Developing a blended learning strategy for teaching Object-Oriented Programming using the 'Model First' approach

Roula Georgantaki[1], Yannis Psaromiligkos[2], Symeon Retalis[1], Vassilis Dendrinos[2], Dimitris Adamopoulos[1]

[1]University of Piraeus, Karaoli & Dimitriou 80, 18534 Piraeus, Greece, {rgeo@unipi.gr, retal@unipi.gr & dxa@otenet.gr}

[2]Technological Education Institute of Piraeus, 250 P.Ralli & Thivwn, 12244 Piraeus, Greece, {jpsa@teipir.gr, vasden@otenet.gr}

This paper proposes a sequence of learning activities (i.e. a learning script) supported by networked technologies for the application of the "model-first" approach in teaching object oriented programming (OOP). The "model-first" approach has been proposed and applied by various researchers and it seems promising for augmenting the learning effectiveness of OO programming instructional practices. This approach focuses on the conceptualisation of OO design models as well as the OO software development process based on these models, prior to exposing students to the programming language's syntactic and semantic details. However, there is a lack of systematic evaluation studies of the application of this specific approach via a specific sequence of learning activities. The paper proposes such a sequence which involves three main phases: (i) observation of the development process of an exemplar OO software application (ii) problem solving tasks with guided instructions, and (iii) autonomous problem solving task. The proposed sequence of learning activities was tested during a two and a half months seminar for postgraduate students showing promising results.

Keywords

Blended Learning, Learning Strategy, Object Oriented Programming, Model First

1. Introduction

Although the OOP paradigm and its concepts reflect the "real world" which consists of objects that have attributes and perform actions, it has been proven that learners find hard to conceptualise the OOP philosophy and its concepts [1], [2], [3], [4], [5], [6]. Students undoubtedly need to overcome a lot of barriers during the instructional process of OOP such as to understand new concepts (e.g. object, class, attributes, encapsulation, inheritance, polymorphism), to learn an OO programming language, etc [7]. Various approaches have been proposed for augmenting the learning effectiveness of OOP instructional practices such as the adoption of "objects-first" strategy [8], the gradual explanation of concepts from simple to higher level ones [9], the use of guidelines like "Don't start with a blank screen", and "Read code" [8] etc.

In [10] a "design driven" approach is presented for a CS1-CS2 object-oriented courses' sequence, and claims the treatment of several disadvantages of programming-first

approaches noted in [11]. This approach, focuses on design issues of a OO software application rather than teaching the syntax of an OO programming language, and encompasses components of design using a subset of UML, design patterns and non-trivial problems to motivate students. This approach stresses the importance of design throughout CS1 and continues this emphasis in CS2 having students working in teams on a semester-long project. This approach is indented to develop mainly design skills to the students that undoubtedly are critical for their academic course and professional career.

Other researches, mainly conducted under the "Comprehensive Object Oriented Learning" (COOL) project [12], propose and describe the "modelling" approach to teach object-orientation in CS1 and present as critical perspective the conceptual modelling [13], the integration of conceptual modelling and coding and the explicit focus on revealing the programming process [13] (Bennedsen & Caspersen, 2000b). These researches refer the conceptual modelling as a lacking perspective in the suggestions for CS1 made in [11].

The "model-first" approach has also been proposed and applied by various researchers [9], [13], [7], [14], [15], [16]. This approach seems to be effective since it focuses on the conceptualisation of OO design models as well as the OO software development process based on these models, prior to exposing students to the programming language's syntactic and semantic details [14]. Most of the above mentioned studies about teaching OO design and programming with the exception of the COOL research project, conducted at Oslo University [12], and a two-year study in high school students learning OOP [3], had a limited scope and did not made evaluation of the OOP learning process for a long time period. This fact is also advocated in [3]. In general, very few systematic and complete studies about the effectiveness of technology enhanced learning approaches have been reported in the literature [17].

