A Constraint-based Collaborative Environment for Learning UML Class Diagrams

Nilufar BAGHAEI and Antonija MITROVIC

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

Abstract. COLLECT-*UML* is a constraint-based ITS that teaches object-oriented design using Unified Modelling Language (UML). UML is easily the most popular object-oriented modelling technology in current practice. We started by developing a single-user ITS that supported students in learning UML class diagrams. The system was evaluated in a real classroom, and the results show that students' performance increased significantly. In this paper, we present our experiences in extending the system to provide support for collaboration. We present the architecture, interface and support for collaboration in the new, multi-user system. A full evaluation study has been planned, the goal of which is to evaluate the effect of using the system on students' learning and collaboration.

1. Introduction

E-learning is becoming an increasingly popular educational paradigm as more individuals who are working or are geographically isolated seek higher education. As such students do not meet face to face with their peers and teachers, the support for collaboration becomes extremely important [8]. Effective collaborative learning includes both learning to effectively collaborate, and collaborate effectively to learn, and therefore a collaborative system must be able to address collaboration issues as well as task-oriented issues [17].

In the last decade, many researchers have contributed to the development of CSCL and advantages of collaborative learning over individualised learning have been identified [14]. 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 articulate their reasoning, and elaborate and reflect upon their knowledge [24, 27]. These benefits, however, are only achieved by active and well-functioning learning teams [15]. Numerous systems for collaborative learning have been developed; however, the concept of supporting peer-to-peer interaction in CSCL systems is still in its infancy. Various strategies for computationally supporting online collaborative learning have

been proposed and used, while more studies are needed that test the utility of these techniques [17].

This paper describes an Intelligent Tutoring System (ITS) that uses Constraint-Based Modeling (CBM) approach to support both problem-solving and collaborative learning. CBM has been used successfully in several tutors supporting individual learning [20]. We have developed COLLECT-$\mathcal{UML}$ [2, 3], a single-user version of a constraint-based ITS, that teaches UML class diagrams. In this paper, we describe extensions to this tutor, which support multiple students solving problems collaboratively. We start with a brief overview of related work in Section 2. Section 3 then presents COLLECT-$\mathcal{UML}$ and the evaluation study conducted with second-year university students taking a course in Introduction to Software Engineering. Section 4 describes the design and implementation of the collaborative interface as well as the system's architecture. Section 5 presents the collaborative model, which has been implemented as a set of meta-constraints. Conclusions are given in the last section.

2. Related Work

Three categories of CSCL systems can be distinguished in the context of the collaboration support [1, 17]. The first category includes systems that reflect actions; the basic level of support a system may offer involves making the students aware of the participants' 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. Sharlock II [21]), 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 [25]). 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 [17]. Finally, the third class of systems offer advice on collaboration. The coach in these systems plays a role similar to that of a teacher in a collaborative learning classroom. 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. Examples of the systems focusing on the social aspects include Group Leader Tutor [19] and DEGREE [6], and an example of the systems addressing both social and task-oriented aspects of group learning is COLER [7].

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 [25] 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 OO design problems. The system first logs data describing the students' speech acts (e.g. *Request Opinion, Suggest*, and *Apologise*) and actions (e.g. *Student 3 created a new class*). It then collects examples of effective and ineffective knowledge sharing, and constructs two Hidden Markov Models which describe the students' interaction in these two cases. A knowledge sharing example is considered effective if one or more students learn the newly shared knowledge (as shown by a difference in pre-post test performance), and ineffective otherwise. The system dynamically assesses

a group's interaction in the context of the constructed models, 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 it could be common for a group of students to agree on the same flawed argument.

CBM has been used successfully in several tutors supporting individual learning. The main contribution of this research is the use of CBM technique to support collaborative learning. 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). CBM is also used to model student and group knowledge.

3. COLLECT-*UML*: Single-User Version

COLLECT-*UML* is a problem-solving environment, in which students construct UML class diagrams that satisfy a given set of requirements. It assists students during problem-solving, and guides them towards a correct solution by providing feedback. The feedback is tailored towards each student depending on his/her knowledge. COLLECT-*UML* 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. For details on system's architecture, functionality and the interface refer to [2, 3]; here we present only the basic features of the system.

