A Collaborative Constraint-based Intelligent System for Learning Object-Oriented Analysis and Design using UML

Nilufar Baghaei

Department of Computer Science and Software Engineering
University of Canterbury
n.baghaei@cosc.canterbury.ac.nz

ABSTRACT

Automatic analysis of interaction and support for group learning through a distance collaborative learning system is at the forefront of educational technology. Research shows that collaborative learning provides an environment to enrich the learning process by introducing interactive partners into an educational system and creating more realistic social contexts.

This paper presents COLLECT-*UML*, 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. Constraint-Based Modelling (CBM) has been used successfully in several tutoring systems, which have proven to be extremely effective in evaluations performed in real classrooms. We have developed a single-user version that supports students in learning UML class diagrams. The system was evaluated in a real classroom, and the results showed that students' performance increased significantly while interacting with the system. We are now extending the system to provide support for collaboration. An overview of both single-user and collaborative versions of the system is presented. A full evaluation study has been planned for April 2006, 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 [3]. There have been several definitions for collaborative learning. The broadest (but unsatisfactory) definition is that it is a *situation* in which *two* or *more* people *learn* or attempt to learn something *together* [4]. A more comprehensive definition states as follows: "... a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem".

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 [6].

In the last decade, many collaborative learning environments have been proposed and used with more or less success. Researchers have been exploring different approaches to analyse and support the collaborative learning interaction. However, the concept of supporting peer-to-peer interaction in Computer-Supported Collaborative Learning (CSCL) systems is still in its infancy, and more studies are needed to test the utility of these techniques. 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 and increasing the possibility of students solving or examining problems in a variety of ways. These benefits, however, are only achieved by active and well-functioning learning teams [11].

This paper describes COLLECT-*UML*, an Intelligent Tutoring System (ITS) that uses Constraint-Based Modeling (CBM) approach to support both problem-solving and collaborative learning. The CBM approach is extremely efficient, and it overcomes many problems that other student modeling approaches suffer from. CBM has been used successfully in several tutors supporting individual learning [7]. We provide a brief overview of the single-user version which we have finished developing [1, 2] and describe extensions being made to this tutor, to support multiple students solving problems collaboratively.

2. RELATED WORK

This section provides examples of three types of CSCL systems, in the context of the collaboration management model [6, 9]:

- **Reflecting Actions:** The most basic level of support a system may offer involves making the students aware of the participants' actions. Actions taken on shared resources, or those that take place in private areas of a workspace may not be directly visible to the collaborators, yet they may significantly influence the collaboration. Raising awareness about such actions may help students maintain a representation of their teammates' activities. The system described in [8] is an example.
- **Monitoring the State of Interactions:** Systems that monitor the state of interaction fall into two categories: those that aggregate the interaction data into a set of high-level indicators, and display them to the participants, and those that internally compare the current state of interaction to a model of ideal interaction, but do not reveal this information to the users. In the former case, the learners are expected to manage the interaction themselves, having been given the appropriate information to do so. In the latter case, this information is either intended to be used later by a coaching agent, or analysed by researchers in an effort to understand and explain the interaction. EPSILON [11] and MArCo [12] are examples of such systems.
- **Offering Advice:** This will include the CSCL systems that analyse the state of collaboration using a model of interaction, and offer advice intended to increase the effectiveness of the learning process. The coach in an advising system 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 advice on strictly collaboration issues or both social and task-oriented issues. Examples include LeCS [10] and COLER [3].

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 [11] 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 (using machine learning techniques) situations in which students effectively share new knowledge with their peers while solving OO design problems. The system dynamically assesses the group interactions and determines when and why the students are having trouble learning the 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.

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 object-oriented design. For details on system's architecture, functionality and the interface refer to [1, 2]; 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. 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 [5], and also used our own experience in teaching UML and OO analysis and design.

We conducted an evaluation study in May 2005 [2]. The study involved 38 volunteers enrolled in an introductory Software Engineering course at the University of Canterbury, which teaches UML modelling as outlined by Fowler [5]. 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. 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. The participants spent two hours interacting with the system, and solved half of the problems they attempted. 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.

We also analyzed the log files, in order to identify how students learn the underlying domain concepts. Figure 1 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.

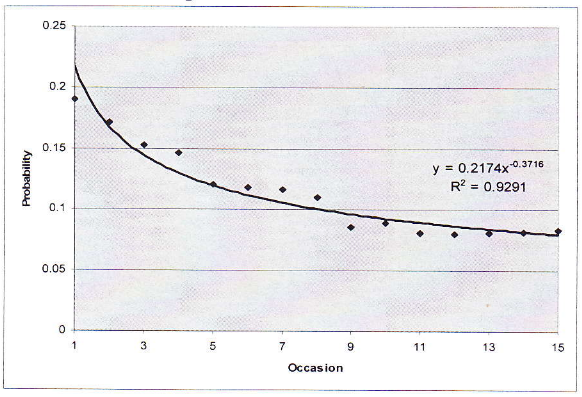

Figure 1. Probability of constraint violation

The results showed that COLLECT-*UML* is an effective learning environment [2]. 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.