Although the "design approach" as well as the "model-first" one seem effective, there are not many learning scripts, i.e. a sequence of learning activities that have to occur at each phase of the learning process, which could guide teachers how to utilize them in real educational setting. A learning script should have the following five attributes [18]:

- the sequence and timing of each phase
- the learning activities that students and teachers have to perform at each phase,
- the mode of teacher-students and student-student interaction (face-to-face, asynchronous, text-based or voice-based, …)
- the mode of interaction among students and learning resources and learning tools

This paper proposes a learning script supported by networked technologies for teaching OOP. It has specific pedagogical characteristics. It can act as an implementation guide of the "model-first" approach in OOP teaching. We also present its application during a seminar where we systematically evaluated the proposed learning script using the CADMOS-E evaluation method [19]. The innovative aspects of our approach are: i) the proposed sequence of blended learning activities based on the "model-first" approach; ii) the emphasis on the pedagogical strategy of cognitive apprenticeship [20], [21]; and iii) the combinatory utilization of CASE and programming tools at specific points of the learning process.

2. Analysis and Design of the Teaching Strategy

Our approach is comprised of blended learning activities that occur during three main phases (see Figure 1): (i) observation of the development process of an exemplar OO software

application (ii) problem solving tasks with guided instructions and (iii) autonomous problem solving task. The blended learning activities involve activities that use online learning resources, student's interaction with CASE tools, asynchronous collaboration with peers and tutors and traditional face to face classroom experiences.

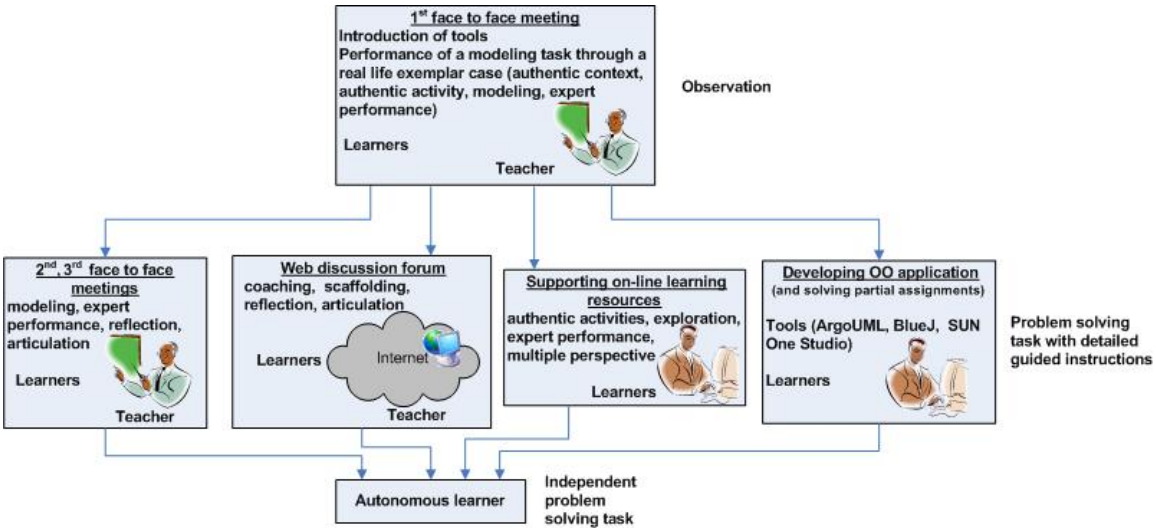

Figure 1 Phases and learning activities of the proposed learning script for OOP.

More specifically, during the "Observation" phase students observe the teacher, who plays the role of an expert software engineer, how he analyses, designs, implements, debugs and executes a simple OO application which is related to an authentic case close to the interests and social context of the students [20], [21]. He also analyses the thinking process leading to specific design and development choices [25]. This is a very important phase because the teacher-expert articulates his tacit knowledge [26] (making it explicit and codifying it) which is a difficult but very important for learning purposes task [27]. This phase usually happens in classroom where the teacher unfolds the process of solving a problem and not only the solution itself utilising a set of CASE tools and making references to the main OOP concepts [9]. In fact, the teacher's lecturing is usually video-recorded in order to serve as an online learning resource for future reference by the students.