At the beginning of interaction, a student is required to enter his/her name, which is necessary in order to establish a session. The session manager requires the student modeller to retrieve the model for the student, if there is one, or to create a new model for a new student. Each action a student performs is sent to the session manager, as it has to link it to the appropriate session and store it in the student's log. Then, the action is sent to the pedagogical module. If the submitted action is a solution to the current problem, the student modeller diagnoses the solution, updates the student model, and sends the result of the diagnosis back to the pedagogical module, which generates appropriate feedback.

COLLECT-*UML* contains an ideal solution for each problem, which is compared to the student's solution according to the system's domain model, represented as a set of constraints [22]. The system's domain model contains 133 constraints that describe the basic principles of the domain. In order to develop constraints, we studied material in textbooks, such as [12], and also used our own experience in teaching UML and OO analysis and design.

Figure 1 illustrates a constraint from the UML domain. The relevance condition identifies a relationship of type aggregation in the ideal solution, and then checks whether the student's solution contains the same type of relationship, or a relationship of a different kind with the same name. The student's solution is correct if the satisfaction condition is met, when the matching relationship is of the same type (i.e. aggregation). The constraint also contains a message which would be given to the student if the constraint is violated. The last two elements of the constraint specify that it covers some aspects of relationships, and also identifies the relationship and the classes to which the constraint was applied.

```
(78
  "Check the type of your relationships. You need to use aggregations between some of
your classes."
    (and (match IS RELATIONSHIPS (?* "@" ?rel_tag "aggregation" ?c1_tag ?c2_tag ?*))
      (or-p (match SS RELATIONSHIPS (?* "@" ?rel_tag ?type ?c1_tag ?c2_tag ?*))
        (match SS RELATIONSHIPS (?* "@" ?rel_tag ?type ?c2_tag ?c1_tag ?*))))
  (test SS ("aggregation" ?type))
  "relationships"
  (?rel_tag ?c1_tag ?c2_tag))
```

Figure 1. An example constraint

We performed an evaluation study [3] in May 2005 with 38 students enrolled in a Software Engineering course. The students learnt UML modelling concepts during two weeks of lectures/tutorials. The study was conducted in two streams of two-hour laboratory sessions. Each participant sat a pre-test, interacted with the system, and then sat a post-test and filled a user questionnaire. 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. Table 1 presents some general statistics about the study. The average mark on the post-test was significantly higher than the pre-test mark ($t = 2.71$, $p = 4.33E-08$). The students spent on average 90 minutes interacting with the system.

Table 1. Some statistics about the study

	Average	s. d.
Attempted problems	5.71	2.59
Solved problems	47%	33%
Attempts per problem	7.42	4.76
Pre-test	52%	21%
Post-test	76%	17%

We also analyzed the log files, in order to identify how students learn the underlying domain concepts. Figure 2 illustrates the probability of violating a constraint plotted against the occasion number for which it was relevant, averaged over all constraints and all participants. The data points show a regular decrease, which is approximated by a power curve with a close fit of 0.93, thus showing that students do learn constraints over time. The probability of violating a constraint on the first occasion of application is halved by the tenth occasion, showing the effects of learning.

Students were offered individualised feedback on their solutions upon submission. The mean rating for the usefulness of feedback was 2.8. 67% of the participants had indicated that they would have liked to see more details in the feedback messages. The comments we received on open questions pointed out several features of the system, which can be improved.

