Evaluating a Collaborative Constraint-based Tutor for UML Class Diagrams

Nilufar BAGHAEI and Antonija MITROVIC
Department of Computer Science and Software Engineering
University of Canterbury, Private Bag 4800, Christchurch, New Zealand
nilufar.baghaei@gmail.com, tanja@cosc.canterbury.ac.nz

Abstract. COLLECT-UML is a collaborative constraint-based tutor for teaching object-oriented analysis and design using Unified Modelling Language. It is the first system in the family of constraint-based tutors to represent a higher-level skill such as collaboration using constraints. We present the full evaluation study carried out at the University of Canterbury to assess the effectiveness of the system in teaching UML class diagrams and good collaboration. The results show that COLLECT-UML is an effective educational tool. In addition to improved problem-solving skills, the participants both acquired declarative knowledge about good collaboration and did collaborate more effectively. The participants have enjoyed working with the system and found it a valuable asset to their learning.

1 COLLECT-UML

COLLECT-UML [1, 2] is a web-based collaborative constraint-based tutor, teaching Object-Oriented (OO) analysis and design using Unified Modelling Language (UML). Constraint-based tutors have been successfully used in the past to support individual learning in a variety of domains. COLLECT-UML is the first constraint-based tutor supporting collaborative learning. This paper briefly describes the system and presents a full evaluation study conducted at the University of Canterbury to examine the system's effectiveness in teaching UML and successful collaboration. COLLECT-UML 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 started by developing a single-user version. The system was evaluated in a real classroom, and the results showed that students' performance increased significantly. For details on the architecture, interface and the results of the evaluation studies conducted using the single-user version, refer to [2].

The student interface is shown in Figure 1. 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 system provides feedback on the individual solutions, as well as on group solutions and collaboration. The domain-level feedback on both individual and group solutions is offered at four levels: *Simple Feedback*, *Error flag*, *Hint* and *All Hints*. The collaboration-based advice is given to individual students based on the content of the chat area, the student's contributions to

the shared diagram and the differences between student's individual solution and the group solution. For more details on the interface, refer to [1].

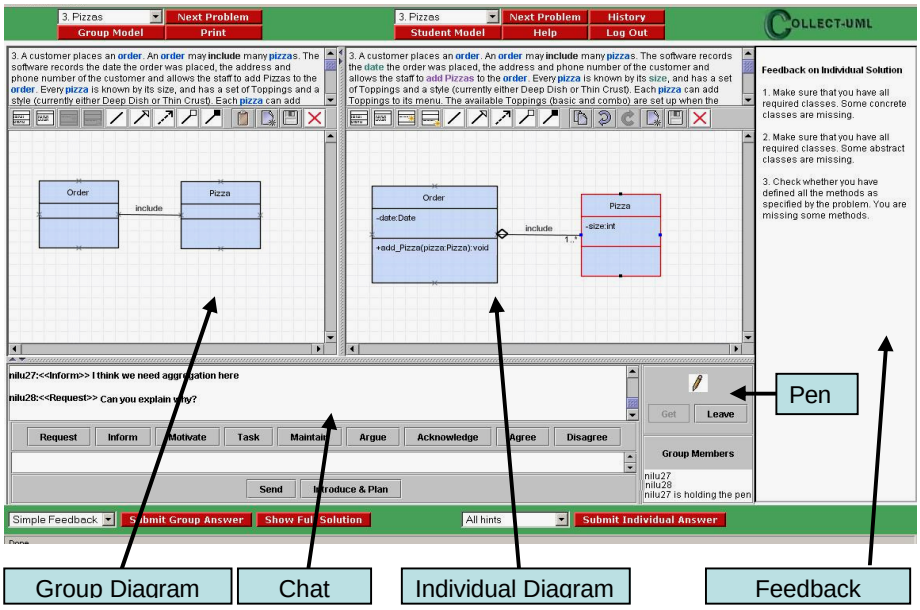

Figure 1. COLLECT-UML interface

The goal of our research is to support collaboration by modelling collaborative skills. COLLECT-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. For more details about the meta-constraints refer to [1].

2 Evaluation

We conducted an evaluation study 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 carried out 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 decide on the pairs and moderators.

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 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 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). 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.

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. The effect size on student's collaboration knowledge is also very high: 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 ($t = 2.03$, $p = 0.03$).

Figure 2 illustrates the probability of violating a meta-constraint plotted against the occasion number for which it was relevant, averaged over all meta-constraints and all participants in the experimental group. There is 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 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 shown by their higher scores 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 as they performed significantly better on the post-test. Finally, the students enjoyed working with the system and found it a valuable asset to their learning (as specified in the questionnaires filled out at the end of the session). The results, therefore, show that constraint-based modelling is an effective technique for modelling and supporting collaboration in CSCL environments.

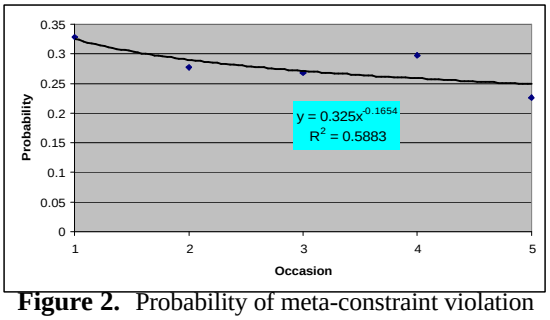

Figure 2. Probability of meta-constraint violation

References

[1] Baghaei, N., Mitrovic, A.: A Constraint-based Collaborative Environment for Learning UML Class Diagrams. ITS 2006, (2006) 176-186

[2] 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(1-2) (2006) (in print)