraint to be satisfied, the corresponding relationships should also appear in the group solution. If that is not the case, the constraint is violated, and the student will be given the feedback message attached to this constraint, which encourages them to discuss those relationships with the group, or add them to the group solution. Constraint 238 is relevant if the student has made a contribution to the chat area, and its satisfaction condition checks whether the student has typed a statement after using any of the available sentence openers. If not, it encourages them to provide more explanation as part of their contribution.

In order to be able to evaluate meta-constraints, the system maintains a rich collection of data about all actions students perform in COLLECT-*UML*. After each change made to the group diagram, an XML event message containing the update and the id of the student who made that change, is sent to the server. Each chat message will also be sent to the server in the XML format.

Histories of all contributions made to the shared diagram as well as the messages posted to the chat area are stored on the server. The meta-constraints are evaluated against these histories, and feedback is given on contributions which involve adding/deleting/updating components in the shared diagram, as well as contributions made to the chat area.

5 Evaluation

An evaluation study was carried out at the University of Canterbury in May 2006. The study involved 48 volunteers enrolled in an introductory Software Engineering course. The students learnt UML modelling concepts during two weeks of lectures and had some practice during two weeks of tutorials prior to the study. The study was conducted in two streams of two-hour laboratory sessions over two weeks. In the first week, the students filled out a pre-test and interacted with the single-user

version of the system. Doing so gave them a chance to learn the interface and provided us with an opportunity to assess their UML knowledge and decide on the pairs and moderators.

```
(223
  "Some relationship types (aggregations) in your individual
   solution are missing from the group diagram. You may wish to
   share your work by adding those aggregation(s)/discuss it with
   other members."
  (and (match SS RELATIONSHIPS (?* "@" ?rel_tag "aggregation"
        ?c1_tag ?c2_tag ?*))
       (match GS CLASSES (?* "@" ?c1_tag ?*))
       (match GS CLASSES (?* "@" ?c2_tag ?*)))
  (or-p (match GS RELATIONSHIPS (?* "@" ?rel_tag "aggregation"
        ?c1_tag ?c2_tag ?*))
        (match GS RELATIONSHIPS (?* "@" ?rel_tag "aggregation"
        ?c2_tag ?c1_tag ?*)))
   "relationships"
  (?rel_tag ?c1_tag ?c2_tag))

 (238
  "Ensure adequate elaboration is provided in explanations."
  (match SC DESC (?* "@" ?tag ?text ?*))
  (not-p (test SC ("null" ?text)))
  "descriptions"
   nil)
```

Fig. 4. Examples of meta-constraints

At the beginning of the sessions in the second week, we told students what characteristics we would be looking for in effective collaboration (that was considered as a short training session). The instructions describing the characteristics of good collaboration and the process we expected them to follow were also handed out. The idea of providing students with such a script and therefore supporting instructional learning came from a recent study conducted by Rummel and Spada [16]. The participants were also given a screenshot of the system highlighting the important features of the multi-user interface (Figure 3).

The students were randomly divided into pairs with a pre-specified moderator. The moderator for each pair was the student who had scored higher in the pre-test. The pairs worked on a relatively complex problem individually and joined the group discussion whenever they were ready – the group diagram was activated after 10 minutes. At the end of the session, each participant completed a post-test and a questionnaire commenting on the interface, the impact of the system on their domain knowledge and their collaborative skills, and the quality of the feedback messages on their individual and collaborative activities.

The experimental group consisted of 26 students (13 pairs) who received feedback on their solution as well as their collaborative activities. The control group consisted of 22 students (11 pairs) who only received feedback on their solutions (no feedback on collaboration was provided in this case). There were four female participants in four different pairs (one from the control group and three from the experimental group). All pairs received instructions on characteristics of good collaboration at the beginning of the second week.

The total time spent interacting with the system was 1.4 hours for the control and 1.3 hours for the experimental group. The pre-test and post-test each contained four multiple-choice questions, followed by a question where the students were asked to design a simple UML class diagram. The tests included questions of comparable difficulty, dealing with inheritance and association relationships. The post-test also had an extra question, asking the participants to describe the aspects of effective collaborative problem-solving. The mean scores of the pre- and post-test are given in Table 1. The numbers reported for the post-test do not include the collaboration question.