The results showed that COLLECT-𝒰𝑀𝐿 is an effective learning environment. The participants achieved significantly higher scores on the post-test, suggesting that they acquired more knowledge in UML modelling. The learning curves also prove that students do learn constraints during problem solving. Subjective evaluation shows that most of the students felt spending more time with the system would have resulted in more learning and that they found the system to be easy to use.
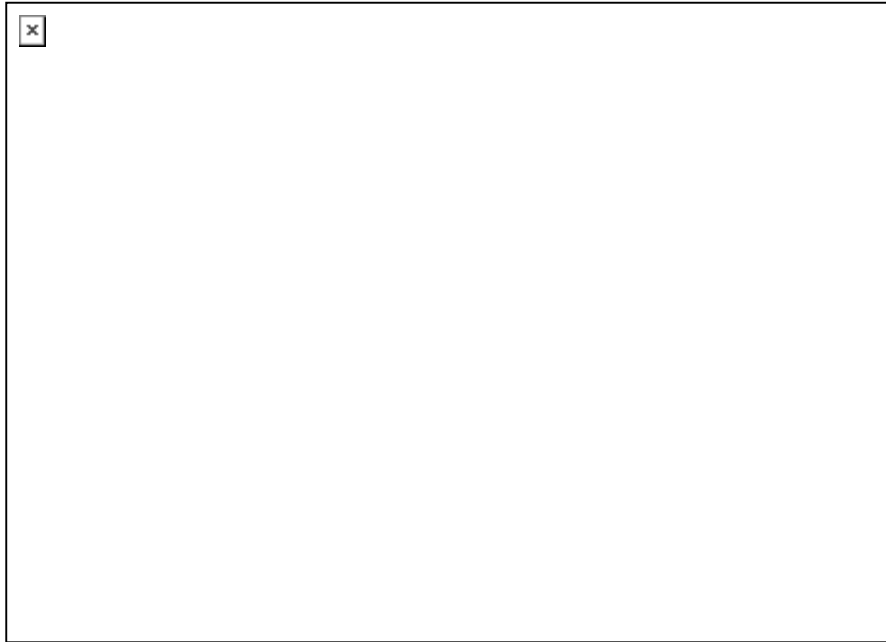
Figure 2. Probability of constraint violation

4. COLLECT-*UML*: Multi-User Version

The collaborative version of COLLECT-*UML* is designed for sessions in which students first solve problems individually and then join into small groups to create group solutions. The system's architecture is illustrated in Figure 3. The application server consists of a session manager that manages sessions and student logs, a student modeller that creates and maintains student models for individual users, a domain model (i.e. the constraint set), a pedagogical module and a group modeller. The system is implemented in Allegro Common Lisp.

The interface is shown in Figure 4. The problem description pane presents a design problem. Students construct their individual solutions in the private workspace (right), and use the shared workspace (left) to collaborate while communicating via the chat window (bottom). The private workspace enables students to try their own solutions and think about the problem before start discussing it in the group. The group area is initially disabled. When all of the students indicate readiness to work in the group by clicking on *Join the Group* button, the shared workspace is activated. The students select the components' names from the problem text. 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. Additionally, this panel shows the name of the student who has the control of this area and the students waiting for a turn.

A recent study [23] defines relevant characteristics of good collaboration and the authors have considered turn-taking as one of those characteristics. According to their results, explicitly handing over a turn can be a good way of compensating for the limited communication channel. An implication of providing such protocol is that deadlocks can be created in cases where one partner cannot proceed with

