Evaluation of StudentUML: an Educational Tool for Consistent Modelling with UML

Dimitris Dranidis

City College, 13 Tsimiski Str, Thessaloniki, Greece, dranidis@city.academic.gr

Several CS courses adopt UML (Unified Modelling Language) in order to teach object-oriented analysis and design techniques. It is acknowledged that appropriate UML modelling tools have to be used in conjunction with the taught material so that students get hands-on experience with the tools and the practices of the trade. Professional UML tools tend to be too complex and lack educational features. StudentUML is a simple yet effective educational tool which supports the construction of consistent UML diagrams. The unique among similar tools capability of StudentUML to validate diagrams and check their consistency substantially supports students in the process of correctly assimilating the taught concepts. In this paper we present the results of experiments that have been conducted in order to evaluate the consistency checking feature of StudentUML.

Keywords

Educational CASE Tools, Learning UML, Object-oriented analysis and design

1. Introduction

Object-oriented programming has become the most prominent programming paradigm both in the industry and the academia. The Unified Modelling Language (UML) [1], being an industrial standard, is typically adopted in numerous CS courses such as object-oriented programming, object-oriented analysis and design, and software engineering. In these courses appropriate UML modelling tools have to be used in conjunction with the taught material so that students get hands-on experience with the tools and the practices of the trade.

StudentUML [2, 3] is an educational tool which supports the construction of valid and consistent UML diagrams. The aim of StudentUML is to provide the students with a simple yet effective tool that meets their learning needs without distracting them from the learning process with unnecessary features. Furthermore, the unique among similar tools capability of StudentUML to validate diagrams and check their consistency substantially supports students in the process of correctly assimilating the taught concepts. In this paper we present the results of experiments that have been conducted in order to evaluate the consistency checking feature of StudentUML.

The paper is structured as follows: Section 2 presents related professional and educational UML tools; section 3 describes typical student mistakes and the consistency checking feature of StudentUML; section 4 presents the results of the experiments conducted and section 5 concludes the paper.

2. Related UML Tools

Several studies [4, 5, 6] have shown that the use of modelling CASE tools in education improves the effectiveness of learning theoretical concepts. While some authors [7] support

the adoption of commercial tools for teaching students UML, with the argument that it is in the interest of students to have hands-on experience with the common tools of the trade, a number of other studies [8, 9, 10, 11, 12] have shown that professional tools are too complex to be suitable for educational purposes. Commercial tools usually aim to be fully compliant with the latest UML syntax and offer a large number of features, which result to a cluttered and confusing interface [13, 14]. In addition, some useful educational aspects such as consistency checking, inter-diagram conversions, educational hints, and illustration of learned concepts are absent, since these tools are targeted to professionals and not to students.

A very limited number of UML tools are available for educational purposes [10, 15, 11, 8, 14]. A detailed comparison of these tools can be found in [3]. Although, these tools have some useful educational characteristics, such as ease of learning, ease of use, limited subset of UML, illustration of learned concepts, and collaborative learning, they have serious shortcomings. None of them support a) interaction diagrams, which are among the most important UML diagrams for object-oriented design, and b) development projects, which consist of more than one diagram.

Furthermore, both professional and educational tools lack consistency and validation capabilities, which we consider as important educational features. Students are allowed to draw incorrect or inconsistent diagrams without getting any feedback from the tools. As a result, such tools discommode, instead of supporting the learning process, since students are left with the wrong impression that what they model is correct.

3. StudentUML's Consistency Checking

The most important educational feature of StudentUML is its ability to check the consistency of diagrams. Students frequently draw diagrams which might be correct when examined in isolation but wrong in respect to other diagrams in the same project. These consistency errors do not allow students to correctly implement their models. StudentUML offers the feature of automatically checking consistency between existing diagrams. The application presents the validation results in the forms of warnings and errors. Notice that the option of automatically fixing the errors (though not the warnings) is provided.

In the following we present some typical mistakes that students do when they design with UML and the results of the consistency checking performed by the tool.

Missing or misplaced methods in class diagrams. In the diagrams of Figure 1, method mtd is placed in class A although the sequence diagram suggests that it is a method of class B.