Table 1. Pre- and post-test scores

	Control		**Experimental**	
	Average	**s. d.**	**Average**	**s. d.**
Collaboration	22%	22%	52%	39%
Pre-test	52%	20%	49%	19%
Post-test	76%	25%	73%	25%
Gain score	17%	28%	21%	31%

There was no significant difference on the pre-test results, meaning that the groups were comparable. The students' performance on the post-test was significantly better for both control group ($t = 2.11$, $p = 0.01$) and experimental group ($t = 2.06$, $p = 0.002$). The experimental group, who received feedback on their collaboration performed significantly better on the collaboration question ($t = 2.02$, $p = 0.003$), showing that they acquired more knowledge on effective collaboration. We also calculated the effect size for the question about collaboration. The common method to calculate it is to subtract the control group's mean score from the experimental group's mean score and divide by the standard deviation of the control group. Using this method, the effect size on student's collaboration knowledge is very high: $(Average\ collaboration_{exp} - Average\ collaboration_{control}) / s.d._{control} = 1.3$.

The experimental group students contributed more to the group diagram, with the difference between the average number of individual contribution for control and experimental group being statistically significant ($t = 2.03$, $p = 0.03$). The meta-constraints generated collaboration-based feedback 19.4 times on average for the experimental group (for each student).

We have also analyzed the students' individual log files, in order to identify how students learnt the underlying domain concepts in the second week. Figure 5 illustrates the probability of violating a domain constraint plotted against the

Fig. 5. Probability of domain constraint violation for individuals in control and experimental group

occasion number for which it was relevant, averaged over all domain constraints and all participants in control and experimental groups. The data points show a regular decrease, which is approximated by a power curve with a close fit of 0.78 and 0.85 for control and experimental groups respectively, thus showing that students do learn constraints over time. The probability of 0.21/0.23 for violating a constraint on the first occasion of application has decreased to 0.09/0.12 at its eleventh occasion, displaying a 61.9%/47.8% decrease in probability for the control/experimental group respectively. Figure 6 illustrates the learning curve for meta-constraints only (for the experimental group). There is also a regular decrease, thus showing that students learn meta-constraints over time. Because the students used the system for a short time only, more data is needed to analyze learning of meta-constraints, but the trend identified in this study is encouraging.

The participants were given a questionnaire at the end of the session to determine their perceptions of the system. Most of the participants (61% of control and 78% of experimental group) responded they would recommend the system to other students. The students found the interface easy to learn and use and enjoyed working with a partner. The comments we received on open questions show that the students liked the system and thought it improved their knowledge, and also pointed out several possible improvements.

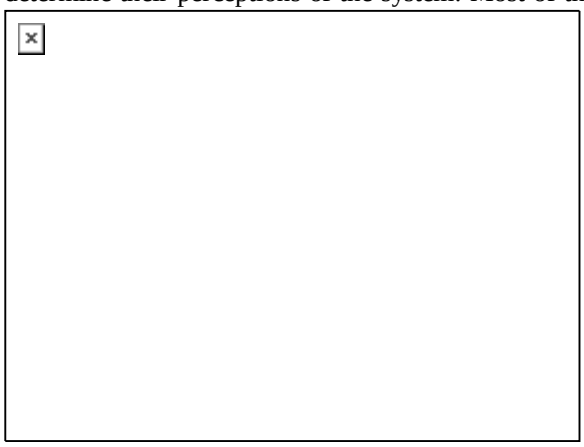

Fig. 6. Probability of meta-constraint violation

6 Conclusions

CBM has previously been used to effectively represent domain knowledge in several ITSs supporting individual learning. The contribution of this research is the use of CBM to model collaboration skills, not only domain knowledge. We described the process of extending COLLECT-*UML*, an ITS for UML class diagrams, to support collaboration. COLLECT-*UML* provides task-based feedback on students' and group solutions, as well as collaboration-based feedback intended to make the collaboration process more effective. The collaborative feedback is provided by analyzing students' activities and comparing them to an ideal model of collaboration.