5

problem-solving alone and at the same time refuses to pass the key over to the other partners. The advantage, however, is that it maintains clear semantics of a participant's actions and roles in the shared workspace [10]. The lack of providing turn-taking protocol in most of computer-mediated collaboration tools is considered to be one of the limitations of such tools [11].
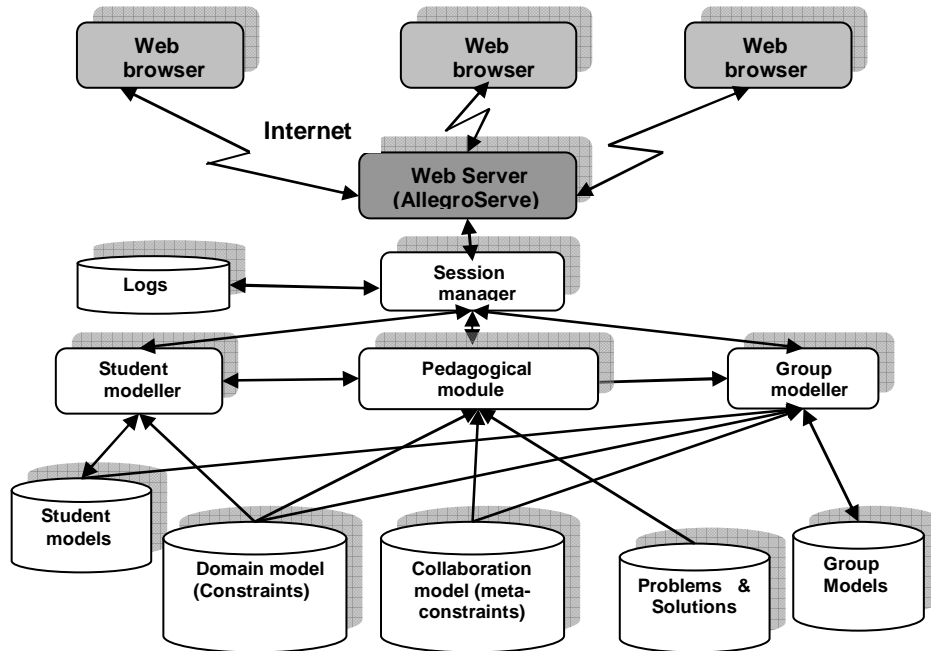

Figure 3. The architecture of COLLECT-$\mathcal{UML}$

The chat area enables students to express their opinions using sentence openers. The student needs to select one of the sentence openers before being able to express his/her opinion. The contents of selected sentence openers are displayed in the chat area along with any optional justifications. Sentence openers structure students' conversation and eliminate off-task discussions. A structured chat interface with specific sentence openers can promote more focus on reflection and the fundamental concepts at stake [5]. Although this kind of dialogue requires more effort from the student than using plain chat or email, as the student needs to categorize their own contributions, research shows that the quality of the dialogue can be higher [16]. In addition, structuring the dialogue makes it easier to analyze computationally [10].

Sentence openers provide a natural way for users to identify the intention of their conversational contribution without fully understanding the significance of the underlying communicative acts [19]. Results from various projects indicate that structured dialogues support students to stay on task and increase reflection [13]. However, requiring learners to select a sentence opener before typing the remainder of their contribution may tempt them to change the meaning of the contribution to fit one of the sentence openers, thus changing the nature of the collaborative interaction. According to Lazonder et al. [18], sentence openers should be derived from naturally occurring online text-based free dialogues, while Soller [24] states that it is critical to provide the widest and most appropriate range of sentence

openers. Some experiments [4] show that in interfaces containing both structured and free chat tools, the former are used more frequently.

The group moderator can submit the solution, by clicking on the *Submit Answer* button on the shared workspace. The system gives collaboration-based advice based on the content of the chat area, students' participation on the shared diagram and the differences between students' individual solutions and the group solution being constructed. The task-based advice is given to the whole group based on the quality of the shared diagram.
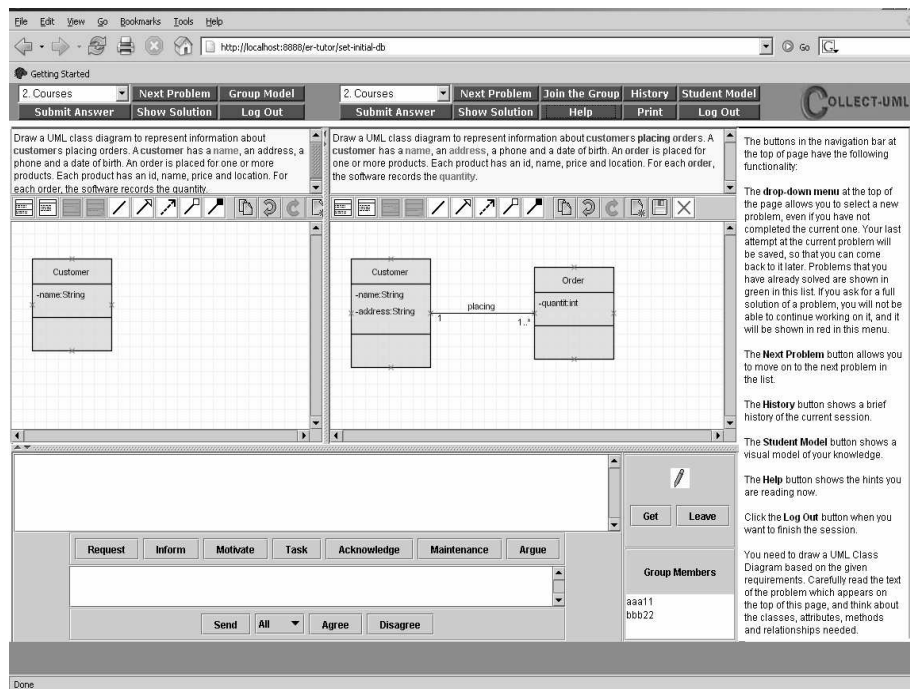