Missing relationships. Another error, which is illustrated in Figure 1, is the missing relationship between A and B classes in the class diagram, which is implied by the interaction between them in the sequence diagram. Note that depending on the context, as it is implied by one or more interaction diagrams, the relationship between classes A and B is either an association or a dependency. (In the above example it is an association)

Wrong relationship type between classes. This type of student mistakes is the most difficult to learn to avoid. These mistakes lead to semantically wrong diagrams which cannot be correctly forward-engineered to code. In Figure 2, the student has identified all classes and their methods correctly, but the relationships are wrong. Actually, there is no need for the association between A and B classes, since the sequence diagram implies that there is only local visibility from object of class A to the object b of class B. The correct relationship between A and B is a dependency and not an association.

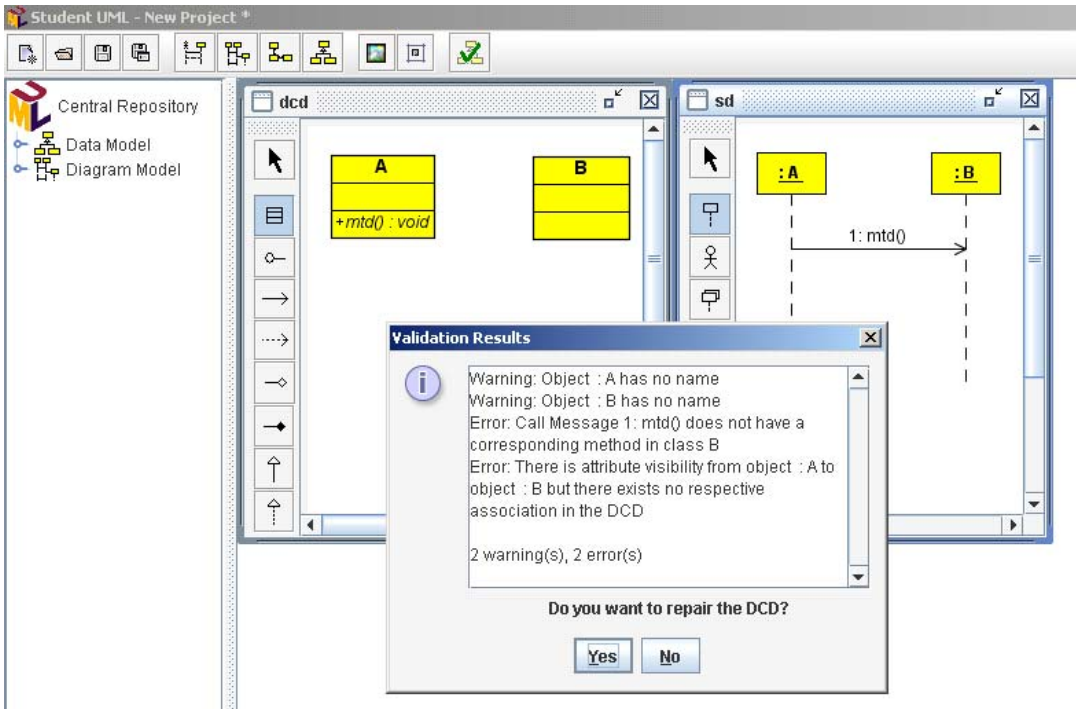


Figure 1. The Validation Results dialog box provides feedback to the student for the mistakes he has done in the diagrams. Note that warnings (not necessarily mistakes) are reported also.