The next phase of this strategy concerns the students' problem solving task with detailed guided instructions. Students should try to internalise the tacit knowledge they have just listened to. So they are asked to develop and deliver a small OO software application following the exact same steps of the expert. Students submit their application in three assignments-deliverables at pre-determined deadlines. Thus, their applications are gradually developed step by step, permitting students to understand how the various OO concepts are interrelated and applied in OO software development process. During this process detailed instructions about OO design and programming, i.e. scaffolds, are given to them [20]. These scaffolds support active and exploratory learning thus offering students the opportunity to explore concepts from multiple perspectives and also to see and consolidate experts' solutions to problems [28]. While students study the learning resources, and explore the initial exemplar case, they are able to communicate with the teacher and other peers using a web discussion forum. Teacher supports the learning process through scaffolding and coaching methods offering hints, reminders, feedback, recommendations, and suggestions [20], and encourages discussions that reveal the process driving to the solution [9].

During the last phase – the autonomous problem solving task – students are asked to design and develop a software application without necessarily following detailed instructions. This

phase tries to examine whether students have acquired knowledge, skills and autonomy in problem solving, through observation and supportive practice respectively in previous phases.

2.1 Learning Resources

The learning resources are mainly on-line integrated into an on-line Learning Management System such as the Moodle system (http://www.moodle.org). The resources offered are:

- the video-taped step-wise teacher's explanation of the OO software application development process of an exemplar OO application
- brief pieces of theory accompanied by representative examples from everyday life
- short exercises for further practice about OO concepts (not obligatory) with their solutions
- well-written code examples
- a well-documented case study of an OO software application. It contains the requirements specification, the OO analysis, the design decisions, the class diagrams, the source code, and testing cases
- study guides included in each didactic unit and other informative material such as tools' installation and usage manuals, links to other resources on Java language, etc.

The learning resources are structured into didactic units that concern the fundamental OO concepts-principles that students encounter during the observation phase. There are seven didactic units as following:

- Introduction to the Object Oriented Programming philosophy
- Object, Class, Attributes, Methods, diagrammatic presentation
- Abstraction, Encapsulation – information-hiding – Separation of behavior and implementation
- Objects' creation – Memory allocation – Constructor – References to objects
- Static variables and methods
- Methods overloading
- Class hierarchies – Inheritance – Polymorphism – Methods overriding – Dynamic Binding

Moreover, as scaffolds for knowledge construction we offered i) an asynchronous web discussion forum where students could post questions that the teacher/expert or other peers could answer thus forming a community of practitioners, and ii) two face to face meetings for addressing learning difficulties in person.

2.2 Integration of Tools

Our approach combines tools at specific points of the learning process. During the "Observation" phase the expert (teacher) utilizes the ArgoUML CASE tool (http://argouml.tigris.org/) which is a quite user friendly, relatively simple and "light" open source tool, for supporting the main design process of the exemplar case and the BlueJ educational environment (http://www.bluej.org) for further development of the initial exemplar OO software application. The latter offers graphical representation, simplicity and interactivity [8], [29] and also force modeling [14], [30]. Finally, the teaching of the implementation process of the exemplar application utilizes the SUN One Studio (http://java.sun.com/). The underlying environment was used as the proper instructional medium to demonstrate the creation, compilation, and execution of the "completed" exemplar OO software application

(with a “main” method). Thus students learn how to use a “professional” development tool and acquire an overall picture about program flow [5].

3. Evaluation Study

The evaluation study followed a specific methodology, called CADMOS-E [19] which is a stepwise method supported by specially developed “pre-test” and “post-test” questionnaires, which provide data for both quantitative and qualitative analysis. The focus of the evaluation was on the learning effectiveness of our approach and it was conceptualized as being related to a multiple measurement index consisting of cognitive and attitudinal outcomes e.g. students’ comprehension of basic OO concepts-principles, acquisition of OOP programming skills, self-estimation about their knowledge level in OOP and their feelings and attitudes towards OOP. All the variables are composite and are measured by multiple items, each measuring a slightly different aspect of the main variable.