Figure 4. COLLECT-$\mathcal{UML}$ interface

The *Next Problem*, *Submit Answer*, *Show Solution* and *Log Out* buttons at the top of the shared diagram are controlled by the group moderator only, while the *Group Model* button can be accessed by all the members. The students can use the *Help* button (at the top of the individual workspace) to get information about UML Modeling, *Submit Answer* to get feedback on their individual solutions and *Next problem* to move on to a new problem (regardless of the problem the group is working on at that point). The students cannot view full solutions in the individual workspaces (that option is only available under the shared workspace). Viewing the full solution by individual members of the group might stop them from thinking about the problem and/or collaborating with the rest of the group members.

5. Modeling Collaboration

The ultimate goal of COLLECT-$\mathcal{UML}$ is to support collaboration by modelling collaborative skills. The system is able to promote effective interaction by

7

diagnosing students' actions in the chat area and group diagram using a set of 22 meta-constraints, which represent an ideal model of collaboration. These constraints have the same structure as domain constraint, each containing 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 existing literature on characteristics of an effective collaboration, such as [9, 23, 24, 26]. Figure 5 illustrates two examples of meta-constraints. Constraint 221 encourages student participation in problem-solving. This constraint makes sure that the student contributes associations from his/her individual solution to the group solution. On the other hand, there are constraints that check whether the student participates in the dialogue. Constraint 237 checks whether the student has specified any justification for their agreement/disagreement with the group solution.

A history of all contributions made by each user to the shared diagram as well as the messages posted to the chat area is maintained on the server, and the meta-constraints are evaluated against this history. Feedback is given on contributions which involve adding/deleting/updating components in the shared diagram, as well as contributions made to the chat area.

```
(221
  "Some relationship types (associations) in your individual solution are missing from the
group diagram. You may wish to share your work by adding those association(s)/discuss it
with other members."
  (and (match SS RELATIONSHIPS (?* "@" ?rel_tag "association" ?c1_tag ?c2_tag ?*))
      (match GS CLASSES (?* "@" ?c1_tag ?*))
      (match GS CLASSES (?* "@" ?c2_tag ?*)))
  (or-p (match GS RELATIONSHIPS (?* "@" ?rel_tag "association" ?c1_tag ?c2_tag ?*))
      (match GS RELATIONSHIPS (?* "@" ?rel_tag "association" ?c2_tag ?c1_tag ?*)))
  "relationships"
  (?rel_tag ?c1_tag ?c2_tag))

(237
  "You may wish to explain to other members why you agree or disagree  with a solution."
  (and (match SC DESC (?* "@" ?tag ?text ?*))
      (or-p (test SC ("agree" ?tag))
          (test SC ("disagree" ?tag))))
  (not-p (test SC ("" ?text)))
  "descriptions"
  nil)
```

Figure 5. Examples of meta-constraints

6. Conclusions and Future Work

This paper presented the single-user version of COLLECT-$\mathcal{UML}$, and the results of the evaluation study performed. The results of both subjective and objective analysis proved that COLLECT-$\mathcal{UML}$ is an effective educational tool. The participants performed significantly better on a post-test after short sessions with the system, and reported that the system was relatively easy to use.