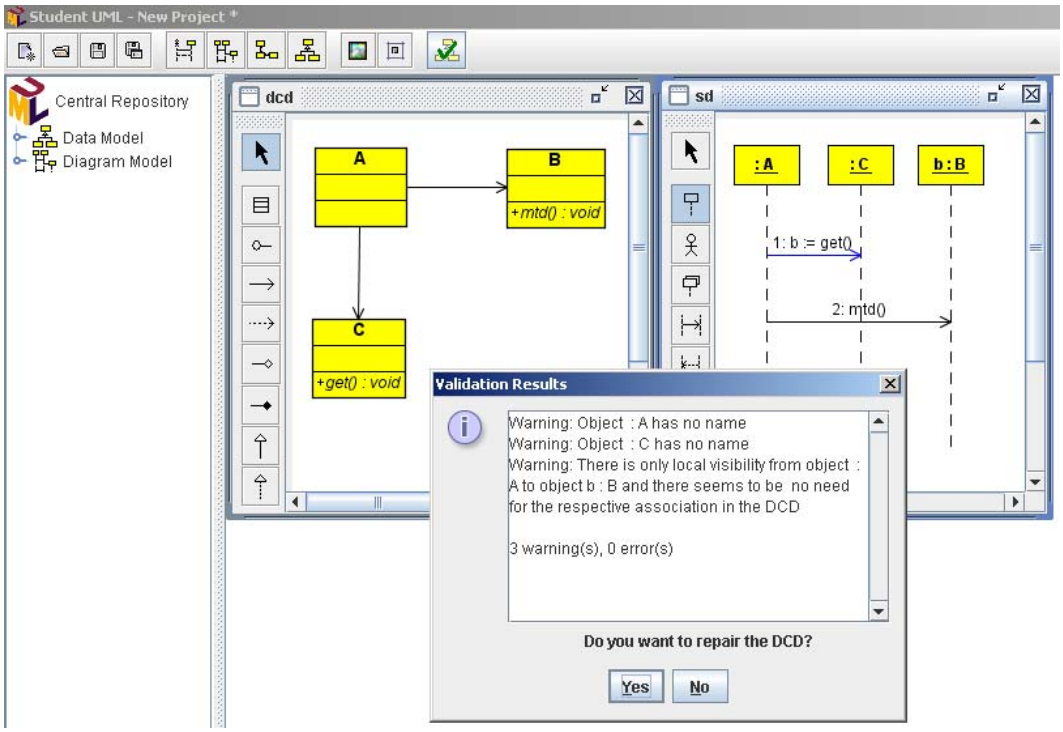


Figure 2 The Validation Results dialog box.

4. Evaluation Experiments

In order to evaluate the tool by its intended users, the students, experiments have been conducted with undergraduate students from the Computer Science at CITY College. The experiments had the following objectives:

The first objective of the experiments was to evaluate StudentUML in terms of its simplicity and ease of learning, relative to professional tools. A usability test was conducted with two groups of undergraduate students in computer labs. One group was given the task to draw several UML diagrams using StudentUML, and the other group using a professional tool (Academic Licence of MagicDraw UML). The time taken to draw the same diagrams correctly using the two different tools was measured. After completing the requested tasks, the students were also asked to complete a questionnaire regarding their experience with the tool.

The second, but most important, objective of the experiments was to assess the educational value of the tool. Before the lab, students were examined by taking a short multiple choice quiz involving concepts which are usually misconceived by students. During the lab, students were asked to perform some UML modelling tasks and use the validation and consistency checking feature of the tool in order to manually correct their errors by using the feedback provided by the tool. After the lab, students took the same multiple choice quiz. The results of both quizzes (before and after using the tool) were analyzed in order to examine whether the tool helped them in answering the questions correctly and eventually understanding the theoretical concepts better.

This paper presents the results of the experiment for the second objective, i.e. the assessment of the educational value of the tool.

4.1 Design of the experiment

The experiment involves one independent variable which is the time of the quiz placement with two nominal values: BEFORE and AFTER (using the tool) and one dependent variable which is the answer to each question of the quiz with two nominal values: CORRECT or WRONG. In order to analyze the results of the experiment a non-parametric test for two related samples is chosen; the McNemar test for significance of changes is used for assessing the results.

The null hypothesis (to be rejected) is: *There are no significant changes in the results of the quizzes before and after using the tool*.

The alternative hypothesis is: *There is a significant increase in the number of students who answered correctly after using the tool.*

4.2 Results before using the tool

The frequent student mistakes presented in a previous section are drawn from the author's experience in teaching object-oriented analysis and design. To strengthen the conclusions drawn from the experience a simple quiz has been performed with CS students at CITY College. The aim of the quiz was to demonstrate that students frequently misplace methods in class diagrams.

The sample of students participating in the quiz consisted of twenty (20) 1st level students and twenty-two (22) 2nd level students. Concerning their knowledge on the topic examined at the time the quiz took place, 1st level students had been recently (1 week before the quiz) exposed to the relevant theory (Systems analysis and design), while 2nd level students had a complete course (Systems analysis and design) during their first year and had practically

applied this knowledge in a 2nd level course (Software Development in Practice) where they used UML for a group-work software development project.

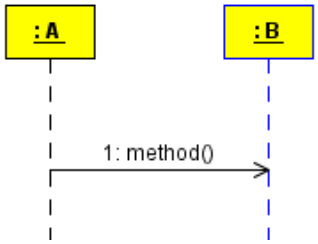
The quiz consisted of two multiple choice questions with a single correct answer. Both questions tested the same knowledge in a slightly different context.