The system's effectiveness in teaching good collaboration and UML class diagrams was evaluated in a classroom experiment. The results of both subjective and objective analysis proved that COLLECT-*UML* is an effective educational tool. The experimental group students acquired more declarative knowledge on effective collaboration, as they scored significantly higher on the collaboration test. The collaboration skills of the experimental group students were better, as evidenced by these students being more active in collaboration, and contributing more to the group diagram. All students improved their problem-solving skills: the participants

from both control and experimental group performed significantly better on the post-test after short sessions with the system, showing that they acquired more knowledge on UML modelling. Finally, the students enjoyed working with the system and found it a valuable asset to their learning.

The results, therefore, show that CBM is an effective technique for modelling and supporting collaboration in CSCL environments.

References

1. Baghaei, N., Mitrovic, A., Irwin, W.: A Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML. Proc. ICCE 2005, (2005) 11-18
2. Baghaei, N., Mitrovic, A.: A Constraint-based Collaborative Environment for Learning UML Class Diagrams. Proc. ITS 2006, (2006) 176-186
3. Baghaei, N., Mitrovic, A., Irwin, W.: Problem-Solving Support in a Constraint-based Tutor for UML Class Diagrams, Technology, Instruction, Cognition and Learning Journal, 4(2) (2006) (in print)
4. Brusilovsky, P., Peylo, C.: Adaptive and Intelligent Web-based Educational Systems. Artificial Intelligence in Education, 13, (2003) 159-172
5. Constantino-Gonzalez, M. A., Suthers, D., Escamilla de los Santos, J.: Coaching web-based collaborative learning based on problem solution differences and participation. Int. J. Artificial Intelligence in Education, 13(2-4), (2003) 263-299
6. Doise, W., Mugny, G.: The social development of the intellect. Int. Series in Experimental Social Psychology, 10, Pergamon Press. (1984)
7. Fowler, M.: UML Distilled: a Brief Guide to the Standard Object Modelling Language. Reading: Addison-Wesley, 3rd edition. (2004)
8. Jarboe, S.: Procedures for enhancing group decision making. In: B. Hirokawa, M. Poole (eds.): Communication and Group Decision Making. (1996) 345-383
9. Jerman, P., Soller, A., Muhlenbrock, M.: From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. In: P. Dillenbourg, A. Eurelings, K. Hakkarainen (eds.) CSCL 2001, (2001) 324-331
10. Martin, B., Mitrovic, A.: Domain Modelling: Art or Science? In: Hoppe, U., Verdejo, F., Kay J. (eds.), Proc. 11th Int. Conference on AIED, (2003) 183-190
11. Mayo, M., Mitrovic, A.: Optimising ITS behaviour with Bayesian networks and decision theory. Artificial Intelligence in Education, 12(2), (2001) 124-153
12. McManus, M., Aiken, R.: Monitoring computer-based problem solving. Int. Journal of Artificial Intelligence in Education, 6(4), (1995) 307-336
13. Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A.: DB-suite: Experiences with three Intelligent, Web-based Database Tutors. Journal of Interactive Learning Research, 15(4), (2004) 409-432
14. Ogata, H., Matsuura, K., Yano, Y.: Active Knowledge Awareness Map: Visualizing learners activities in a Web based CSCL environment. Int. Workshop on New Technologies in Collaborative Learning, (2000) 89-97
15. Ohlsson, S: Constraint-based Student Modelling. In: J. Greer, G. McCalla (eds.): Student Modelling: the Key to Individualized Knowledge-based Instruction, Berlin: Springer-Verlag (1994) 167-189
16. Rummel, N., Spada, H.: Learning to collaborate: An instructional approach to promoting collaborative problem-solving in computer-mediated settings. Journal of the Learning Sciences, 14(2), (2005) 201-241
17. Soller, A.: Supporting Social Interaction in an Intelligent Collaborative Learning System. International Journal of AIED, 12, (2001) 40-62
18. Soller, A., Lesgold, A.: Knowledge acquisition for adaptive collaborative learning environments. AAAI Fall Symposium: Learning How to Do Things. (2000)
19. Vizcaino, A.: A Simulated Student Can Improve Collaborative Learning. Int. Journal of Artificial Intelligence in Education, 15, (2005) 3-40