We then presented the multi-user version of the same intelligent tutoring system. We have extended COLLECT-*UML*' interface, and developed meta-constraints, which provide feedback on collaborative activities. The goal of future work is to complete the implementation of the multi-user version and conduct a full evaluation study with second-year University students enrolled in an undergraduate software engineering course. The study is planned for April 2006. Participants will be divided into three groups. The experimental condition will receive feedback on the domain model as well as their collaborative activities. The students will also be provided with a script addressing the characteristics of a good collaboration and the phases they are expected to go through, at the beginning of the session. The second group will receive feedback on their solutions only. These students will be provided with the same script at the beginning of the session, but will not receive feedback on collaboration. The control group will only receive feedback on the domain level. There will not be any type of support on the collaboration process available to this group. Our hypothesis is that all groups will increase their problem-solving skills, but that only the experimental group will improve collaboration skills. All participants will be assessed on their understanding of what characterises good collaboration at the end of the session by answering questions in the post-test. Their interaction in the shared diagram and chat area will also be analysed.

CBM has been used to effectively represent domain knowledge in several ITSs supporting individual learning. The contribution of the project presented in this paper is the use of CBM to model collaboration skills, not only domain knowledge. Comprehensive evaluation of the multi-user version of COLLECT-*UML* will provide a measure of the effectiveness of using the CBM technique in intelligent computer-supported collaborative learning environments.

References

[1]   Baghaei, N. and Mitrovic, A. (2005) *COLLECT-UML: Supporting individual and collaborative learning of UML class diagrams in a constraint-based tutor.* In Khosla, R., Howlett, R. and Jain L. (eds.) KES 2005, pp.458-464.

[2]   Baghaei, N., Mitrovic, A. and Irwin, W. (2005) *A Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML.* In Looi, C., Jonassen, D. and Ikeda M. (eds.) ICCE 2005, pp.11-18.

[3]   Baghaei, N., Mitrovic, A. and Irwin, W. (2006) *Problem-Solving Support in a Constraint-based Tutor for UML Class Diagrams*, Technology, Instruction, Cognition and Learning Journal, 4(1-2) (in print).

[4]   Baker, M. J. and Lund, K. (1997) *Promoting reflective interactions in a computer-supported collaborative learning environment.* Journal of Computer Assisted Learning, 13(3), 175-193.

[5]   Baker, M., de Vries, E., Lund, K. and Quignard, M. (2001) *Computer-mediated epistemic interactions for co-constructing scientific notions: Lessons learned form a five-year research program.* In Dillenbourg, P., Eurelings, A. and Hakkarainnen, K. (eds.) European Perspectives on CSCL (CSCL 2001), pp.89-96.

[6]   Barros, B. and Verdejo, M.F. (2000) *Analysing student interaction processes in order to improve collaboration: The DEGREE approach.* Artificial Intelligence in Education, 11, 221-241.

[7]   Constantino-Gonzalez, M., and Suthers, D. (2000) *A coached collaborative learning environment for Entity-Relationship modelling.* In Gauthier, G., Frasson, C. and VanLehn, K. (eds.) 5th Int. Conf. Intelligent Tutoring Systems, pp.324-333.

[8]   Constantino-Gonzalez, M., and Suthers, D. (2002) *Coaching Collaboration in a*

Computer Mediated Learning Environment. In Stahl, G. (eds.) CSCL 2002, pp.583-584.

[9]   Constantino-Gonzalez, M. A., Suthers, D. and Escamilla de los Santos, J. (2003) *Coaching web-based collaborative learning based on problem solution differences and participation.* Artificial Intelligence in Education, 13 (2-4), 261-297.