3.1 Subjects

The underlying script had been applied in a two and half month seminar for postgraduate students at the Technological Education Institute of Piraeus. Eighteen (18) students registered for the seminar (17 men, 1 woman). Most of them had been previously taught OOP using VB.NET in the “Software Development” course of their Msc program on “Information Technology”. Two students had not participated in this course but they had some experience with OOP via informal training, and two students had not at all previously got in touch with OOP. All of them were very interested in the OO subject, since they considered this knowledge very helpful in accomplishing the learning goals of other courses in the MSc curriculum.

3.2 Instruments for Data Collection

The study was based upon two kinds of questionnaires that were given to the students. The first kind (“pre-test”) consisted of 40 questions and it was given to the students at the end of the first face to face meeting after they have listened to the seminar’s instructional philosophy. The second questionnaire (“post-test”) consisted of 53 questions and it was returned by the students after the end of the seminar. Some questions of the “pre-test” were replicated in the “post-test” in order to measure the seminar’s effect. However, the second questionnaire mainly consisted of a wide number of closed-end questions that were used to evaluate the contribution of various factors to the seminar’s effectiveness. The answers in closed-end questions were measured in a five-point Likert-type. It is also included a section with a number of open-ended questions to supplement the quantitative data. The open-ended section is related to students’ likes and dislikes towards the learning resources, the deficiencies concerning the resources and the approach and suggestions for improving either of them.

We also collected students’ opinion using focus group interviews from three (3) students (randomly chosen). The main purpose of the focus group interviews was to verify various findings that appeared form the analysis of the students’ questionnaires. In general, due to inherent difficulties in performing evaluation in general, and evaluation in technology enhanced learning in particular, the mixed evaluation method is the most appropriate one [31], [32].

3.3 Data Analyses

Since the size of the sample was not statistically appropriate for quantitative analysis we performed a comparative statistical analysis of the data collected from the "pre-test" and "post-test" questionnaires. The basic statistical analysis, which was conducted, depicted the trends of the learners' opinion concerning the self-estimation of students' level in OOP, the learning effectiveness of the seminar's instructional approach, the seminar's effect to the students' attitudes about OOP, the evaluation of the quality of learning resources, and the evaluation of the effectiveness of used environments.

The first goal of the evaluation study was to investigate the learning effectiveness of the proposed technology enhanced learning script. Overall, the analysis of the students' assignments (i.e. the three deliverables of their project) revealed that students conceptualized well the OO concepts and acquired abstract knowledge that was applied successfully to the new problem. The great majority was able to model an OO software application, to design class diagrams in ArgoUML tool, and successfully implement their assignment using the BlueJ tool, first, and the SUN One Studio, at the end. Students' performance in a scale from 0 to 100 is shown in Figure 2.

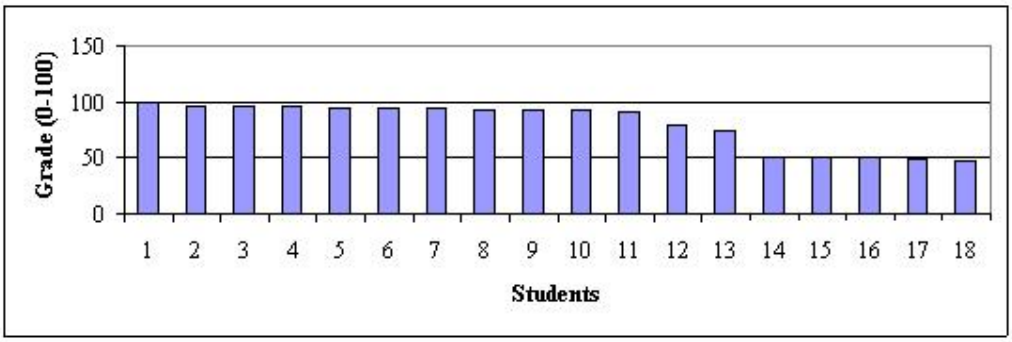

Figure 2 Students' grades for the final assignment.

As it can be seen, the great majority of students achieved a very good grade, indicating the learning effectiveness of our strategy. Concerning the students who achieved a grade around the "pass" level, they had time constraints and hence, they could not to deliver the final part of their projects in a complete form. Actually, only one student failed to successfully complete the seminar.