First question

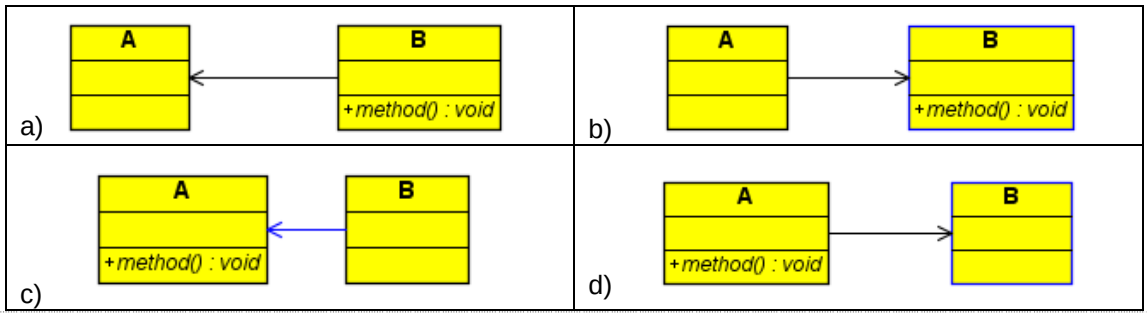
In the first question, the sequence diagram consisted of two objects and a single interaction via a method call (this is the simplest sequence diagram one can draw). The correct answer is (b); the method called *method()* belongs to class *B* and the association arrow should point towards the class *B*.

QUESTION 1

Given the following sequence diagram:



Which of the following design class diagrams is correct?



The results of the answers of the students to the first question are summarized in Table 1.

Table 1 Answers of the students to the first question before using StudentUML.

	a		b		c		d	
1st level			7	35%			13	65%
2nd level			8	36%	2	9%	12	55%
Total			15	36%	2	5%	25	59%

Only 15 (7+8) out of 42 students (36%) answered correctly. It is interesting to observe that there was no significant difference between the results of 1st and 2nd level answers; one would expect that 2nd level students would perform better.

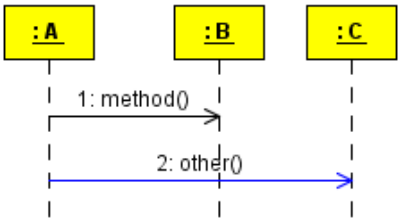
By further investigating the wrong answers, out of the 27 students who answered wrongly, in 25 (13+12) of the cases (93%) the answer was (d): the association arrow is correctly directed but the method is placed in the wrong class. Even in the case of the two students who answered (c) the method is misplaced.

Second question

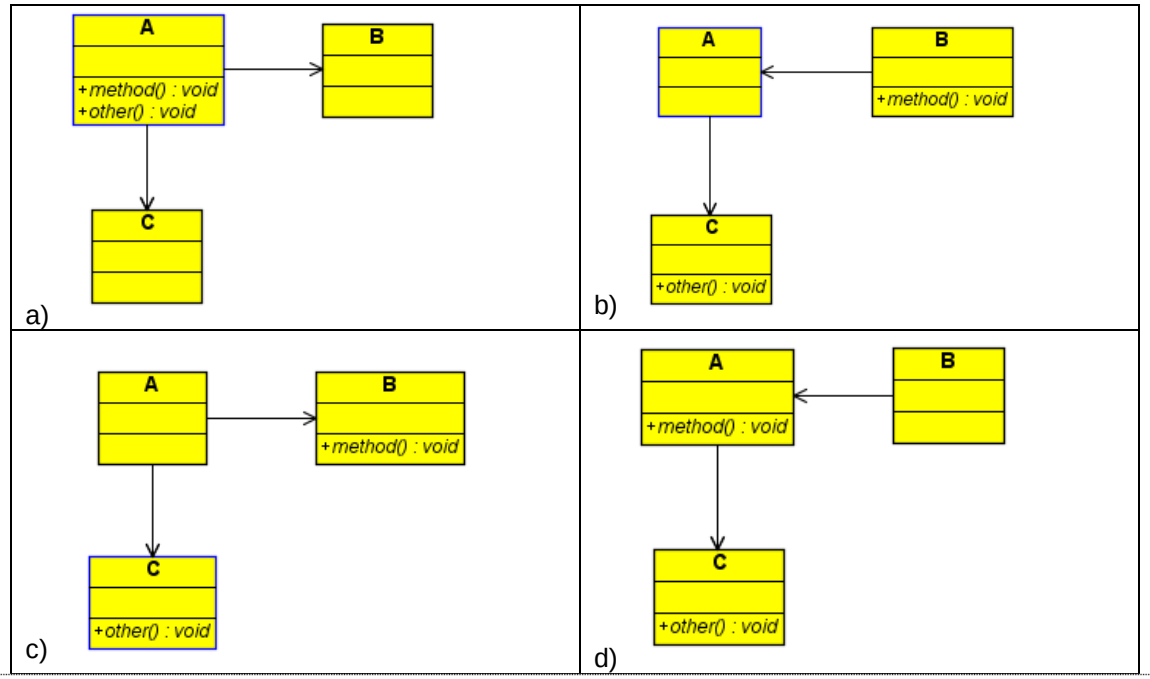
The second question of the quiz examined the same knowledge in a slightly more complex context. The sequence diagram consisted of three objects each one of different class and two operations. The correct answer is (c); the method named *method()* belongs to class *B* and the method named *other()* belongs to class *C*. Both association arrows should point towards classes *B* and *C*.