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 has a distributed architecture, where the tutoring functionality is distributed between the client and the server.

The interface, which is an extension of the single-user interface, is shown in Figure 2. The problem description pane presents a design problem 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 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, and they can start placing components of their solutions in the workspace. 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 chat area enables students to express their opinion regarding objects added to the shared area using sentence openers. The student needs to select one of the sentence openers before being able to express his/her opinion. When the student clicks on *Agree* or *Disagree* buttons for example, the sentence "I agree …" or "I disagree …" appears in the chat window and the student may complete the sentence. The contents of selected sentence openers are displayed in the chat area along with any optional justifications.

The group moderator can submit the solution, by clicking on the *Submit Answer* button on the shared workspace. The feedback messages on the individual solutions as well as contribution to the group solution and collaboration will appear on the frame located in the right-hand side (Figure 2). 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.

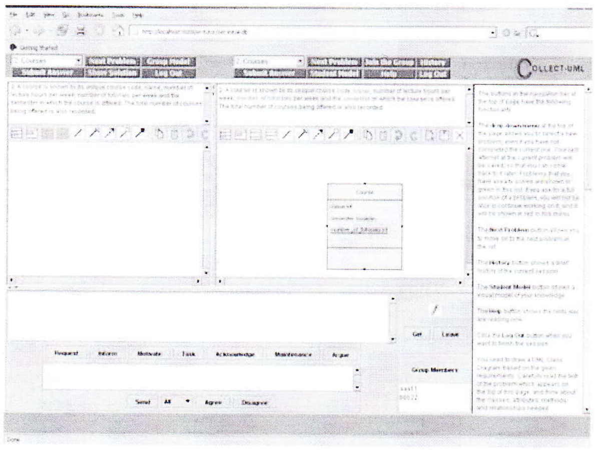

Figure 2. COLLECT-*UML* interface

Much research on learning has shown the potential effectiveness of collaboration for improving student's problem-solving skills. These benefits, however, are only achieved by active and well-functioning learning teams. An intelligent educational system therefore needs to provide support not only on the domain level, but also explicitly on collaboration.

The ultimate goal of COLLECT-*UML* is to support collaboration by modelling collaborative skills. The system is able to promote effective interaction by 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. Figure 3 illustrates an example of a meta-constraint. This constraint makes sure that the student does take part in solving exercises and/or chatting.

5. CONCLUSIONS AND FUTURE WORK

This paper presented the single-user version of COLLECT-*UML,* and the results of the evaluation study performed. The results of both subjective and objective analysis proved that COLLECT-*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.

```
(240
"Would you like to contribute to the group discussion?"
T
(or-p (match SC CLASSES (?* "@" ?class_tag ?*))
      (match SC METHODS (?* "@" ?method_tag ?*))
      (match SC ATTRIBUTES (?* "@" ?attr_tag ?*))
      (match SC RELATIONSHIPS (?* "@" ?rel_tag ?*))
      (match SC DESC (?* "@" ?tag ?*)))
"descriptions"
nil)
```

Figure 3. An example of a meta-constraint

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 collaborative version of COLLECT-*UML* will provide a measure of the effectiveness of using the CBM technique in intelligent computer-supported collaborative learning environments.

6. REFERENCES

[1] Baghaei, N., Mitrovic, A. and Irwin, W. *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.

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

[3] Constantino-Gonzalez, M., and Suthers, D. *Coaching Collaboration in a Computer-Mediated Learning Environment.* (CSCL 2002), (NJ, USA, 2002), pp.583-584.

[4] Dillenbourg, P. *What do you mean by "Collaborative Learning".* In Dillenbourg, P. (Eds.), Collaborative Learning: Cognitive and Computational Approaches, Amsterdam: Elsevier Science. pp.1-19, 1999.

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

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

[7] Mitrovic, A., Mayo, M., Suraweera, P. and Martin, B. *Constraint-based Tutors: a Success Story.* (IEA/AIE-2001), (Budapest, 2001), Springer-Verlag Berlin Heidelberg LNAI 2070, pp.931-940.

[8] Plaisant, C., Rose, A., Rubloff, G., Salter, R. and Shneiderman, B. *The design of history mechanisms and their use in collaborative educational simulations.* (CSCL 1999), (California, USA, 1999), pp.348-359.

[9] Reimann, P. *How to support groups in learning: More than problem solving.* In Aleven, V. (Eds.), Artificial Intelligence in Education (AIED 2003), (Sydney, Australia, 2003), pp. 3-16.

[10] Rosatelli, M., Self, J., and Thirty, M. *LeCs: A collaborative case study system.* 5th International Conference on Intelligent Tutoring Systems (ITS 2000), (Montreal, Canada, 2000), pp.242-251.

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

[12] Tedesco, P. and Self, J. A. Using meta-cognitive conflicts in a Collaborative problem solving environment. (ITS 2000), (Montreal, Canada,2000), pp.232-241.