Table 1 Students' self-estimation about their level in OOP

Question	Mean (from pre-test)	Mean (from post-test)
How do you rate your level about OOP?	2.72*	2.83*
How would you assess your ability to write programs in Java?	-	3.22**
Based on what you have been taught till now, how well do you think that you have learned OOP?	3.06*	3.33***

* Answers' coding: 5= Professional, …, 1= Novice
** Answers' coding: 5= Very much capable, …, 1= Not at all capable
*** Answers' coding: 5= Very much, …, 1= Not at all

Additionally, students estimated in the pre-test questionnaires that their competence in OOP and their ability to write programs in Java would have been at a good level at the end of the instructional process. As stated in the post-tests (and verified by students' grades), their

knowledge level was even better than expected after the end of seminar as shown in Table 1. Of course, these statistics show some trend since the number of students who participated in the evaluation study was relatively small.

Furthermore, as shown in Table 2, the students' feelings towards OOP remain highly positive at the end of the seminar and the degree of students' satisfaction was enough high. Although students' fears concerning OOP seems to be a little higher after seminar's attendance, they stated that seminar's instructional approach offered them great help in overcoming their difficulties and fears. Students, when asked at the interviews, stated that to become a competent OO programmer demands a lot of effort, and this can be explained with the opinion that students performed a lot of learning activities in OOP during the seminar and thus they formulated a clearer opinion about this unquestionably demanding subject.

Table 2 Students' feelings and attitudes towards OOP.

Question	**Mean (from pre-test)**	**Mean (from post-test)**
Do you appreciate the cognitive subject of OOP?	4.39	4.23
Have the reasons that brought you to attend the seminar been satisfied?	-	3.50
The seminar aided me to get over flaws concerning OOP	-	3.67
The seminar aided me to get over fears that I was feeling concerning OOP	-	3.22
Do you have difficulties with OOP?	2.78	2.77
Do you fear OOP?	1.89	2.17
Do you think that OOP demands much competence and a lot of effort?	3.44	3.56

Answers' coding: 5= Very much, …, 1= Not at all

As already mentioned, with this evaluation study we also wanted to check which factors contribute to the acquisition of knowledge and skills in OOP. Thus, we asked students before and after the seminar to estimate the importance of various factors to the enhancement of their knowledge and skills in OOP. The mean values of their answers are shown in Table 3.

Table 3 Contribution of various factors to the enhancement of knowledge and skills.

Question	**Mean (from pre-test)**	**Mean (from post-test)**
Studying the case study.	4.50	4.39
Participating in the discussions on the subject matter with other students via Moodle.	4.22	3.72
Participating in the discussions on the subject matter with teachers via Moodle.	4.56	4.28
Solving exercises in the context of the seminar (partial assignments).	4.39	4.28
Performing the assignment in steps.	4.50	4.00
The hands-on practice with programming environments/tools.	4.50	-
Studying the "step-by-step" structured learning material that introduced us gradually from simple to more complex concepts.	-	4.28
Studying the learning material with the specific organization in each didactical unit (short theory, code examples, exercises with solutions, activities in educational environments).	-	4.44
Attending the face to face meetings of the seminar and the lectures during these meetings.	-	3.89

Answers' coding: 5= Absolutely important, …, 1= Totally unimportant

As it can be seen from the above table, all the factors are highly appreciated (mean values > 4.00 almost in all cases). Some factors had been rated a bit lower in "post-test" questionnaire than in the pre-test one. This is explained by the fact that students usually (and naturally) have particularly high expectations at the beginning of a seminar (especially when an innovative approach is about to be applied). The students' involvement in learning activities utilizing the educational tools, the variety of the learning resources, the case study, the teacher's support through web discussion, the scaffolding organization of the learning material (from simple to more complex) and the assignments that students performed during the second step of the process was really appreciated. As these results depict as well as the students' viewpoints as gathered during the interviews, all these factors offered great help and were significant scaffolds for students while performing learning tasks.