QUESTION 2:

Given the following sequence diagram:



Which of the following design class diagrams is correct?



The results of the answers of the students to the second question are summarized in Table 2.

Table 2 Answers to second question before using StudentUML.

	a		b		c		d	
1st level	10	50%			10	50%		
2nd level	11	50%			10	45%	1	5%
Total	21	50%			20	48%	1	2%

Only 20 out of 42 students (48%) answered correctly. There was again no significant difference between the results of 1st and 2nd level students.

By further investigating the wrong answers, out of the 22 students who answered wrongly, in 21 (13+12) of the cases (95%) the answer was (a): the association arrows were correctly directed but both methods were placed in the wrong classes. In the single wrong answer (d) still one of the methods is misplaced.

Surprisingly, students performed better in the second question (48%) than in the first question (36%) which was undoubtedly easier at least in terms of the objects and methods present in the respective diagrams. A possible explanation could be the order of the questions. From the fifteen (15) students who answered correctly the first question only, two (2) answered wrongly the second one; the rest (13) answered correctly both questions. Seven (7) students, who answered wrongly the first question, answered correctly the second. It might be the case that they realized the error they have done. Note that students were not allowed to change their answers once answering the first question. Another possible explanation could be the richer context of the second question. What we might regard as a more complex context could have caused students more thinking and eventually the conclusion to the correct answer.

From the results of the quiz, one may draw the conclusion that misplacing methods is indeed a frequent student mistake. The task of creating a design class diagram from a sequence diagram may not be a trivial problem for the students.

4.3 Results after using the tool

After the students took the short quiz, they were given about an hour to work with the StudentUML tool. Students performed several tasks in which they had to draw class diagrams and sequence diagrams. Occasionally students were asked to use the support of the consistency checking feature in order to correct their design mistakes.

After using the tool, the students were asked to complete the same short quiz again (the answers were reordered to avoid memorizing their first answer).

The results of both questions are provided in Tables 3 and 4. Note, that although in the quiz the answers were reordered for the sake of the comparison we use the same answer letters as in the first quiz.

Table 3 Answers to the first question after using StudentUML.

	a		b		c		d	
1st level			12	60%			8	40%
2nd level	1	5%	12	55%			9	41%
Total	1	2%	24	57%			17	40%

Table 4 Answers to the second question after using StudentUML.

	a		b		c		d	
1st level	7	35%	1	5%	12	60%		
2nd level	9	41%	2	9%	10	45%	1	5%
Total	16	38%	3	7%	22	52%	1	2%

First question

As a first observation, there was an increase in the overall percentage of correct answers (Before the tool: 36%, after the tool: 57%). However it is more important to examine whether individual students improved. Table 5 lists the results of student performance in the first question both before and after using the tool;13 students answered correctly the question both before and after using the tool, 16 students answered wrong both before and after using

the tool, 11 students answered wrong before and correct after using the tool, and 2 students answered correct before and wrong after using the tool.

Table 5 Comparisons of answers to first question before and after using StudentUML.

	Before	
After	Correct	Wrong
Correct	13	11
Wrong	2	16

After performing the McNemar test on the above data we may conclude that:
There was a significant increase in the number of students who answered correctly the first question after using the tool (McNemar test, significance level p = 0.022).