[10]   Dimitracopoulou, A. (2005) *Designing Collaborative Learning Systems: Current Trends & Future Research Agenda.* In Koschmann, T., Suthers, D. and Chan T.W. (eds.) 6th Int. Computer Supported Collaborative Learning Conf. CSCL 2005.

[11]   Feidas, C., Komis, V. and Avouris, N. (2001) *Design of collaboration-support tools for group problem solving.* In Avouris, N. and Fakotakis, N. (eds.) Advances in Human-Computer Interaction, pp.263-268.

[12]   Fowler, M. (2004) *UML Distilled: a Brief Guide to the Standard Object Modelling Language.* Reading: Addison-Wesley, 3rd edition.

[13]   Gogoulou, A., Gouli, E., Grigoriadou, M. and Samarakou, M. (2005) *ACT: A Web-based Adaptive Communication Tool.* In Koschmann, T., Suthers, D. and Chan T.W. (eds.) 6th Int. Computer Supported Collaborative Learning Conf., pp.180-189.

[14]   Inaba, A. and Mizoguchi, R. (2004) *Learners' Roles and Predictable Educational Benefits in Collaborative Learning; An Ontological Approach to Support Design and Analysis of CSCL.* In Lester, J., Vicari, R. M. and Paraguacu, F. (eds.) 7th Int. Conf. Intelligent Tutoring Systems, pp.285–294.

[15]   Jarboe, S. (1996) *Procedures for enhancing group decision making.* In Hirokawa B. and Poole M. (eds.) Communication and Group Decision Making, pp.345-383.

[16]   Jermann, P., Soller, A. and Lesgold, A. (2004) *Computer software support for CSCL.* In Dillenbourg P., Strijbos J.W., Kirschner, P.A. and Martens R.L. (eds.) Computer-supported collaborative learning: Vol 3. What we know about CSCL ... and implementing it in higher education, pp.141-166.

[17]   Jerman, P., Soller, A. and Muhlenbrock, M. (2001) *From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning.* In Dillenbourg, P., Eurelings, A and Hakkarainen, K. (eds.) European Perspectives on CSCL (CSCL 2001), pp.324-331.

[18]   Lazonder, A., Wilhelm, P. and Ootes S. (2003) *Using sentence openers to foster student interaction in computer-mediated learning environments.* Computers & Education, 41, 291-308.

[19]   McManus, M. and Aiken, R. (1995) *Monitoring computer-based problem solving.* Artificial Intelligence in Education, 6(4), 307-336.

[20]   Mitrovic, A., Mayo, M., Suraweera, P. and Martin, B. (2001) *Constraint-based Tutors: a Success Story.* 14th Int. Conf. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp.931-940.

[21]   Ogata, H., Matsuura, K., and Yano, Y. (2000) *Active Knowledge Awareness Map: Visualizing learners activities in a Web based CSCL environment.* Int. Workshop on New Technologies in Collaborative Learning, pp.89-97.

[22]   Ohlsson, S. (1994) *Constraint-based Student Modelling.* Student Modelling: the Key to Individualized Knowledge-based Instruction, Springer-Verlag, pp.167-189.

[23]   Rummel, N. and Spada, H. (2005) *Learning to collaborate: An instructional approach to promoting collaborative problem-solving in computer-mediated settings.* Journal of the Learning Sciences, 14(2), 201-241.

[24]   Soller, A. (2001) *Supporting Social Interaction in an Intelligent Collaborative Learning System.* Artificial Intelligence in Education, 12, 40-62.

[25]   Soller, A. and Lesgold, *A.* (2000) *Knowledge acquisition for adaptive collaborative learning environments.* AAAI Fall Symposium: Learning How to Do Things, Cape Cod, MA.

[26]   Vizcaino, A. (2005) *A Simulated Student Can Improve Collaborative Learning.* Artificial Intelligence in Education, 15, 3-40.

[27]   Webb, N. M., Troper, J. D. and Fall, R. (1995) *Constructive activity and learning in collaborative small groups.* Journal of Educational Psychology, 87, 406-423.