Students highly appreciated our choice to introduce them in OO philosophy through the analysis and modeling of a real life exemplar case. Characteristically, a student expressed his enthusiasm very vividly after having attended the analysis of the "Street market" application during the first meeting, saying that only with this exemplar case he comprehended all about OOP philosophy. That student had previously taught OOP and Java and had spent a lot more time studying those course subjects than the duration of the face to face meeting. He had highly appreciated the chosen exemplar case from everyday life and also the chosen tools. He mentioned: "I feel that I have discovered the new World inside me". All the above, show that students liked the chosen way for introducing them in OOP, and also that the applied learning strategy was effective with respect to the learning outcomes and the students' feelings.

The students, when asked to rate the quality of learning resources as a whole, they highly appreciated them, with a mean value of quality 4.06 (5= High quality, …, 1= No quality). More analytically, when they answered about the contribution of various learning resources to the enhancement of knowledge and skills, the following mean values resulted (Table 4).

Table 4 Contribution of learning resources to the enhancement of knowledge and skills.

Question	**Mean (from pre-test)**
Exercises with solutions, examples.	4.39
Theory.	4.17
The activities in the BlueJ environment.	4.17
The case study.	4.06
The study guides included in each didactical unit.	4.06
The completed OO application that we developed as assignment.	4.06
The ready made BlueJ projects.	3.94
Java source code files included in the material.	3.89
Questions, exercises	3.72
The activities in the ArgoUML environment	3.28

Answers' coding: 5= Absolutely important, …, 1= Totally unimportant

As it can be seen, the majority of the learning resources offered great help to students basically due to their high quality. Students did not highly appreciate the use of the ArgoUML environment. This finding in combination with some students' answers in related questions showed that although they appreciated the task to design class diagrams and create code skeletons for their applications using a CASE tool, they faced some usability problems when using this tool. Nevertheless, all students managed to submit their designs which most of them were correct.

From the interviews and the open-ended questions of the “post-test” questionnaire, some students’ dislikes about the seminar’s learning strategy emerged, mainly concerning with its duration and their obligations. Some students said that seminar should have lasted more time for better consolidation of the subject. They also stated that more examples in using ArgoUML should be needed, for its better utilization.

3. Conclusions

In this paper we presented a blended learning strategy for teaching Object-Oriented Programming using the “Model First” approach which was highly appreciated by the students as the evaluation of the pilot study showed. Our approach integrates three innovative aspects: i) a reusable blended learning script based on the “model-first” approach, ii) the emphasis on the pedagogical strategy of cognitive apprenticeship, and iii) the utilization of CASE tools and programming environments at specific points of the learning process.

Our future plan is to perform some kind of calibration of our technology enhanced learning script by applying extended tests (both to undergraduate and postgraduate students). We also plan to apply the script in different OOP languages. We have already developed such a script for teaching OOP using the VB.NET language showing promising results too. Also, we plan to enrich the script using an extension (plug-in) in BlueJ educational programming environment for classes’ creation, which will enable users to design systems using the UML notation, and will generate classes’ code with code skeletons for further editing in the BlueJ text editor. The ultimate aim is to ask students to use only the BlueJ environment throughout the learning process since it is a tool highly appreciated for its simplicity and usability. Thus, the usability problems of the ArgoUML that some students mentioned will be vanished.

References

1 Teif, M., & Hazzan, O. (2004). Junior High School Students’ Perception of Object Oriented Concepts. Paper presented at 18th European Conference on Object-Oriented Programming, 8th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, Oslo, Norway.

2 Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding object misconceptions. Proceedings of the 28th SIGCSE, 131–134.

3 Ragonis, N., & Ben-Ari, M. (2005). A Long-Term Investigation of the Comprehension of OOP Concepts by Novices. Computer Science Education, 5 (3), 203-221.

4 Fleury, A. E. (2001). Encapsulation and reuse as viewed by java students. ACM SIGCSE Bulletin, 33 (1), 189 – 193.

5 Ragonis, N., & Ben-Ari, M. (2005). On Understanding the Statics and Dynamics of Object-Oriented Programs. ACM SIGCSE’05, 226-230.

6 Milne, J., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming – Views of Students and Tutors. Education and Information Technologies, 7 (1), 55-66.

7 Schulte, C., & Niere, J. (2002). Thinking in Object Structures: Teaching Modelling in Secondary Schools. Paper presented at 16th European Conference on Object-Oriented Programming, 6th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, Malaga, Spain.