Second question
Unfortunately we may not draw a similar conclusion from the analysis of the second question. In this case the increase was very small (Before the tool: 48%, after the tool: 52%). Table 6 summarizes the comparison results.

Table 6 Comparisons of answers to second question before and after using StudentUML.

	Before	
After	Correct	Wrong
Correct	14	8
Wrong	6	14

After performing the McNemar test on the above data we conclude that:
There was no significant difference in the number of students who answered correctly after using the tool (McNemar test, significance level p = 0.791).

Although, in the simple context of the 1st question the tool helped the students to improve their answers, it did not influence significantly the results of the 2nd question. Possible reasons for that result may be (a) the order of questions (significant increase of correct answers before using the tool in the 2nd question) and (b) the feedback provided by the tool in cases like the second one is more cluttered; students have difficulties in understanding the reported errors.

5. Conclusions

StudentUML is a simple UML modelling tool with the capability to validate diagrams and check their consistency. This unique feature substantially supports students in the process of correctly assimilating the taught concepts. The experiments conducted at labs with CS students have indicated that StudentUML's consistency checking feature helps students in identifying and correcting their modelling mistakes and helps students in reinforcing correct diagrammatic techniques. This result is more evident and statistically significant in simple cases. Work needs to be done to improve the tool's feedback mechanism so that structured and helpful feedback will be provided to the students in more complex cases. We strongly believe that this enhancement will improve the educational value of the tool.

Acknowledgements

The author would like to thank all CITY students from the 1st and 2nd level of the BSc in Computer Science who participated in the experiments.

References

1 Object Management Group (OMG). OMG Unified Modeling Language Specification. Version 2.0, Document formal/05-04-01, http://www.omg.org/technology/documents/formal/uml.htm, 2006.
2 Dranidis D, Ramollari E. Learning object-oriented modeling with a UML learning assistant tool, in Proceedings of the 4th International Conference on Informatics, Educational Technology and New Media in Education, Sombor, Serbia, April 2007.
3 Ramollari E, Dranidis D. StudentUML: An Educational Tool Supporting Object-Oriented Analysis and Design, in Proceedings of the 11th Panhellenic Conference on Informatics (PCI 2007), May 2007.
4 Burton P J, Bruhn R E. Using UML to facilitate the teaching of object-oriented systems analysis and design, JCSC, vol. 19, no. 3, 2004, pp. 278-290.
5 Mynatt B T, Kleventhal L M. A CASE primer for Computer Science educators, in Proceedings of the twentieth SIGCSE technical symposium on Computer science education, Louisville, Kentucky, USA, ACM Press, 1989, pp. 122-126.
6 Ho Y C. To what extent will CASE tools assist users in the systems development – A case study in academic environment, in Proceedings of the 1992 ACM SIGCPR conference on Computer personnel research, Cincinnati, Ohio, USA, ACM Press, 1992, pp. 93-96.
7 Smith H H. On tool selection for illustrating the use of UML in system development, JSCS, vol. 19, no. 5, 2004, pp. 53-63.
8 Auer M, Tschurtschenthaler T, Biffl S. A flyweight UML modeling tool for software development in heterogeneous environments, in EUROMICRO'03, 2003.
9 Buck D, Stucki D J, Design early considered harmful: Graduated exposure to complexity and structure based on levels of cognitive development, in Proceedings of the 31-st SIGCSE Technical Symposium on Computer Science Education, Austin, Texas, USA, 2000.
10 Crahen E, Alphonce C, Ventura P. QuickUML: A beginner's UML tool, in OOPSLA '02, Seattle, Washington, USA, 2002.
11 Turner S A, Perez-Quinones M A, Edwards S H. minimUML: A minimalist approach to UML diagramming for early Computer Science education, Computing Research Repository, http://arxiv.org/abs/cs.HC/0603121, 2005.
12 Iivari J. Why are CASE tools not used, Communications of the ACM, vol. 39, no. 10, 1996pp. 94-103.
13 Flint S, Gardner H, Boughton C. Executable/Translatable UML in computing education, in Proceedings of Sixth Australasian Computing Education Conference (ACE2004), Dunedin, New Zealand, 2004.
14 Hansen K M, Ratzer A V. Tool support for collaborative teaching and learning of object-oriented modeling, in Proceedings of ITiCSE '02, Aarhus, Denmark, 2002.
15 Alphonce C, Ventura P. QuickUML: A tool to support iterative design and code development, in OOPSLA '03, Anaheim, California, USA, 2003.