8 Kölling, M., & Rosenberg, J. (2001). Guidelines for Teaching Object Orientation with Java. Proceedings of 6th IT iCSE, Canterbury, 33–36.

9 Bennedsen, J., & Caspersen, M. (2004). Teaching Object-Oriented Programming – Towards Teaching a Systematic Programming Process. Paper presented at 18th European Conference on Object-Oriented Programming, 8th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Ori-ented Concepts, Oslo, Norway.

10 Alphonce, C. and Ventura, P. J. (2002). Object-Orientation in CS1-CS2 by Design. In Proceedings of Innovation and Technology in Computer Science Education, Aarhus, Denmark, June 24-26, 70-74.
11 CC2001 Computing Curricula 2001 (final report) (2001). The Joint Task Force on Computing Curricula (IEEE Computer Society and Association for Computing Machinery), December 15. Retreived January 27, 2007, from: http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/cc2001.pdf
12 Fjuk, A., Karahasanovic, A., Kaasbøll, J. (Eds.) (2006). Comprehensive Object-Oriented Learning: The Learners Perspective, Informing Science Press, California
13 Bennedsen, J., & Caspersen, M. (2004). Programming in context: a model-first approach to CS1. ACM SIGCSE Bulletin, 36 (1), 477−481
14 Berge, O., Borge, R. E., Fjuk, A., Kaasbøll, J. & Samuelsen, T. (2003). Learning Object Oriented Pro-gramming. Norsk Informatikkonferanse NIK'2003, Tapir Akademisk Forlag, 37-47.
15 Groven, A., Hegna, H., & Smørdal, O. (2003). OO learning, a modelling approach. Paper presented at 17th European Conference on Object-Oriented Programming, 7th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, July, Darmstadt, Germany.
16 Karahasanović, A., & Holmboe, C. (2006). Challenges of learning object-oriented analysis and design through a modeling-first approach. In Fjuk, A., Karahasanovic, A., Kaasbøll, J. (Eds.) Comprehensive Object-Oriented Learning: The Learners Perspective, Informing Science Press, California. (pp. 49−66).
17 Wallace, R. (2003). Online learning in higher education: a review of research on interactions among teachers and students. Education, Communication & Information, 3(2), 241-280
18 Dillenbourg, P. (2002). Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed). Three worlds of CSCL. Can we support CSCL (p. 61-91). Heerlen, Open Universiteit Nederland.
19 Psaromiligkos, Y., & Retalis, S. (2003). Re-Evaluating the Effectiveness of a Web-based Learning System: A Comparative Case Study. Journal of Educational Multimedia and Hypermedia, AACE, 12 (1), 5-20.
20 Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L.B. Resnick (Ed.), Knowing, learning and instruction: Essays in honour of Robert Glaser, Hillsdale, NJ: LEA, pp. 453-494.
21 Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18 (1), 32-42.
25 Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. American Educator, 6-46.
26 Reber, A.S. (1993). Implicit Learning and Tacit Knowledge: An Essay on the Cognitive Unconscious. Oxford University Press, New York, NY.
27 Hislop, D. (2002). Mission impossible? Communicating and sharing knowledge via information transfer. Journal of Information Technology, 17 (3), 165-177.
28 Hadjerrouit, S. (1999). A Constructivist Approach to Object-Oriented Design and Programming. Proceedings of the 4th ITiCSE, 171-174.
29 Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. Journal of Computer Science Education, Special Issue on Learning and Teaching Object Technology, 13 (4), 249–268.
30 Borge, R. E., & Kaasbøll, J. (2003). What is "OO first"? Submitted as a position paper for the 7th Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts at 17th European Conference on Object-Oriented Programming, July, Darmstadt, Germany.
31 Goodyear, P., Banks, S., Hodgson, V. & McConnell, D., eds (2004), Advances in Research on Networked Learning, Dordrecht: Kluwer
32 Treleaven, L. (2004). A New Taxonomy for Evaluation Studies of Online Collaborative Learning. Online Collaborative Learning : Theory and Practice (Roberts, T.S. ed., pp. 160-180). Hershey: Information Science Publishing.